Karlsruher Institut für Technologie

Video-based Bed Monitoring

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)
genehmigte

Dissertation
von

Manuel Martinez

aus Barcelona, Spanien

Tag der mündlichen Prüfung:	28. Juli 2017
Hauptreferent:	Prof. Dr.-Ing. Rainer Stiefelhagen Karlsruher Institut für Technologie
Korreferent:	Prof. Dr.-Ing. Jürgen Beyerer Fraunhofer-Instituts für Optronik, Systemtechnik und Bildauswertung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig angefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie die wörtlich und inhaltlich übernommen Stellen als solche kenntlich gemacht und die Regeln zur Sicherung guter wissenschaftlicher Praxis des Karlsruher Instituts für Technologie beachtet habe.

Karlsruhe, den 29. November 2017

Abstract

Quality of sleep is difficult to quantify, yet it critically affects our life in many aspects.

Many sleep disorders have effective treatments, but first we need to diagnose them, and this is performed in hospitals using a test named polysomnogram. It involves sleeping while being connected to more than 100 sensors that perform a plethora of tests including electroencefalogram, electrocardiogram, electrooculogram, and electromyography. Not only are polysomnograms intrusive, but they are also expensive, and therefore are only performed on the most severe patients.

A non-invasive way to monitor sleep in domestic environments would revolutionize sleep medicine, but developing such a system is challenging and requires collaboration between the fields of sensor engineering, machine learning and sleep medicine.

In this thesis, we present a computer vision system designed to monitor beds that leverages on the recent developments in depth camera technology and machine learning algorithms. Compared to other sleep monitoring technologies, computer vision gives us essential context information that allows us to better understand the environment. Furthermore, we worked closely with sleep medicine experts that guided our efforts and provided us insights about the topic.

Our first thesis contribution is the sensing device we developed, that has been certified for installation in hospitals and nursery homes. While developing this device, we thoroughly analyzed depth camera technology, developed a new communication protocol that stresses reliability, and developed new compression algorithms, resulting in publications in different areas.

Using our recording device, we collected several datasets totaling more than 5000 hours of video. One of them is recorded in a sleep hospital on real patients, and includes polysomnogram data for reference. Another is captured in a nursing home and is used for long term sleep quality summaries. To the best of our knowledge, our datasets are the largest ever used in the topic by a significant margin, and are an important contribution of this thesis. The datasets are large enough to train machine learning methods and obtain statistically significant results.

We also present a method for processing depth video targeted to our scenario named Bed Aligned Maps (BAM). Depth cameras are popular for sleep monitoring because they are robust to texture and illumination changes. However they still suffer many of the problems related to video monitoring: dealing with changes in camera or bed positions, low information density, and privacy concerns. To alleviate those problems, we propose the BAM, which is a low resolution descriptor based on

depth images that uses the bed position for anchoring, and solves the alignment and the privacy issues while being 500 times smaller than the source depth data.

Unlike most sensing approaches that are targeted to monitor a single sleep quality aspect, computer vision is less specific and can be used to monitor many of them. First, we analyze respiration, as it is the vital sign most related to sleep disorders, and in particular to sleep apnea. This sleep disorder, is characterized by various kinds of interruptions and other alterations of the normal respiration pattern, and its diagnostic is one of the main targets of the polysomnography test.

To analyze respiration, we first develop a mathematical model of the problem based on signal processing that reveals how different camera parameters affect the signal-to-noise ratio of the chest motion. We show that the signal-to-noise ratio diminishes by a factor of the fourth power of distance between the patient and the camera. This explains why most methods stop working if the camera is more than one meter away from the patient. We present new methods to estimate the respiratory rate of a patient that are reliable even if the camera is placed 4 meters away from the patient's chest, and can be used as well to find the patient in the image. Finally, we present a machine learning method designed to recognize apnea events with promising results.

Next, we analyze a variety of sleep quality cues that have been highlighted by our medical advisors. We present novel metrics for agitation and bed occupation that are simple, reliable, and objective. We show how to use those metrics to summarize in a visual way sleep patterns during long term periods. We analyze the task of recognizing bed related actions, like entering or leaving the bed, changing sleep positions, interacting with a nurse, or manipulating objects. We show a machine learning method to recognize sleep position that performs better than the sensor used in our reference sleep laboratory. Finally, we discuss the task of recognizing between sleep and awake stages.

This thesis aims to make sleep monitoring technology accessible to everyone. Our ultimate goal is that no sleep disorder should remain untreated simply because it is not identified in time.

Kurzzusammenfassung

Die Quantifizierung der Schlafqualität ist eine schwierige Aufgabe, dennoch ist sie ein wichtiger Bestandteil unseres Wohlbefinden und Lebensqualität.

Für viele Schlafstörungen existieren bereits effektive Behandlungsmethoden, aber zuerst müssen diese diagnostiziert werden, in Krankenhäusern geschieht dieses durch ein Polysomnogramtest. Dazu müssen Patienten mindestens eine Nacht im Krankenhaus verbringen und an mehr als 100 Sensoren angeschlossen sein. Diese Sensoren können unter anderem Tests wie das Elektroenzephalogramm, Elektrokardiogramm, Elektrookulogramm oder das Electromyographie durchführen. Derartige Tests sind nicht nur aufdringlich, sondern auch teuer, und werden deshalb nur bei den schwerwiegendsten Fällen eingesetzt.

Eine nicht-invasive Methode der Schlafüberwachung, die in einer häuslichen Umgebung ausgeführt werden kann, würde daher die Schlafmedizin revolutionieren. Die Entwicklung eines solchen Systems ist jedoch sehr kompliziert, da eine enge Zusammenarbeit zwischen Sensorexperten und Schlafmedizinern erfolgen müsste, welche in der heutigen Forschung jedoch kaum anzutreffen ist.

Wir entwickeln Systeme zur Schlafüberwachung, welche aktuelle Methoden im Bereich des Machinellen Lernens sowie Entwicklungen von Tiefenkameras nutzt. Durch den Einsatz von Computer Vision Techniken wird der erforderliche Kontext geschaffen, um die aktuell überwachte Situation zu verstehen. Zusätzlich haben wir eng mit Schlafexperten zusammengearbeitet, die unsere Forschungen begleiteten.

Unser erster Beitrag der Thesis ist die Sensorbox, welche für eine Installation in Krankenhäusern und Altersheimen zertifiziert wurde. Während der Entwicklung haben wir Tiefenkamera Technologien analysiert, ein neuartiges Kommunikationsprotokoll und Kompressionsalgorithmus entwickelt, was in Publikationen in verschiedenen Bereichen resultierte.

Mit Hilfe unsere Sensorbox haben wir mehrere Datensätze gesammelt, welche insgesamt mehr als 5000 Stunden Videoaufzeichnungen enthalten. Einer der Datensätze wurde in einem Krankenhaus mit echten Patienten aufgenommen und beinhaltet Polysomnogramdaten. Ein weitere Datensatz wurde in einem Altersheim aufgenommen, um uns eine langfristige Übersicht der Schlafqualität zu liefern. Unsere Datensätze sind, soweit uns bekannt, die größten Datensätze in der Forschung und bilden damit einen wesentlichen Beitrag zu unserer Thesis. Beide Datensätze reichen aus, um Methoden des Machinellen Lernens zu benutzen und damit statistisch signifikante Ergebnisse zu erzielen.

Weiterhin präsentieren wir eine Methode, um Tiefenaufnahmen für unser Szenario zu verarbeiten: *Bed Aligned Maps (BAM)*. Tiefensensoren sind bei der Schlafüberwachung beliebt, da sie robust gegenüber Textur- und Beleuchtungsveränderungen sind. Dennoch sind auch diese anfällig für viele Probleme der Videoüberwachung: kleine Kamera- und Bettposition Veränderungen, niedriger Informationsgehalt, und Privatsphärebedenken. Um diesen Problemen entgegenzuwirken, haben wir die BAM entwickelt, ein auf Tiefendaten basierender niedrigauflösender Deskriptor, der die Bettposition als Referenz benutzt und die Zentrierungs- und Privatsphäreprobleme löst, während er um ein 500-faches kleiner ist, als die originalen Tiefendaten.

Im Gegensatz zu den meisten sensorbasierten Ansätzen, die als Ziel die Beobachtung eines einzelnen Schlafaspektes haben, kann Maschinelles Sehen viele Aspekte gleichzeitig überwachen. Zuerst führen wir eine Atemmuster-Analyse durch, da Atmen das wichtigste Anzeichen für Schlafstörungen ist, insbesondere bei Apnoe. Diese Schlafstörung beinhaltet verschiedene Arten von Unterbrechungen und Abweichungen des normalen Atemrhythmus und deren Diagnose ist daher eine der Hauptziele der Polysomnografie.

Um die Atmung zu analysieren entwickeln wir zuerst ein mathematisches Model des Problems, basierend auf der Signalverarbeitung, welches aufzeigt, wie verschiedene Kamera Parameter das Rauschverhalten der ermittelten Brustbewegung beeinflussen. Wir zeigen, dass dieses Rauschverhalten sich antiproportional in der vierten Potenz bezüglich des Abstandes zwischen Patient und Kamera verhält. Das erklärt, warum die meisten Methoden nicht mehr funktionieren, wenn die Kamera mehr als einen Meter vom Patienten entfernt ist. Wir entwickeln neue Methoden zur Atemrhythmusschätzung eines Patienten, die auch dann noch zuverlässing sind, wenn die Kamera 4 Meter von der Brust des Patienten entfernt ist und zusätzlich den Patienten im Bild lokalisieren. Abschließend präsentieren wir ein Maschinelles Lernverfahren speziell dazu entwickelt, um das Vorkommen von Apnoe mit vielversprechenden Ergebnissen zu erkennen.

Weiterhin analysieren wir diverse Hinweise für die Schlafqualität, die durch unsere Schlafexeperten empfohlen wurden. Wir präsentieren neuartige Metriken für die Agitation und Bettbelegung, die einfach, zuverlässig und objektiv sind. Wir zeigen, wie diese Metriken genutzt werden können, um Schlafmuster über einen längeren Zeitraum hinweg zu visualisieren. Wir analysieren weiterhin bettbezogene Aktionen, wie z.B. das Ein-/Aussteigen, wechseln der Schlafposition, interagieren mit einer Krankenschwester oder die Benutzung diverser Objekte. Wir entwickeln ein Maschinelles Lernverfahren um die Schlafposition zu erkennen, welches besser als der im Schlaflabor benutzte Sensor arbeitet. Abschließend diskutieren wir die Aufgabe, zwischen Wach- und Schlafzustand zu unterscheiden.

Das Ziel dieser Thesis ist, Schlafüberwachungstechnologie für jeden zugänglich zu machen. Unser ultimatives Ziel ist, daß keine Schlafstörung unbehandelt bleibt, nur weil sie nicht rechtzeitig identifiziert und korrekt diagnostiziert wurde.

Acknowledgments

An incredible number of people has supported me and paved the way towards this thesis. To avoid making the acknowledgements larger than the thesis itself, I will simply say:

Thanks, to each and every one of you.

List of Abbreviations

AASM American Academy of Sleep Medicine

ANN Artificial Neural Network

AR autoregressive

ARM Advanced RISC Machine

BAM Bed Aligned Map

BGA Ball Grid Array

BM Block Matching

BoW Bag of Words

BPM Breaths per Minute

CE Cross Entropy

CMOS Complementary Metal-Oxide-Semiconductor

CNN Convolutional Neural Network

CPLD Complex Programmable Logic Device

CPU Central Processing Unit

CUDA Compute Unified Device Architecture

CV:HCI Computer Vision for Human-Computer Interaction

DAC digital-to-analog

dBAM difference Bed Aligned Map

DOG Degree of Freedom

DSP Digital Signal Processor

DW Durbin-Watson

ECG electrocardiography

EEG electroencephalogram

EHS Evangelishe Heimstiftung GmbH

EMC Electromagnetic Compatibility

EMD Earth Mover's Distance

EMG electromyography

EM Expectation-Maximization

EOG electrooculography

ESD Electrostatic Discharge

FC Fully Convolutional

FFT Fast Fourier Transform

FOV Field of View

FPGA Field-Programmable Gate Array

fps frames per second

GPIF General Programmable Interface

GPU Graphical Processing Unit

GRU Gated Recurrent Unit

HEMD Hierarchical Earth Mover's Distance

HoG Histogram of Oriented Gradients

HT High Throughput

ICU Intensive Care Unit

JTAG Joint Test Action Group

QSXGA Quad Super Extended Graphics Array

KLD Küllback-Leibler divergence

kNN k-Nearest Neighbors

LDA Linear Discriminant Analysis

LED Light-emitting Diode

LLC Locality-constrained Linear Coding

LMNN Large Margin Nearest Neighbor

LSTM Long Short-Term Memory

MGM Modified Geometric Moments

MLP Multi-Layer Perceptron

MP Megapixel

MRD Medical Recording Device

MSE Mean Squared Error

NIR near infrared

NREM Non-Rapid Eye Movement

NUC Next Unit of Computing

PCA Principal Component Analysis

PCB Printed Circuit Boards

PLL Phase-locked Loop

POV Point of View

PSD Power Spectral Density

PWM Pulse Width Modulation

RBF Radial Basis Function

ReLU Rectified Linear Unit

REM Rapid Eye Movement

RERA Respiratory Event Related Arousal

RNN Recurrent Neural Network

RAII Resource Acquisition Is Initialization

RAM Random Access Memory

RFID Radio-Frequency IDentification

ROI Region of Interest

ROS Robot Operating System

RR Respiration Rate

SAAS United Kingdom Sleep Assessment and Advisory Service

SD Sinkhorn Distance

SDR Software Defined Radio

SGD Stochastic Gradient Descent

SPHERE Schlafüberwachung im Pflege- und Heimbereich mittels Remotesensorik

SVM Support Vector Machines

SXGA Super Extended Graphics Array

TOF Time-of-Flight

TCP Transmission Control Protocol

THX Thoraxklinik-Heidelberg GmbH

USB Universal Serial Bus

UWB Ultra Wide Band

VGA Video Graphics Array

VESA Video Electronics Standards Association

VHDL VHSIC Hardware Description Language

VF Variable to Fixed

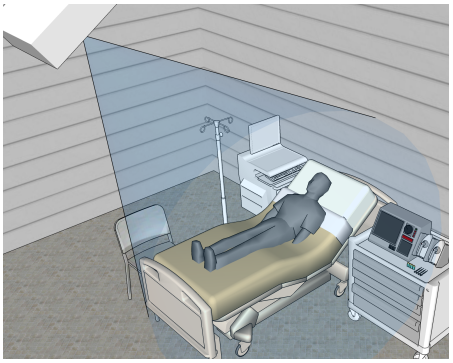
VIPSAFE Automated Visual Monitoring for Improving Patient SAFEty

VQ Vector Quantization

Contents

1. Introduction

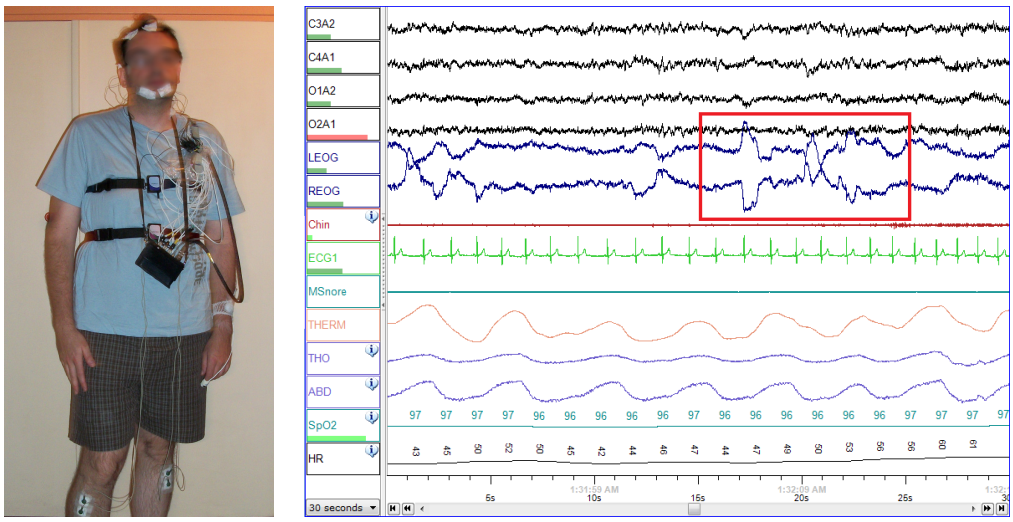
1.1. Motivation



We present a health monitoring system that uses a camera to watch the bed of a sleeping person and recognize breathing patterns, sleep positions, agitated behaviors, and actions such as getting in and out of the bed. We aim at several target scenarios and applications including but not limited to: detecting dangerous behavior in Intensive Care Units (ICUs), quantifying sleep quality for the elderly in hospices and ageing-at-home scenarios, and helping in the screening and follow-up procedures in sleep studies.

The main design goal of our system is to achieve the lowest possible intrusiveness. This target was originally dictated by our original scenario: the ICU. There, among other limitations, we were not allowed to interact or alter in any way the normal function of the ICU. However, low intrusiveness is also a critical requirement when monitoring elderly people. We want to preserve the pride of the patient who is, generally, an experienced adult person that has been taking care of himself and others for a long time and does not want to be watched like a child. Also, a system that requires interaction from the user by, *e.g.*, wearing a wrist band, may be simply forgotten after a few weeks, more so if the patient is affected by a senile ailment.

For low intrusive monitoring of elderly people, the bed is the golden monitoring target. Its main advantage is the fact that, regardless of how varied the daily routine of a person is, we tend to sleep at least once a day in the same bed, every single day. Therefore, many technologies have been designed around bed monitoring, like pressure mattresses, laser sensors that detect falls, and more recently ultrasound and microwave radars that monitor breathing patterns. Those technologies are readily available and in use in hospices, however they still have several problems. Currently available sensors have a very limited perception of the environment, thus rely on simplistic hypotheses that often fail in real live, triggering an excessive number of false alarms. Also, some sensors often require custom installation, calibration, and maintenance procedures, making it difficult to use them in the patients own home.



Source: `en.wikipedia.org/wiki/Polysomnography`

Figure 1.1.: Left: patient equipped with polysomnographic sensors. Right: polysomnographic record with eye movements highlighted, indicating REM stage.

Bed monitoring using cameras does not require to modify the bed, and can be installed almost anywhere. Cameras also provide a holistic view of the scene, thus allowing us to employ more intelligent algorithms that provide a better understanding of what is happening in the bed. On the other hand, monitoring humans using cameras is not trivial. Compared to a laser sensor that signals a fall simply when its beam is interrupted, we need a fully fledged machine learning system to process the camera stream and provide a similar output, and given the many variables that appear in unconstrained dormitories, this is not a simple task.

1.2. Scenarios and Use Cases

Most sleep monitoring systems are targeted to a single scenario, and often address only one task, like recognizing sleep position, detecting a person falling out of the bed, or to monitor breath rate, etc.

On the other hand, camera based systems offer a high degree of flexibility and thus can be used for many purposes. Here we analyze the target scenarios where a camera based monitoring system can be of use.

1.2.1. Sleep Medicine

The most common way to study sleep is a protocol named polysomnography, where more than one hundred sensors are attached to a patient and he is recorded during

an entire night in an hospital (see Fig. 1.1). Not only is the procedure cumbersome, but the polysomnographic record must be analyzed later by a specialist. As a consequence, sleep studies are expensive. Even though screening is performed before testing, long waiting lists are common.

Due to the exposed limitations, full sleep studies involving polysomnography are often restricted to the most grave patients. Those patients are generally diagnosed once, given a possible treatment option, and then their progress is monitored using very basic sensors in their own homes.

An automated sleep monitoring system that can be used in the patient's own home, even if it is not as reliable as a polysomnogram, would be of immense help in the whole pipeline. It would improve patient screening, and help on the follow-up monitoring of patients after diagnose. However, and most importantly, it would help to diagnose the patients that do not qualify for a full polysomnogram, and thus improve significantly their quality of live.

The analysis of the respiration patterns is an important focus of sleep medicine, and consequently a main target of this thesis, as one of the most common sleep disorders is sleep apnea (periods with dysfunctional breathing).

1.2.2. Intensive Care Units

In ICUs, patients are connected to multiple sensors that measure vital signs (breathing rate, pulse, blood pressure, blood oxygenation, etc.). Obtaining those vital signs from the camera is not necessary in this scenario.

One use case for intelligent perception systems in ICUs is to improve the management of accidents. Accidents in ICUs are rare, but their consequences are often severe, and it is crucial to detect them as soon as possible. Typical examples of accidents are when patients wake up from anesthesia feeling disoriented, and fall out of the bed, or when they remove their own infusion lines. Another event that requires immediate intervention is when a patient displays aggressive behavior, either towards itself or towards others.

A perception system could alert when an accident happens, although it would be even better if its able to predict when an accident is about to happen, in order to alert the nurses in time to take the necessary actions to prevent it.

A second use case is to help in managing *postoperative delirium*. Delirium, also known as acute confusional state, is a clinical syndrome usually characterized by a diminished mental state where the patient is not aware of his surroundings or is able to complete even the simplest tasks. Although statistics vary from hospital to hospital, Vasilevskis et al. (2012) rates the incidence rate for delirium as high as 80%. Citing Vasilevskis: *"a patient who experiences delirium is more likely to experience increased short- and long-term mortality, decreases in long-term cognitive function, increases in hospital length of stay and increased complications of hospital care"*.

The causes of delirium are not well understood, and its study is hampered by the fact that delirium can not be reliably diagnosed from the usual vital signs. A patient with a suspected diagnosis of delirium is asked to answer some basic questions to confirm the diagnostic. The most common protocol used to diagnose delirium is the Confusion Assessment Method-ICU test, introduced in (Inouye et al., 1990), and takes between 3 to 5 minutes to perform. To improve the diagnostic rates of delirium in ICUs, Fick et al. (2015) suggested a fast test to screen for delirium that only uses the following two questions: *"cite the months of the year backwards"* and *"what is the day of the week?"*.

Several studies suggest links between behavioral changes and delirium, but measuring behavior objectively is challenging (see Grap et al. 2011). Instead, a camera based perception system could be used to generate objective behavioral indicators.

1.2.3. Elderly Care

The second main scenario for sleep monitoring are hospices, nursing homes, and ageing-at-home initiatives.

In this scenario, the residents generally appreciate devices that provide cues that can be used to improve their sleep quality. The nurses, are interested in systems that alerts them in case of accidents, nightmares, panic attacks, coughing attacks, snoring, agitation, smoking and urination. Additionally, it would be interesting to trigger an alert if a resident is unable to change his sleep position by himself, as he can develop pressure ulcers in his skin, as noted in (Maklebust and Sieggreen, 1996).

Furthermore, the medics that design and manage those installations are interested in ways to improve the quality of life of the residents in the long term. Towards this goal, it is important to acquire long term statistics of the sleep quality as well as a record of the environment factors that can potentially affect sleep.

Finally, it must be noted that any monitoring system that triggers alarms must be very reliable. It is known that many sleep sensors trigger an excessive number of false positive alarms and induce alarm fatigue on the caregivers (see Cvach 2012).

1.3. Three Pillars of Vision-based Sleep Monitoring

There have been several attempts at monitoring sleep using computer vision, but still none of them have been successful in a practical setting. We argue that a successful monitoring system relies on three pillars: data acquisition and management, machine learning, and medical expertise. A lack of strength in any of those pillars seriously diminishes the performance of the final system, but, at the same time, being strong in the three pillars is difficult as they belong to significantly different research fields.

1.3.1. Data Acquisition and Management

The goal of this pillar is to provide raw data of sufficient quality in sufficient quantity.

If a sensor is too noisy or does not have enough resolution, the signal that we expect to recover would be degraded beyond recovery. However, acquiring high resolution images at low compression ratios generates large amounts of data that is difficult to capture, transport, store, and analyze. Therefore it is necessary to find the right balance between quality and quantity for each particular task.

In this thesis, our algorithms that analyze respiratory patterns have the finest requirements in terms of quality, to the point that we must store the depth images using lossless compression algorithms. Our sensor captures the equivalent of 1.5 TiB of data per night, that we compress online to 50 GiB.

Conversely, the tasks that are based on machine learning techniques, like, for example, recognizing sleep position, require as much data as we can acquire, and the information density of our data is very low (*e.g.*, we shift our position between two and four times per hour of sleep). Thus for this task we use a 5 TiB dataset that contains 99 recorded nights, equivalent to 800 hours of video.

1.3.2. Machine Learning

To extract information from the sensor feed we need to apply signal processing and machine learning techniques. Signal processing is far from trivial, but computer vision processing is a very specialized branch within signal processing with its own techniques. Inside computer vision there are two major fields: computer vision in industrial settings, and computer vision in human environments. It is precisely the computer vision in human environments that is relevant to this task, and it has seen the most advances in the last 5 years. In particular, since the re-introduction of deep learning (see Krizhevsky et al. 2012), the field has seen a huge revolution where the performance of computer vision systems went, for some tasks, from infra-human to super-human performance.

As the results of the system depends on machine learning, and the relevant algorithms are currently the subject of an intense study, it is imperative that a computer vision researcher who is familiarized with the latest technologies is involved in the research.

1.3.3. Medical Expertise

The sleep medicine specialization in Germany requires a minimum training time of 18 months[1]. It is not an easy discipline to learn. It is crucial to have the collaboration

[1] http://ic.daad.de/imperia/md/content/informationszentren/icdamaskus/facharztausbildung_engl.pdf

of a medical expert to design a sound data collection, define relevant experiments, and to ensure that our analysis is done correctly.

One of the most important roles for the medical expert is to design a sound evaluation protocol. The evaluation metrics used by medics and computer vision researchers are very different. Medical publications base their conclusions on tests of statistical significance, and finding which is the right test for each case is not trivial.

On the other hand, few medics are familiar with machine learning evaluation techniques, where the observed group must be split in non overlapping train and test groups in order to achieve relevant results.

Therefore a collaboration between both specialists is required to thoroughly evaluate sleep monitoring systems based on machine learning algorithms. In this thesis we have enjoyed a very successful collaboration with the sleep doctors from the Thoraxklinik-Heidelberg GmbH, who provided invaluable insight during our analysis and evaluations.

1.4. Contributions and Outline

The contributions in this thesis are the following:

Chapter 2: Current Trends in Non-Invasive Sleep Monitoring. In this chapter we review the state of the art in various non-invasive technologies to monitor sleep, including computer vision, and we analyze each publication on the base of two variables: how is the data processed, and the quality of their evaluation. Finally we compare them with our published contributions.

Chapter 3: Medical Recording Device. The Medical Recording Device (MRD), is one of the major contributions of this thesis. The MRD integrates multiple sensors as well as processing elements into a compact box that has been CE certified for installations in hospitals. This chapter describes all the sensors used in the MRD, the custom boards designed to house them, as well as the main algorithms developed to reliably capture its data. Furthermore, due to our high requirements with regards to image quality, we thoroughly analyzed the depth sensor used. This analysis is also a significant contribution and is discussed in appendix A. Finally, due to the need to compress high bandwidth data in real time, we developed new compression algorithms that are optimized for speed, including Marlin, which is discussed in appendix B.

Chapter 4: Datasets. Due to the low density of information in sleep, most research is the area is based on simulated environments or very small realistic datasets. Within the framework of this thesis we collected three datasets. The first is a small simulated dataset containing activities that happen rarely in real life settings. The second contains 99 sleep monitoring sessions of 80 real patients simultaneously annotated with a polysomnogram. The third dataset contains more than 5000 hours collected

in a nursery home. To the best of our knowledge, those are the largest datasets used for video-based sleep monitoring.

Chapter 5: Bed Aligned Map. The Bed Aligned Map (BAM) is another of the main contributions of this thesis, as it solves three of the main problems that video has for sleep monitoring: privacy issues, data size, and alignment. The BAM uses the bed to anchor the point cloud provided by the depth camera, and generates a low resolution description. Being low resolution, it is privacy friendly and also uses 500 times less space than the raw depth image. By being automatically aligned to the bed, it makes the installation of the sensor calibration-free. We use BAMs for most of our algorithms, as it is impractical to process all the original depth images due to their size.

Chapter 6: Analysis of Respiration Patterns. Respiration is the most important vital sign related to sleep quality, and thus our analysis of how respiration patterns can be recognized from depth cameras is one of our main contributions. Measuring breathing rate from a depth camera is simple in ideal conditions, when the measured person is healthy, stays quiet, and the depth camera is close to the chest, but those conditions rarely happen in real world scenarios. In this chapter we derive a mathematical formulation for the problem of measuring chest movements from a depth camera, as well as two new algorithms to monitor breathing rate, and evaluate them in real world conditions as well as in simulated extreme conditions, where they outperform the state of the art. We conclude the chapter suggesting new algorithms based on deep learning targeted specifically to detect breathing anomalies with promising results. While developing those algorithms, we contributed to the deep learning theory by developing a novel loss function, the relaxed earth mover's distance, which is discussed in appendix C.

Chapter 7: Analysis of Sleep Quality Cues. In this chapter we suggest a new metric for measuring sleep agitation that has a simple physical interpretation, and allows us to provide an objective measure of a usually considered subjective behavior. We also show new metrics for bed occupancy and facial stress, and use them to create a graphical summary of the sleep of a person during a few months in a single page. We follow up by developing machine learning techniques to classify actions that happen in a bed, and show that the classifier performs better if we split them into different complexity levels. We present a deep learning model that recognizes sleep and awake with a precision above 80%, and another that classifies sleep position better than the contact sensor used in a sleep laboratory.

Chapter 8: Conclusion. We summarize the main contributions made in the field of vision-based sleep monitoring and discuss possible directions for future work.

1.5. Previously Published Contributions

The contributions of this thesis have been published in different conference proceedings. Our MRD is discussed in (Martinez et al., 2016a, 2013b). Marlin, one of the compression algorithms developed in this thesis, was published in (Martinez et al., 2017a). Our analysis of the Kinect depth sensor was presented in (Martinez et al., 2014a, 2013c). Our BAM algorithm was presented in (Martinez et al., 2013a). Our analysis of breathing patterns have been published in (Martinez et al., 2017b; Benz et al., 2016; Martinez et al., 2012), while our analysis of sleep quality cues have been published in (Martinez et al., 2016a,a; Grimm et al., 2016; Martinez et al., 2013a,a,b). Our study of the Earth Mover's Distance (EMD) within the framework of Deep Learning was published in (Martinez et al., 2016c), and expanded in (Martinez et al., 2016b).

2. Current Trends in Non-Invasive Sleep Monitoring

Bed monitoring has gained importance in recent years and many papers have been published about the subject.

However, the topic has not generated a unified community, on the contrary, research comes from such diverse fields as sensors, embedded devices, mechatronics, electrical engineering, pervasive sensing, assisted living, sleep medicine and computer vision, among several others.

The variety of fields involved, and the lack of objective benchmarks and metrics to compare results between different approaches, has propitiated few cooperation and much work duplication. Hence, despite the large amount of published works, most areas are still in the early stages of development.

In this chapter we organize the review of the current state of the art based on the sensing technology used. In addition, we classify each paper into one of the following three categories:

Sensing: indicates that the target of the published method is to acquire a naturally occurring signal without further interpretation, *e.g.*, electroencephalogram (EEG), electrocardiography (ECG), electrooculography (EOG), electromyography (EMG), recovering chest movements, bed occupancy or agitation.

Ad hoc: shows that some sort of post-processing over the captured signal has been performed to obtain a higher level interpretation, *e.g.*, apnea events, sleep position, sleep stage, etc., but the method has been trained/designed and tested on the same patients.

Machine learning: also indicates that post-processing algorithms have been applied, but in this category the algorithms that are trained in a set of patients and tested in a different set. This separation between test and training sets is critical to obtain a realistic performance evaluation.

Furthermore, we note for each contribution, if it has been evaluated in a real or a simulated environment and we attach the number of participants used in the study, *e.g.*, **sim-7** indicates that the evaluation is performed in a simulated environment with 7 participants, while **real-100** indicates that 100 patients have been monitored in a real setting.

Finally, we discuss the common limitations observed in the state of the art, and we highlight how this thesis makes significant contributions to address those limitations.

2.1. Pressure Sensors

Pressure sensors integrated either in the mattress or the pillow are a popular way to monitor sleep because of their reduced cost. The most popular use case for this modality is to use a single sensor to simply monitor bed occupancy (*i.e.*, testing if the bed is full or empty), and this kind of beds are already in use in nursery homes.

On the negative side, pressure sensors are easily fooled, have very little context information, and need specific maintenance (*e.g.*, special cleaning procedures for the bed clothes), thus they are rare in domestic environments.

Due to their popularity, there have been several attempts to use pressure sensors for more advanced tasks. Harada et al. (2000) *{sensing, sim-2}* designed a pressure pillow to monitor breathing motion. Liu et al. (2013) *{ad hoc, sim-14}* used a dense grid of pressure sensors integrated in the mattress to classify sleep position. Rus et al. (2014) *{sensing, sim-2}* designed a new kind of sensor that detects capacitance instead of pressure and is thus less fooled by inert distractors, and they used them to sense sleep position. Samy et al. (2014) *{ad hoc, sim-7}* also used a dense grid of pressure sensors, this time in the form of a textile sheet, and used it to recognize sleep stages.

2.2. Intelligent Beds

Intelligent beds have built-in sensors and are popular in elderly homes.

Most related to our research is the work from Aoki *et al.* and his group, which has received very little attention outside Japan. Their work focuses on the creation of a depth camera whose design is functionally identical to a Kinect v1: a near infrared (NIR) laser is projected through a transmissive diffraction grating to generate a cloud of dots that are captured by a camera with a matching NIR filter. Due to the geometry involved, it is simple to recover the actual distance between the camera and each of the dots. To summarize: they invented a Kinect-class depth camera and used it to sense chest movements. The camera is integrated with the frame of the bed to solve alignment problems. This approach is explored in (Aoki et al., 2001) *{sensing, sim}* and (Aoki et al., 2003) *{sensing, sim}* where they sense chest movements, as well as the related work by Takemura et al. (2005) *{sensing, real-5}* where they use a similar setup on five real patients to record apnea events. Sadly, by 2012 their technology became obsolete due to the release of Kinect, and they migrated to the Microsoft camera in (Aoki et al., 2012) *{sensing, sim}*.

Also relevant in the category of intelligent beds is the work by Nishida and Hori (2002) *{sensing, sim-5}* where they use a bed with integrated pressure sensors and well as other ad hoc sensors to record respiration signals. And the work by Hoque et al. (2010) *{sensing, sim-10}* where the bed has integrated RFID antennas, the pijama has RFID tags, and together can recognize sleep positions.

2.3. Smartphones

We have covered previously sensing modalities popular in elderly homes, and thus, designed for professional environments. Conversely, smartphones have become a very popular way to monitor sleep in domestic environments. However, few of the sleep monitoring applications available in online stores have been thoroughly evaluated.

We start with a very simple but effective approach, Chen et al. (2013) *{ad hoc, sim-8}* monitor the sound levels to predict the amount of sleep, assuming that we do not make any noise while sleeping. Gradl et al. (2013) *{ad hoc, sim-10}* expects you to put the phone under your pillow before going to bed and uses its motion sensor to predict sleep-awake cycles. On the other hand, Hao et al. (2013) *{ad hoc, sim-7}* presented an approach with the commercial name of iSleep that aims to predict *"sleep quality"* using the microphone, and Gu et al. (2014) *{ad hoc, sim-45}* presented an approach with a similar goal but use microphones and motion sensors instead.

The most formal but also the most original contribution in the field comes by the hand of Nandakumar et al. (2015) *{sensing/ad hoc, real-37}*, who turns the phone into an active radar using the loudspeakers to emit an inaudible tone, and the microphone to capture it. They use the radar to sense chest movements and use an ad hoc system to classify between apnea gravity indexes on real patients.

2.4. Actigraphy

Actigraphy is the science of measuring human movements. Actigraphic sensors (or monitors) are usually in the form of wristbands and count the number of movements that the sensor registers. Due to the simplicity of the metric captured, the reduced hardware cost, and the minimal burden that the device causes, it has been one of the preferred ways to monitor sleep in the last years.

(Cole et al., 1992) *{ad hoc, real-41}* is the seminal work that links actigraphy and sleep analysis, in particular, to predict sleep/awake cicles, and its metrics are still considered almost state-of-the-art. Later research revealed that pure actigraphic sensors have a very low specificity and thus are not indicated for medical use, therefore actigraphy has been combined with ECG in (Devot et al., 2010) *{ad hoc, real-35}*, (Rofouei et al., 2011) *{ad hoc, sim-1}*, and (Grap et al., 2011) *{ad hoc, sim-30}*.

2.5. Pressure and Video

For the specific task of recognizing sleep position, it is popular to combine pressure sensors and computer vision technologies, as in (Huang et al., 2010) *{machine learning, sim-3}* and (Torres et al., 2016) *{machine learning, sim-5}*, however in both cases the tested environment was far from realistic.

2.6. Ultra Wide Band Radar

There is a specific sensing technology that has been designed explicitly to monitor breathing patterns: Ultra Wide Band (UWB) radars on chip. Those radars measure distances and velocities of objects in a close neighborhood (although they can not point directions). Most of the publications regarding those radars focus on the technical development of the technology (*e.g.*, Zito et al. 2011), instead of viability or usability studies, so its actual capabilities are not well known. In any case this has not hindered the release of consumer products based on the technology, like the OMR (2012) which is a small device with the form and size of an alarm clock that analyzes your sleep patterns.

Although interest on UWB radars faded when smartphones occupied their niche, a few new works have been published recently. Lee et al. (2014) *{sensing, sim-6}* tests UWB radars on 6 volunteers to sense *movements*, Rahman et al. (2015) *{machine learning, sim-8}* use random forests to predict sleep stage, and Lin et al. (2016) *{ad hoc, sim}* presents SleepSense, a commercial product, citing from their abstract: *"In the 75-minute sleep study, SleepSense demonstrates wide usability in real life."*.

2.7. Other Sensors

Scalise et al. (2011) *{sensing, real-6}* used a laser vibrometer to measure accurately the chest movements of neonatal infants, and Liu et al. (2015) *{sensing}* use the surprising fact that our bodies reflect high frequency WiFi signals from our domestic routers, and our chest movements change, very slightly the path of the WiFi signals. Using this principle, they show how to recover respiration movements by using a few routers and monitoring their signals.

2.8. Computer Vision

Finally, we review the state of the art in sleep monitoring using computer vision.

The first attempts at using computer vision for sleep monitoring were limited to very simple tasks and measured either the amount of pixels that changed between images or the amount of movement (*i.e.*, optical flow). Those attempts started with (Nakajima et al., 2001), that used a NIR camera to monitor posture changes and estimate the respiratory rate of a single subject that wears clothing with a very specific mosaic pattern. Then, Chase et al. (2004) *{sensing/ad hoc, real-5}* used also the difference between consecutive images to measure agitation in ICUs on color cameras. Similarly, Liao and Yang (2008) *{sensing, sim-10}* measured movement overnight using NIR cameras, and Reyes et al. (2010) *{sensing, sim-1}* measured agitation using difference of images on a color camera.

Probably based on those simple technologies, a sleep monitoring product based on computer vision was released: Secumatic (2006), but its specifications have not been made public.

In the meantime, there were discreet attempts to use thermal cameras to monitor sleep and breathing patterns. Sleep monitoring using thermal cameras was found to be complicated due to the afterimages that we create when we heat up the bed. On the other hand, Zhu et al. (2005) *{sensing/ad hoc, sim-1}* showed that hot air exhaled when we breath is visible in thermal images.

Then, in 2010 a paper about vital signs recognition (Poh et al., 2010) *{sensing, sim-12}* became famous when it showed that it is possible to estimate the heart rate of a person from a webcam. Although this took the computer vision community by surprise, it was a well known fact for the medical community and in 2007 (Takano and Ohta, 2007) *{sensing, real-14}* evaluated if heart rate estimations from webcams were affected by heavy makeup in a very nice paper evaluated on *sic*: *"14 sound and healthy female subjects"*. The answer is, surprisingly, that makeup does not affect the method. This example reflects the existing disconnection between different communities working on this topic.

Work on the area continued to be based in very basic algorithms and ideas, like the work by (Mansor et al., 2010a,b) *{ad hoc,sim-1}* that is heavily based on skin color recognition, which is unavailable during the night, and classifies between sleep and awake by detecting if the eyes are open or closed.

On the other hand, the work by Kittipanya-Ngam et al. (2012) is notable as they installed a camera system in a real hospital. It is the only published work that covers the ethically and privacy related problems derived of recording video in an hospital, and suggest possible solutions. Sadly their suggested computer vision algorithms were basic and most were not tested even in simulated conditions. They aimed to detect the bed position, bed occupancy, and a series of ICU-related accidents.

Finally, from 2013 onwards, practically all publications have involved the Kinect sensor or one of its variants. Kinect is a depth camera which works well during the night, is impervious to changes in light conditions, and does not depend on having specific texture patterns in the image, thus it is ideal for sleep monitoring. Furthermore, as most of the sleep monitoring algorithms measure movement, not texture, using a depth camera is highly beneficial.

Most relevant papers that use the Kinect camera include the work by Yu et al. (2013) *{sensing/ad hoc, sim-8}* that estimates sleep position and breathing rate, Lee et al. (2015) *{ad hoc,sim-20}* that estimates sleep position, and Al-Naji et al. (2017) *{sensing/ad hoc, sim-8}* that aims to recognize apnea, but ends up with a wrong extrapolation: *"we can accurately measure respiratory rate and therefore detect apnea in infants and young children"*. All those papers use such simplified scenarios that none of the techniques have changes to be used in real scenarios (*e.g.*, Al-Naji et al. (2017) requires the children to sleep without a blanket).

	occupancy & agitation	breathing analysis	sleep/awake	sleep position
< 5 patients	Reyes et al. 2010 Martinez et al. 2016a^R Martinez et al. 2013b	Harada et al. 2000 Aoki et al. 2001 Aoki et al. 2003 Zhu et al. 2005 Rofouei et al. 2011 Aoki et al. 2012	Mansor et al. 2010b Mansor et al. 2010a Lin et al. 2016	Huang et al. 2010 Rus et al. 2014
< 20 patients	Chase et al. 2004 Liao and Yang 2008	Nishida and Hori 2002 Takemura et al. 2005 Scalise et al. 2011 Martinez et al. 2012 Yu et al. 2013 Lee et al. 2014 Al-Naji et al. 2017	Chen et al. 2013 Gradl et al. 2013 Hao et al. 2013 Samy et al. 2014 Rahman et al. 2015	Hoque et al. 2010 Liu et al. 2013 Yu et al. 2013 Torres et al. 2016
≥ 20 patients	Grap et al. 2011 Martinez et al. 2013a Martinez et al. 2015	Nandakumar et al. 2015^R Martinez et al. 2016a^R Martinez et al. 2017b^R	Cole et al. 1992^R Devot et al. 2010^R Gu et al. 2014 This thesis, Chapter 7^R	Martinez et al. 2013a Lee et al. 2015 Martinez et al. 2015 Grimm et al. 2016^R

Table 2.1.: Comparison of our published contributions (violet) to the related work (green) on four of the main sleep monitoring areas. papers that evaluate on real world scenarios are marked with ^{R}. We used real world data and a large number of participants when possible.

The situation is even worse for papers that, having been published on very different fields, claim to be the first to obtain the respiratory rate using a Kinect, like (Centonze et al., 2015; Harte et al., 2016; Kumagai et al., 2016; Tahavori et al., 2015; Tanaka, 2015).

On the positive side, in (Ortmüller et al., 2015) Kinect is successfully used to track the chest movements of patients that receive radiation treatment for cancer, allowing the doctors to better aim at the tumor.

2.9. Discussion and Comparison with our Work

A major problem is that most published methods are ad hoc and are tested only on simulated environments with a reduced amount of people. This is directly coupled to the fact that most problems are weakly described, and there is no common framework that allows us to objectively compare two different methods. Due to these circumstances, many papers report excellent scores that are not representative of the performance of their methods in real world conditions. *i.e.*, all papers that estimate breathing rate provide values for the correlation above 99%.

Conversely, this thesis presents contributions in all four main areas of sleep monitoring. Our algorithms are generally developed using real world datasets and are evaluated on a large number of patients (see Table. 2.1).

3. Medical Recording Device

In this chapter we present our Medical Recording Device (MRD). Its goal is to capture the data used to train and evaluate novel algorithms for sleep monitoring in hospital and nursery homes.

Our MRD contains a depth camera that records the whole bed and a small amount of its environment. The depth camera we used is based on Kinect, which we researched extensively (see Appendix A). The main goal of the depth camera is to monitor the human body. We use a stereo camera to detect objects that are too small to be captured by the depth camera like, *e.g.*, infusion lines, glasses, newspapers, etc. We also want to evaluate if we can use the stereo camera to monitor the human body instead of the depth camera, as currently depth camera technology is proprietary and protected by several patents. The MRD has a camera specifically to observe the patient's face. This camera has its Region of Interest (ROI) altered in real time to track the patient's face, this way we are able to obtain high resolution and frame rate images. All cameras work in the NIR range, thus we added an NIR light which uses the ceiling as a reflector to avoid bright spots in the image, and its power can be adjusted to achieve a good balance between image quality and heat generation.

To track changes in the environment, we added an illumination sensor (covering both, NIR and visible wavelengths), a barometric pressure sensor, and temperature sensor (humid, and dry). Those sensors reside in an independent board with its own airflow path to avoid the heat by the MRD itself. The MRD also has a remote control receiver, and a dual color led as a status indicator. For audio input, we use stereo microphones that are integrated in the depth camera.

All those sensors and devices are connected via USB to a processing board that is based around a Samsung Exynos4412 Prime CPU (Cortex-A9 Quad Core 1.7Ghz) running Ubuntu, and receives, packs and encodes all data from the sensors and sends it through Ethernet to an external storage.

Everything is integrated in a custom designed 177×122×55 mm box that weighs less than 800g, and can be easily hanged to the ceiling using a VESA mount.

3.1. Challenges of Collecting Sleep Monitoring Data

Compared to typical data collections performed in research, sleep monitoring has several challenges:

Privacy and ethical challenges. The subject of the recording is very ethically sensitive. There are few activities as private as sleeping, thus video recording of sleep can be considered uncomfortable. This is made worse in health impaired elderly people, as some situations may arise in the bed that can affect one's dignity. We tackle this problem in depth. Our final design is small, white, noiseless, and inconspicuous, the idea is that the patient is not being continuously made aware that he is being recorded. Furthermore, we add a remote control that can stop the recording at will (*e.g.*, for bathing and cleaning patients that can not leave the bed). Finally, we enabled methods to remove recordings a posteriori. This implies that we were not allowed to access our MRDs during data capture sessions to check their status.

Safety challenges. The device needs to be installed in an intensive care room. This means that the device must not interfere with the rest of the medical devices, and must be safe to operate at all costs. Safety is not trivial, given that the box contains a large quantity of custom sensors and devices. Some devices produce a significant amount of heat (processing board, NIR light), and some parts have a significant potential for dangerous malfunctions (connectors, power management). We need to ensure that the device is safe to operate in all cases as we will have no access to the installation for the duration of the recording.

We tackle the heat management by adding a fan. This fan is designed to operate in a perfectly quiet way for most of the time (providing fresh air intake to the environmental sensors only), but in case of overheat, the fan speed will rise to remove the excess of heat (this happened only on the hotter rooms in August). We applied a full risk management pipeline: if the fan fails in a very hot day, the CPU thermal protection will be triggered and shut down the system. If both the fan fail and the CPU thermal protection fails to trigger, the case is prepared to withstand high temperatures without deforming or catching fire.

Both internal and external power supplies are medically rated and shielded. In case of a power supply malfunction there would be no spark, and the problem will not escalate outside the power supply (*i.e.*, surge and short-circuit protections).

Finally, the full box is CE compliant, certified by an external laboratory.

Autonomy. As stated previously, we had no access to the camera once installed, so we are not allowed to do any calibration of the environment like, *e.g.*, annotate the location of the bed. We also were not allowed to interact with the normal functioning of the hospital, so we could not add markers or any other information to the room. This means that the MRD must adjust all of its sensors and the devices to the environment conditions in real time and without external assistance. The

most crucial adjustments required are the fan speed, the NIR light, and some of the camera parameters.

Although most image sensors used can adjust automatically brightness and exposure time, the hard-coded parameters are not robust enough for our application. Furthermore, NIR light improves image quality but produces heat, heat needs to be cleared up by the fan but it is noisy, and also heat diminishes image quality. Our brightness-exposure-light-fan control loop is designed to maximize image quality while minimizing fan speed and required extensive testing.

Finally, a damping was added to the control loop to avoid oscillation of the fan speed, which can be noticeable by the patient during quiet nights.

The second autonomous auto-adjustment loop that will be discussed is the face detection and tracking loop which runs within the device and is required to record the patient's face in high resolution and frame rate.

Reliability. In most situations, access to the device was significantly limited. Some MRDs were installed in an elderly care home 300 km from our lab, and some MRDs installed in ICUs could only be accessed during maintenance breaks.

Something as simple as to unplug and replug a sensor in case of failure is not possible.

The MRD has two system devices (the NIR light and the fan), and ten sensors, each has its own connection methodology, firmware, and driver, and application. The uninterrupted utilization of those sensors, together with all 4 CPU cores, all communication channels, the entire memory bandwidth, and the Digital Signal Processor (DSP) coprocessor, exerts a heavy demand on the system.

Adding to this, we expect the system to be running uninterrupted for months. Few systems not specifically designed for surveillance are capable of this feat. Just as an example, Windows 95 and 98 would hang if run uninterrupted for 49 days, 17 hours, 2 minutes and 47 seconds, as the counter that monitors the clock ticks from its last reboot would overflow. This bug is now solved, but similar bugs appear continuously in multitude of drivers and hardware implementations, and are notoriously difficult to find. In fact, it took 4 years for Microsoft to find and solve the 49-day bug.

We designed the MRD to degrade its performance gracefully if needed, and recover from most problems automatically.

Most basic, the onboard system of the MRD has only 2 GiB of RAM, thus it is crucial to avoid memory leaks. The main management program is written entirely in C++11 using RAII for memory managemen. C++11 was selected instead of Java or C# for performance reasons and to avoid garbage collection, since we are very close to the maximum CPU and memory bandwidth utilization, we wanted to have a tight control of the memory buffers.

We designed a circuit board containing the stereo cameras and the sensor boards. The communication protocol used in this board is packet based and not connection based, thus if some packets are lost, images will be lost but the connection would

not be interrupted and thus the system would not stop working. Workarounds to restart the USB subsystem were placed, as most hardware USB implementations in ARM are buggy and suffer for stability issues. For the depth camera, we also needed a workaround to ensure stability (particularly, of its audio subsystem).

The compression pipeline used most of the CPU power, but in case of CPU usage peaks (or, most commonly, to avoid high temperatures), the framerate of the depth camera would be reduced and thus less compression would be needed. This would also happen in case of network throttling. Finally, a network library was developed with two main goals: avoid memory copies, and recover automatically from any problem.

As a result, our devices installed in the field worked flawlessly from day-0. Several sensor failures and recoveries were logged, corresponding to a total of 2 minutes of video lost in around 800h of recordings (0,0042%).

As for hardware failures, our devices logged 10000 hours, and we only experienced one unreliable Kinect camera, and one USB malfunction, which was promptly repaired.

3.2. Sensors

Due to the long ethical and privacy protocols required to install the MRDs in the hospital, we knew we would not be able to improve iteratively the sensing capacities. We needed to foresee all sensing modalities required for the task during the specification phase, which happened in late 2010.

The original project description called for using a single camera, but it was readily apparent that this single modality would drastically limit our capabilities. Hospital bed clothing is generally texture free, this makes it very difficult to accurately track the patient movements under occlusion. Also, there is a trade-off between resolution, frame rate, and field of view: if the camera is to observe the full bed, we can only obtain low resolution images from the face.

Therefore, a single color camera would not be able to track accurately the covered parts of the body, and would offer low resolution images of the non-occluded parts. We definitely needed something better.

We decided to use a **depth camera** to track the body movements even under occlusion. We added a **face camera** that uses a reduced field of view to track the face in high resolution and speed. A **stereo camera** is used to aim the face camera and detect small objects that are not well captured by the depth camera.

We also added an array of environmental sensors to track changes in temperature, pressure, humidity and illumination.

Finally, we have a stereo microphone array integrated within the depth camera.

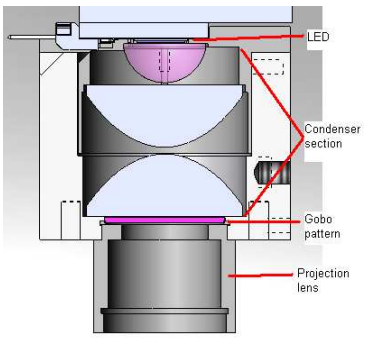


Figure 3.1.: Before Kinect was released, we implemented the (Konolige, 2010) projected lens approach pictured above, however the mechanical design involved is quite complex. Our proof of concept had a significantly worse resolution than the recently released Kinect, so the idea was abandoned.

3.2.1. Depth Camera

Back in 2010, depth cameras were too expensive and inaccurate, and Kinect was only a rumor called project Natal. To make it fit within the budget, we decided to implement the Konolige (2010) projected texture stereo camera. The idea behind this algorithm is to shine a powerful LED through a carefully designed pattern, in order to project it to the scene. This pattern adds a unique texture, and thus the depth information can be recovered precisely and efficiently using a normal stereo camera and a simple block matching algorithm.

This camera system, although inexpensive, its difficult to implement reliably (see Fig. 3.1). We designed a first prototype of this LED projector system replacing the red LED used by Konolige for an infrared led, to avoid projecting visible light during the night. We also implemented a block matching algorithm in OpenCL to be run by the integrated GPU.

We were struggling to find a good lens system and the right material to mask the pattern (few printing materials have a high enough contrast in NIR) when Kinect was released.

Kinect needs little presentation. It is a depth camera released by Microsoft as a gaming peripheral for the Microsoft XBox console, and it became widely popular due to its low price and good performance, effectively democratizing the concept of depth camera.

The technical solution provided by Kinect was, in essence, very similar to Konolige's approach with our modifications (*e.g.*, NIR lighting), but is solidly constructed and uses a Field-Programmable Gate Array (FPGA) to do the block matching algorithm online. As developing a depth sensor is not the main goal of this thesis, we swiftly adopted the new sensor.

3.2.2. Stereo Camera

We included a stereo camera in the system for four reasons.

1. To compare the precision of the depth fields obtained from our stereo camera and the Kinect.

2. To acquire visual images for labeling and perform visual NIR image processing.

3. To localize spatially objects with a distinctive texture.

4. To localize spatially objects with a small surface, *e.g.*, infusion lines.

On our first prototype we used a stereo camera based on dual MT9V034 sensors from Micron (currently OnSemi as of 2017). The MT9V034 was one of the first CMOS sensors with global shutter, and has a plethora of options that make it well suited for machine imaging applications. Its resolution of only 752×480 pixels is comparatively low for current standards, but this translates to a relatively large pixel size that works well in low illumination conditions.

Although the MT9V034 includes stereo synchronization capabilities, its interface is relatively *ad hoc*, so we choose to build a design that runs two MT9V034 in perfect clock synchronization.

For the second prototype we studied the possibility of upgrading the system to the MT9M021 sensor, which is the evolution of the MT9V034 but has a resolution of 1.2Megapixel (MP). However, the increased bandwidth of the MT9M021 would have required us to use a USB 3.0 communication chip instead of USB 2.0. Also, it would be excessively demanding for the already saturated communication and compression systems.

3.2.3. Face Camera

The face camera is designed to obtain high resolution images of the face at high framerate (see Fig. 3.2). We use the MT9M001 sensor, which is relatively large, and thus works well in low light environments. The MT9M001 has a resolution of 1280×1024 pixels, and our design goal is a framerate of 100 fps.

Sending the full 1.3MP frame at 100 fps requires a bandwidth of 130 MiB/s. This is larger than the maximum available USB 2.0 bandwidth (48MiB/s) and is in the range of the data that the ARM CPU can move around. Just trimming this amount of data without dedicated hardware would consume excessive processing power.

Therefore we implemented a digital pan and tilt system, where the camera provides a fixed region of 240×240 pixels instead of the full image. This reduces the bandwidth needs to around 5 MiB/s. However, it must be noted that the automated noise cancellation mechanisms of the sensor does not perform well when configured in such

Figure 3.2.: Images from one of the stereo cameras and the face camera. Images from the face camera are more detailed and can be captured at 100 fps due to its reduced ROI. As the faces obtained from our device are rarely frontal, we must combine tracking and detection to aim the camera.

a small ROI. This can be seen as a large amount of line noise that we corrected using a custom algorithm.

The digital pan and tilt system we implemented is relatively complex, it works by combining periodic face detection and tracking.

We periodically detect frontal faces in both left and right stereo cameras using Viola Jones algorithm (Viola and Jones, 2001). If we found a single face in each camera, and both faces lie on the same baseline, we triangulate its position. If the triangulated face is at around the distance we expect the bed to be, we accept this as a successful face detection.

Each time we successfully detect a face, we discard current tracks (if existing), and initialize new tracks on both cameras using the tracker algorithm from (Kalal et al., 2012). The output of the tracker is also triangulated to detect the face in World coordinates, and if the estimation has sufficient confidence, we send a message to the face camera to update its ROI.

The combination of tracking and detection is required, as the face detectors we can run on our limited resources are only reliable in frontal position, and this position is rare in our setup, as seen in Fig. 3.2. The continuous detection and the 3D consistence check between both cameras ensures that the tracker suffers few drift.

3.2.4. Environment Sensors

One of the goals of the system is to provide long term statistics for sleep quality. We realized that it would be interesting to find correlations between sleep quality and environmental conditions, thus we added sensors to capture the level of the ambient light, temperature, humidity and barometric pressure.

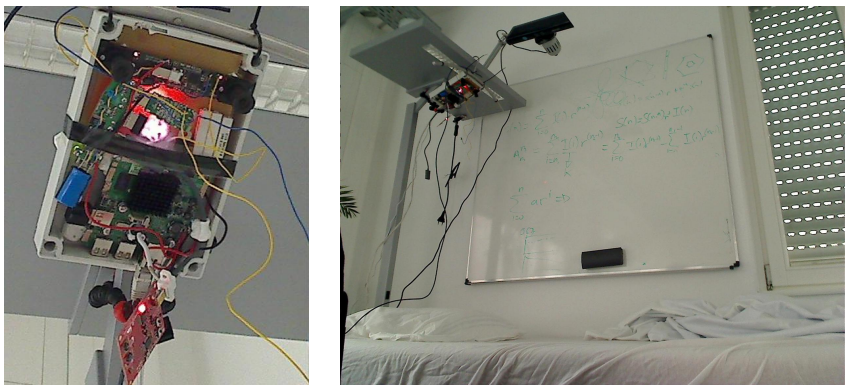


Figure 3.3.: Alpha prototype system on the left, and the full alpha setup on the right. Note the Kinect attached to the system.

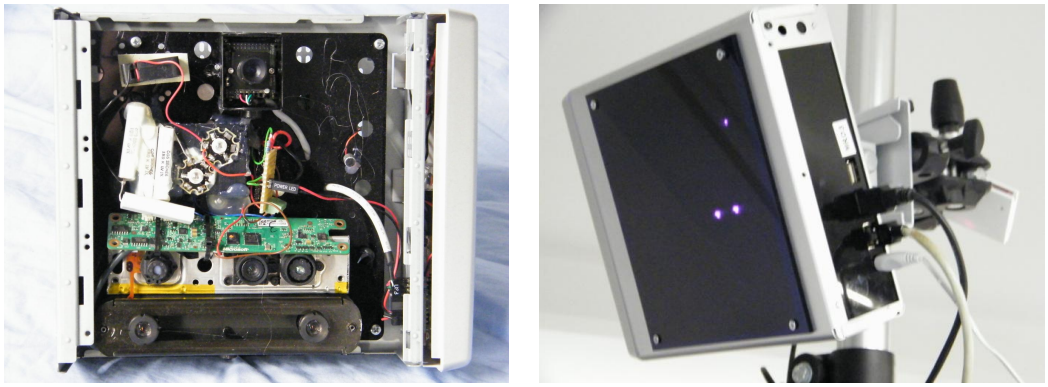


Figure 3.4.: VIPSAFE Prototype. Left: cover removed shows the Kinect board, the stereo camera, and the face camera. Right: when the cover is installed the cameras are not visible.

3.3. Prototypes

3.3.1. Alpha Prototype

Our first prototype (see Fig. 3.3) was designed to test the technologies and sensors that would be used in the future. Originally designed only to include a stereo camera and an ARM communications board, it was rapidly extended with an environmental board and a Kinect. This prototype did not include the face camera.

Notable from this prototype was the addition of the Kinect, which required us to develop an ARM driver for it from scratch, because, at the moment there was no public driver available.

3.3.2. VIPSAFE Prototype

The next iteration of the MRD was the VIPSAFE prototype, which was used in the project with the same name[1]. Five VIPSAFE prototypes were built. Four were installed in Intensive Care Units while one remained at our lab for continued testing and as spare.

The design was relatively compact, fitting a standard Mini-ITX case (see Fig. 3.4). The housing design leverages significantly on the Mini-ITX specifications. All components are tightly fit to a black laser-cut board with the size of a Mini-ITX motherboard. Both sides of the board are used, sensors lie above the board, while the processing board, power supplies and communication devices are on the other side. Although visibly unappealing, the design is robust and the tolerances are so small that the calibration values of the five prototypes we built were almost identical.

To make the Kinect fit into the design, we removed all unnecessary parts and cut its housing as seen in Fig. 3.4. The power supplies from the Kinect were taken from the CPU board instead.

As the design of the sensors was visually unappealing, we covered them using a black film that was transparent to infrared. As we could not find a single material that worked well for the purpose, we built the cover by combining orange and blue tinted plastics.

Unlike the next design, the VIPSAFE does very little processing onboard. The CPU board is based on a Marvell Armada 370 processor which is of a quite old architecture, even by 2010 standards (ARMv7). Although image processing in the ARMv7 was not practical, the processor had the unique property, at the time, of supporting Gigabit Ethernet, and it was designed to act as a bridge between USB and Ethernet devices, thus both interfaces were very reliable. Therefore, the main function of this device was to supervise the correct functioning of the sensors and transfer all the data through Ethernet to an external recording PC that did all the processing.

For this prototype we developed our communication protocol designed to be extremely reliable.

All systems underwent extensive tests for several months before being installed in the hospital. This testing was critical to find a problem that killed communication every few hours, but we could track the bug to the actual Ethernet cards we used in our computers.

A most serious problem was found after inspecting the boards that have been working for several days. The infrared illuminator that was integrated within the board was starting to melt the cover. As adding a loud fan was not an option, we decided to replace the internal illuminator for an external one.

[1] https://cvhci.anthropomatik.kit.edu/vipsafe/

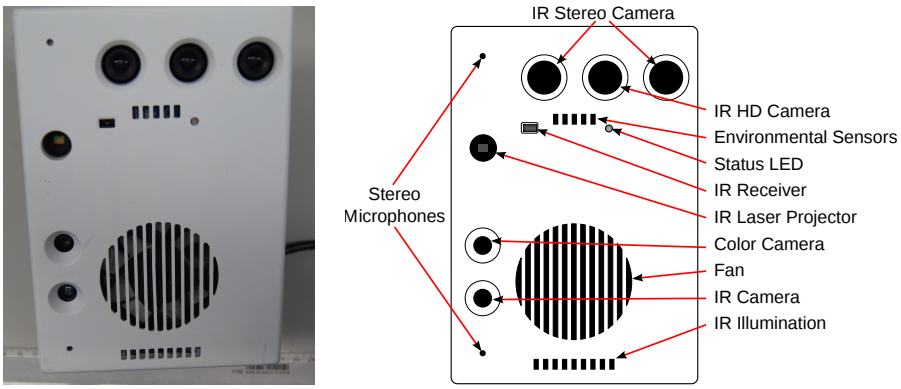


Figure 3.5.: SPHERE Prototype and its sensors. Diagram by Tobias Gehrig.

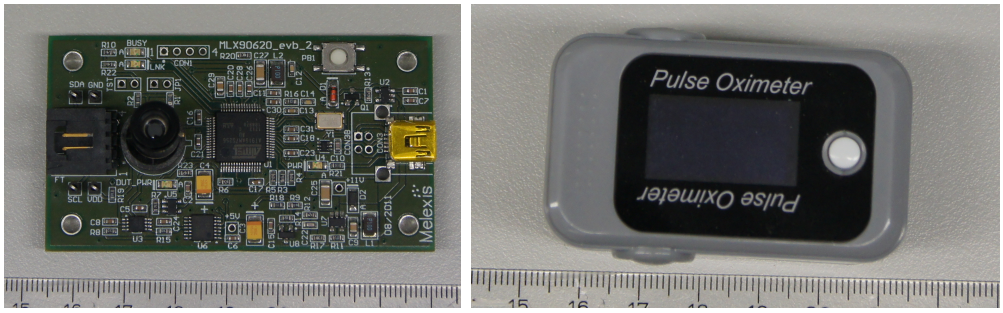


Figure 3.6.: Other sensors that we evaluated for the MRD but were not used. Left: thermal camera with 16x4 pixel resolution. Right: bluetooth pulsoximeter.

Once those two problems were solved, all systems ran without problems for months at a time.

However, we needed to reduce the resolution and framerate of the cameras in order to not avoid saturating the Gigabit Ethernet connection. Thus we decided to remove this problem on the next version by compressing the images onboard.

3.3.3. SPHERE Prototype

The current iteration of the MRD (see Fig. 3.5) is the SPHERE prototype, also named after a project with the same name[2].

For this prototype, we designed a custom case that encapsulates all sensors and the processing CPU in a very compact box. The case was designed to be almost silent (although a throttleable fan is included), but, at the same time, the airflow is designed to keep the sensors cool, increasing image quality with respect to previous designs.

[2] http://sphere-projekt.de/

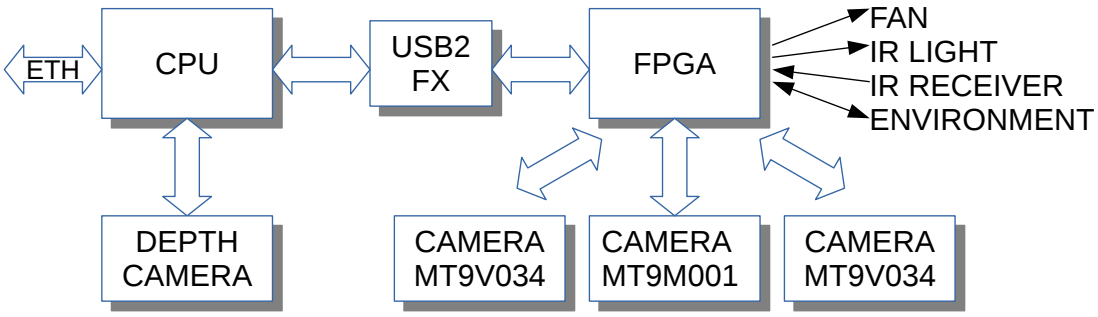


Figure 3.7.: Overview of the main active components of the MRD and its connections.

The processing board has been upgraded to a 4 core ARM9 processor with 2GiB of RAM and 64GiB of internal storage, thus the system can be used autonomously if required.

To aid in the communication and compression purposes, a FPGA was added that acts as a buffer between the cameras and the USB2.0 communication chip, allowing us to increase the image bandwidth (resolution and frames per second), without loosing frames. The FPGA also performs gamma correction to lower the precision needed from the native 10 and 12 bits per pixel of the sensors to 8 bits per pixel. This substantially improves image quality with respect to the previous iteration. The FPGA also performs minor functions, as a remote control receiver, controlling the environmental board, and throttling both the fan and the illumination board.

Also, to reduce the work done by the main CPU, the spare cycles of the USB communication processor are used to adjust the gain and exposure settings. This tight adjustment loop also increases the image quality.

Finally, the NIR illuminator that we used previously did not provide a uniform illumination. We addressed this problem by using the ceiling as a reflector: the SPHERE prototype has an illumination device that shines towards the ceiling above the bed. Therefore, the patient is flooded by indirect light, which is visibly more uniform than before.

Overall the SPHERE prototype is capable of providing images of significant better quality compared to the previous prototypes. The current bottlenecks are the ability of the CPU to compress those images, and the capacity of our servers to store them.

3.4. Hardware Design and Challenges

Here we discuss the hardware design in more detail (see Fig. 3.7), as well as the main challenges faced during the design, as well as the solutions we implemented to solve them.

3.4.1. CPU Board

There are many small embedded boards that are powerful enough for our purposes, and thus we did not require to build a CPU board from scratch. Still we did not discard this possibility, in case we found it necessary.

Our main goal for the CPU board was to be as powerful as possible while complying to a certain amount of requirements:

1. CPU needs to be reliable while being used at 100% for long periods of time. Most embedded processors are optimized for smartphones, in that scenario, the processor is idle the majority of the time and execution of tasks only happens in bursts. We found that many of those processors and embedded boards do not work reliably when run for extended periods of time. This is a show stopper for us.

2. Both USB and Ethernet interfaces also need to be reliable in the long term. In embedded processors, and specially on processors designed for the mobile phone market, both USB and Ethernet interfaces are added as an afterthought, and are often not reliable. We know that we can not expect a level of reliability comparable to a common Intel computer, but we need to check at least that we can recover from any problem using software.

3. Accelerator modules. We will be running many algorithms in parallel and most of them benefit of parallelization. Most ARM processors include a combination of OpenCL capable GPU, DSP, hardware video encoders or programmable hardware. We prioritize the boards that include such accelerator modules.

4. Support for Linux. All embedded CPUs need some modifications on the kernel to run the Linux operative system in a reliable manner. We prioritize boards that have sound Linux support.

5. Little-endianness. ARM processors can be run in both big-endian and little-endian modes, although the more modern variations are always little-endian. By choosing a processor with the same endianness as Intel processors, we will avoid the trouble and bug-prone process of making our code endianness agnostic.

6. Good power management. It should not generate too much heat, or we will need to add a loud fan to cool the device.

7. Practical factors. Budget, time constraints, availability, and possibility to be integrated.

We did a pre-evaluation of the specifications of many boards, and we purchased a few finalists to perform a more thorough evaluation and some reliability tests.

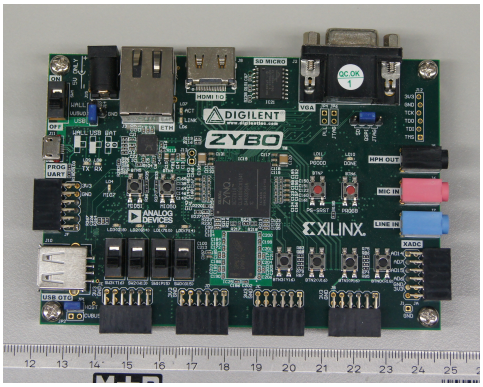


Figure 3.8.: Development board for the Zynq-7000 processor, from Xilinx. The Zynq-7000 is a hybrid chip that contains 2 ARM cores at 1GHz and a powerful FPGA. The architecture is theoretically well suited for computer vision tasks, but the performance of the ARMs is weak and is cumbersome to develop for the FPGA.

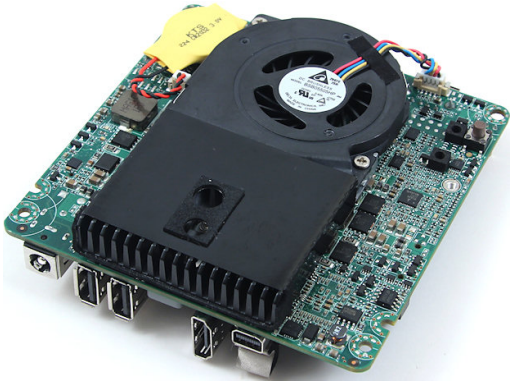


Figure 3.9.: Intel Next Unit of Computing (NUC) board. It has an soldered Intel CPU and plenty of connectors. However its power/performance ratio is lacking compared to ARM solutions.

Although we settled on a relatively traditional ARM solution, we evaluated some alternatives. On the risky side of the spectrum, we evaluated a board that uses the Zynq-7000 processor from Xilinx (see Fig. 3.8). The Zynq-7000 processor is a combination of a FPGA and a dual core ARM processor. Although the Zynq-7000 ARM cores are very slow, we expected to leverage the FPGA to do most of the algorithmic work that is very suitable for parallelization. However, at the time of the design, the Zynq-7000 was relatively new, and the software tools required to interface the FPGA and the processor were really cumbersome to use. This, together with the fact that we would need to redesign the Printed Circuit Boards (PCB) and that we did not have a fallback solution in case the FPGA perform worse than expected, convinced us that the Zynq was too risky.

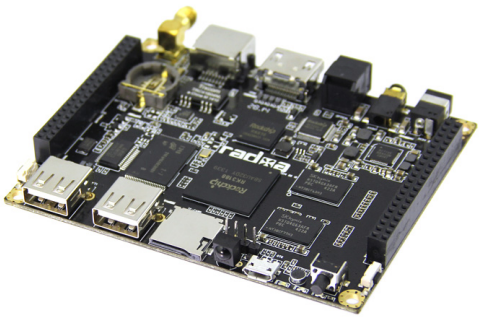


Figure 3.10.: Radxa Rock Pro board. It includes a 4-core Cortex-A9 ARM CPU from Rockchip. It is not stable under heavy loads.

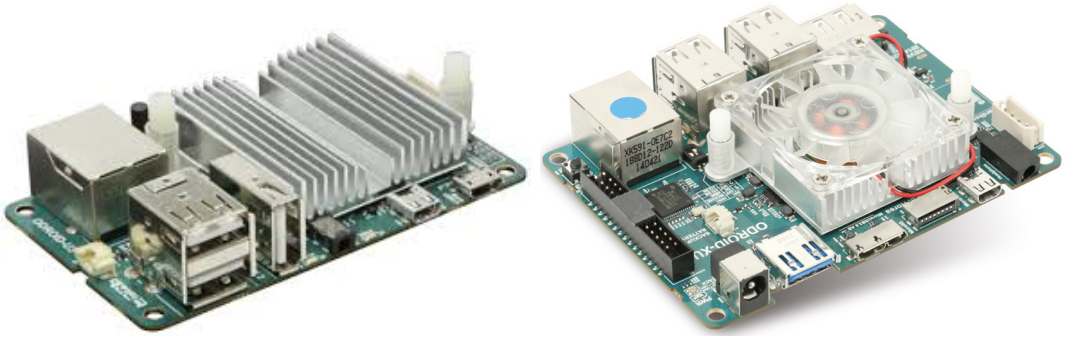


Figure 3.11.: Odroid boards. Left: Odroid-U3 with a 4th generation Samsung Exynos processor that contains 4 Cortex-A9 cores. Right: Odroid-XU, with a 5th generation Samsung Exynos processor that contains 4 Cortex-A7 and 4 Cortex A15 cores.

On the opposite corner, we evaluated an embedded Intel solution based on a NUC platfom (see Fig. 3.9). At the moment of the evaluation, we were in talks with depth camera manufacturers, and some of them only provided precompiled drivers for Intel platforms, not ARM platforms. Therefore, we wanted to evaluate an Intel board in case our software requirements forced us to chose x64 instead of ARM.

The Intel board was the most expensive of the ones we evaluated because, unlike ARM embedded boards, Intel boards require an external memory and external hard drive. Furthermore, and very surprisingly, the Intel platform was significantly less powerful than our ARM processors, and the GPU included was not well suited for parallel programming tasks. On the other hand its reliability was superb, and we used it on the lab for several years as the supervisor of the other boards.

We choose three ARM boards for evaluation: the RADXA (see Fig. 3.10), Odroid U3, and the Odroid XU (see Fig. 3.11). We asked NVIDIA about their Tegra K1 SoC which would have been ideal for our application as it contains a CUDA capable GPU, but we did not obtain an answer. When NVIDIA released the Jetson TK1

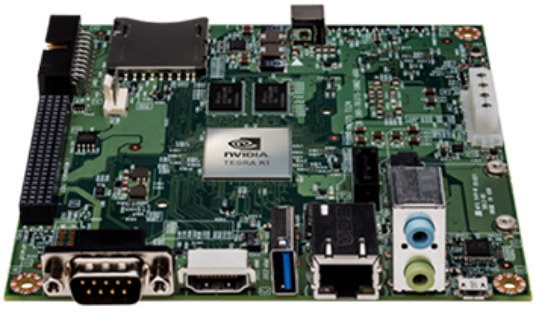


Figure 3.12.: Jetson TK1 development board from NVIDIA. Includes a Tegra CPU with 4 Cortex-A15 cores and a CUDA capable GPU. It was released too late for us to evaluate it.

developer board based on the Tegra K1 SoC (see Fig. 3.12) it was too late for us to evaluate it. The popular Raspberry Pi board was not considered as its CPU architecture is very old and not fast enough for our task.

Both the RADXA and the U3 have similar specifications, sporting a quad core Cortex-A9 processor at 1.6GHz. The RADXA processor is provided by Rockchip and has the advantage that it incorporates both WiFi and bluetooth interfaces, and the board provides many connectors where we could easily plug our custom made PCB boards. The U3 also sports a quad core Cortex-A9 at 1.6GHz, this time provided by Samsung, and has more limited expansion options, but also a more compact build and, being the 3rd iteration of the board, was very reliable. Finally the XU was the high end model of the Odroid manufacturer, and includes an asymmetric octacore design with 4 weak but power conscious Cortex-A7 CPU, and 4 powerful Cortex-A15 CPUs running at 2Ghz.

We ran several benchmarks and reliability tests on all boards. First, we found out that neither the RADXA nor the XU were as reliable as we would have liked them to be, but we solved the problems of the XU by upgrading the power supply. The RADXA remained unreliable, and thus it was promptly discarded.

The benchmarks shown in Table. 3.1 reveal some unexpected results. We executed the very old Unixbench benchmark, which is sufficient for our purposes. The RADXA was unable to complete the benchmark and thus it is not included in the table.

Interestingly, the Intel processor in the NUC was significantly slower on the synthetic CPU tasks (Dhrystone and Whetstone), but performed comparatively well on the system tasks due to its better memory interface. This result highlights the fact that the ARM architectures are mainly bottlenecked by their minute memory subsystem, and thus effective use of memory transfers must be considered when programming for ARM.

On the ARM boards, although the Cortex-A15 in the XU should be twice as fast than the old Cortex-A9 in the U3, we saw instead that both architectures achieve comparable scores on the test. We delved deep into the problem, and saw that the

Platform	Cores	Speed	Dhry.	D-P Whet.	Execl thr.	System Score
Marvell Armada	1	1.0GHz	167.7	8.0	154.4	54.9
Intel NUC DN2820FYKH	2	1.1GHz	684.2	663.5	893.5	763.0
Odroid U3	4	1.6GHz	2377.9	814.6	985.2	806.6
Odroid XU	4[1]	2.0GHz	2657.4	538.5	1166.3	771.2
Raspberry Pi Model B	1	0.7GHz	180.8	55.4	54.8	102.4
Intel PIII 800EB	1	0.8GHz	229.0	112.4	203.8	188.6
Intel Core i7-4771	4[2]	3.5GHz	14680.8	2899.4	6403.3	5978.6

[1] the CPU has 8 cores, but only 4 can be active simultaneously.
[2] the CPU has 4 real cores and 8 virtual cores due to hyperthreading.

Table 3.1.: Selected results from the Unixbench benchmark. The finalist boards to be integrated in the system were the Odroid-U3, the Odroid-XU and the Intel NUC. We can see that the U3 provides the best overall result, and to excessive power consumption of the XU caused throttling. In all cases, all those boards provide a huge performance boost with respect to the Marvell Armada processor used in the 1st version of the device. Note that the results of the PIII, i7 and Raspberry PI were added as a reference and were obtained from http://enchufado.com/proyectos/unixbench.html.

A15 achieves high performance at a significant power consumption cost. Continuous usage of the A15 throttles down its speed, thus in the long term it performs worse than the A9. We tried to solve this by testing different scheduler settings and improving the cooling solution of the XU, but the performance did not improve significantly.

On the other hand, the A9 architecture provided steady and reliable results without thermal throttling. Therefore, we selected the A9 architecture of the Odroid-U2 to be used.

3.4.2. Depth Camera

The depth camera is the most important sensor in the MRD. Selecting the right depth camera took many months, as there are not many devices to choose, and we could not find one that fulfilled all our requirements. We needed to compromise.

Performance wise, we required at least the same performance of the Kinect v1 that we used on the VIPSAFE version. Of course, we would like to improve both the spatial and the depth resolution of the camera if possible.

In particular, we wanted to use the Kinect v2 (see Fig. 3.13), that was just released during the specifications phase of SPHERE, and had impressive specifications. However, the Kinect v2 is difficult to integrate into a portable platform: it is large, generates a lot of heat, requires a USB3.0 interface, the depth calculation is performed by the CPU, and it requires a full Windows PC. Some of those requirements have

Figure 3.13.: Kinect v2 from Microsoft. It is the best depth camera currently available, however it is bulky, and requires a powerful PC just to capture the depth signal, thus it is not well suited to be integrated.

faded overtime, as it can be used from Linux now, but still it is not an optimal solution for an embedded system.

It would be normal to think that we should have used the Kinect v2, even if it was dully impractical at the moment, because future depth cameras will be cheaper and better than the Kinect v2. But this is not the case. Depth cameras were very expensive and, will be expensive again after the Kinect enthusiasm fades. Since the introduction of Kinect v1 in 2010 the market has seen only a handful of competitors that barely come close to the price and performance of the original Kinect v1.

The Kinect v2 is in its own league. No competitor has matched its performance in the last 4 years, and it does not seem that the market is large enough to invest the required money to build an alternative. In short, we are stuck with the few depth cameras we have now, and it would be unwise to expect that in the mid-term future a better camera than the Kinect v2 will appear.

Time-of-Flight (TOF)

Of the two technologies currently in use for depth cameras, projected texture like Kinect v1, and Time-of-Flight (TOF) for Kinect v2, we strongly preferred the second one. The reason is that we wanted to capture also good quality infrared images, and we did not want them to be polluted by a noise pattern.

We could find only two TOF cameras that fit our budget and had an advertised performance close to that of Kinect v1, the PMD NANO (see Fig. 3.14), and the SoftKinetic reference depth camera (see Fig. 3.15).

The PMD was quickly discarded for several reasons: it was a prototype, it was not CE compliant, its performance was not sufficient at our target distance, and the depth presented a few artifacts due to its internal heat melting some of the lubricants used in the lens. The camera did not appear to be production ready.

Figure 3.14.: The PMD NANO is a compact TOF device well suited for integration, but it is better suited for close range interaction tasks.

Figure 3.15.: The SoftKinetic development system is also a TOF camera, and it was also better suited for close range interaction tasks.

The SoftKinectic camera was then extensively tested (see Fig. 3.15). Although originally its performance envelope did not suit our application, we talked extensively for months with engineers at SoftKinetic with the aim to adapt the device to our task, but in the end we found a showstopper.

Sadly, after months of evaluation, we could not find a suitable TOF camera for our MRD.

Projected pattern

As a backup plan, we also evaluated projected pattern cameras.

Google was developing a projected pattern camera for embedded use within its Google Tango project, but the camera has not been released as an independent product.

Intel has an alternative depth camera technology called RealSense, with multiple devices available. We purchased one for evaluation (see Fig. 3.16). However it is mainly designed for close range interaction and does not perform well for our mid range task.

Figure 3.16.: Intel RealSense device. It is a projected pattern device also better suited for close range interactions.

Figure 3.17.: ASUS Xtion, based on the same chip as Kinect v1 but in a compact form factor. It is the device that was finally installed in our MRD.

The only option remaining was to use the technology from PrimeSense (*e.g.*, Kinect v1). The actual Kinect v1 is no longer in production and requires substantial modifications to be embedded in a portable system. Therefore, we used the ASUS Xtion which is based on the same PS1080 chip from PrimeSense that is used in the Kinect v1, but its more compact (see Fig. 3.17).

Then PrimeSense was acquired by Apple, and all its open source code and its future projects were instantly canceled. The future of the ASUS Xtion is uncertain, but it was the only depth camera that could fulfill our most basic requirements, so it is the one that was selected for our MRD.

3.4.3. Camera Board

The camera board integrates the stereo camera, the face camera, and the power supplies. It is connected to, and powers, the fan, the NIR light, the environmental board, the CPU board. It uses a single power supply of 12V.

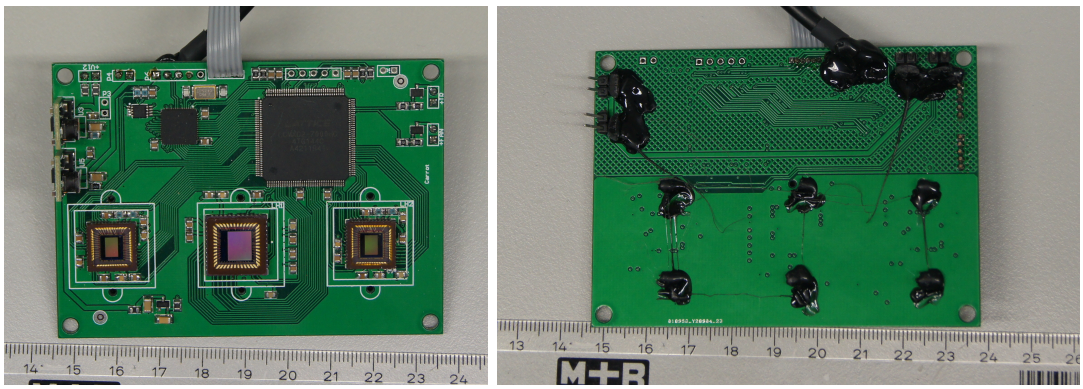


Figure 3.18.: Front and back views of the camera board. This board integrates the imaging sensors as well as performing all minor tasks required for the system: remote control receiver, PWM for the fan and the light, etc. Its main components are the FX2 USB chip (small), an FPGA (larger), and the three imaging sensors.

We designed and routed the PCB. The PCB itself was built by Eurocircuits, and the components were soldered by Lab Circuits.

In the previous approach, we used independent cameras and connected them through USB. Although we made it work, the approach had several problems, the worst of them were the congestion problems that happened when different cameras wanted to send information simultaneously could cause loss of data.

Our previous solution to this problem was to drop the resolution and framerate of the cameras and develop an *ad hoc* USB scheduling algorithm that was found to be reliable for that precise configuration.

For the SPHERE prototype we wanted a better solution, and we decided to design a board that managed the three image sensors using buffer (see Fig. 3.18). Our resulting camera board can transmit data at almost the maximum data rate that the USB2.0 specification allows (48MiB/s), while at the same time the extra buffer allows to share the USB bus with other devices without dropping frames (*i.e.*, the depth camera).

We also used this board to integrate other management functions: NIR illuminator, fan, remote control, power supplies (see Fig. 3.19). As a result the SPHERE prototype has few internal cables, and that is helpful to achieve CE certification.

Board design. The design of the board is conservative, as we did not have time or budget for trials. We selected components big enough to be soldered by hand if required, and avoided Ball Grid Array (BGA) components that require the use of very tight tolerances. We routed the board using only 4 layers and a reduced amount of bias. This was achieved by routing all signals to the FPGA pins, which can be rearranged later in software.

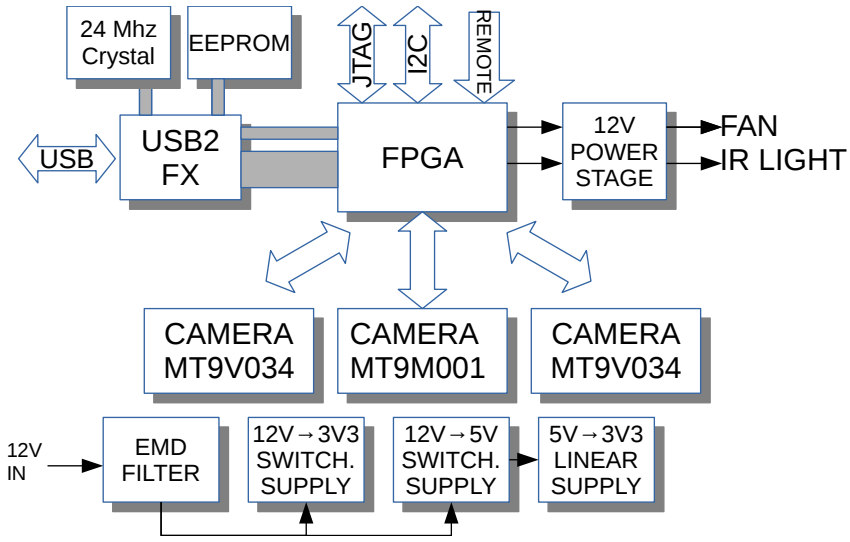


Figure 3.19.: Overview schematic of the camera board.

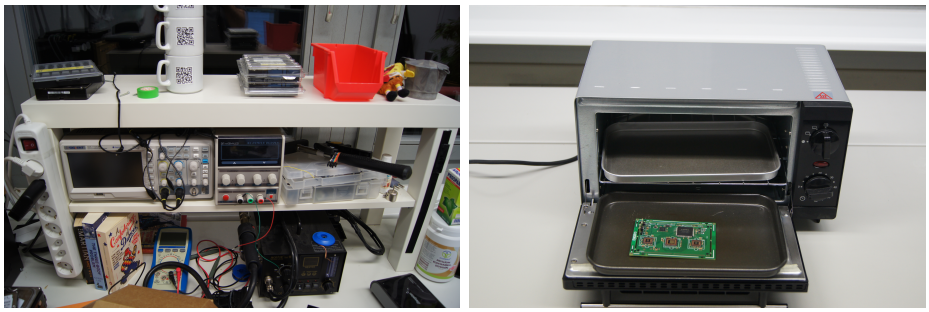


Figure 3.20.: Left: workbench we built to debug the problems with the camera board. Right: pizza oven used to resolder all delicate components.

Knowing that switching power supplies can be a significant source of electromagnetic noise, we opted to use a well known integrated design for the 5V and 3.3V supplies. Note that the CPU board is powered from the camera board through its USB interface. To increase the reliability of the board, we only used components that require a single 3.3V supply, avoiding the 2.5V, 1.8V and 1.2V supplies that we used in previous designs. However, we used an additional linear 3.3V power supply for the analog parts of the board (namely, the image sensors). In retrospective, it would have been even better to use an independent analog power supply for each of the sensors as the two MT9V034 inserted a significant amount of noise to the analog supply and this affected the image quality of the MT9M001.

Manufacturing. When we received the board, it did not work as planned. The power supplies were fine, and there were no obvious short circuits or manufacturing mistakes, but neither the FX2 chip nor the FPGA were responding. There were two possibilities: either the design of the PCB was faulty, or the assembly process

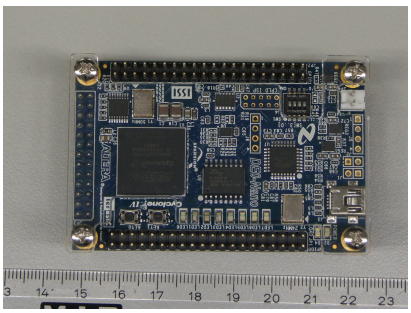


Figure 3.21.: Cyclone development board. Cyclone FPGAs are comparatively slow, but have a large amount of integrated memory. The Cyclone performed well for our task but we found a simpler alternative with the MachXO2.

was at fault. To find the culprit we desoldered and soldered again each one of the components, by hand, using a hot air pistol and a pizza oven that was used to pre-heat the board (see Fig. 3.20).

We found out that during the assembly one of the specified components was replaced by a similar but not compatible alternative, in particular, the main 24MHz crystal. Being the two crystals visually identical, it was very complicated to find the problem. Afterwards, we also found shortcircuits in the CMOS image sensors, which are common if soldered by inexperienced people.

With both problems found, the board was working as originally planned, and thus a PCB redesign was not required. We hired Lab Circuits to assemble the components of the PCB and to perform an X-ray test to discard soldering defects and shortcircuits.

FPGA Design

The previous version of the prototype was a mess of cables, so we decided to use a Field-Programmable Gate Array (FPGA) to manage all our custom devices and get rid of cables. The other three design goals for the FPGA were to add a buffer to smooth the USB communication, perform gamma correction on the image sensors, and, if possible, run a Block Matching (BM) algorithm on the stereo camera pair.

On the practical side, we wanted the FPGA design to be simple and robust.

The majority of FPGA designs require multiple external components. Those include at least 2 or 3 power supplies, an external memory chip to store its configuration, and a RAM chip for temporary transfers. The power supplies and the flash chip are rarely problematic, but the RAM required for our use case used a wide 333MHz bus that is prone to routing errors and electromagnetic emissions. We decided that using an external RAM chip was too risky, but FPGAs with sufficient internal memory are very expensive. We decided then to drop the requirement of running the BM algorithm.

Figure 3.22.: MachXO2 development board. The MachXO is a FPGA marketed as a Complex Programmable Logic Device (CPLD).

The four main FPGA manufacturers are Xilinx, Altera (now a part of Intel), Actel, and Lattice Semiconductor. Initially we wanted to use Actel devices, as their FPGAs use non-volatile memory to store their configuration, and thus do not require an external chip, but we could not find one with our memory requirements for the buffer. We evaluated the Spartan 6 from Xilinx and the Cyclone from Altera. Although the Cyclone used older technology, it had more integrated RAM, therefore we purchased one for evaluation (see Fig. 3.21). Finally, we purchased and evaluated an FPGA from Lattice that is commonly marketed as a CPLD, the MachXO2 (see Fig. 3.22). Although its performance is very low when compared to the Xilinx and Cyclone FPGAs, it has all the benefits of the CPLDs: only needs one power supply and requires no external configuration chip.

The FPGAs receives the 24Mhz clock signal from the USB chip and generates clocks and controlling signals for the three sensor devices. It performs the gamma correction, controls the throttling of the NIR illuminator and the fan, and has a small state-machine used as a remote NIR receiver.

Most importantly, we program the FPGA to act as a buffer for image data. Our buffer implementation is tightly coupled with the USB chip. We configure the USB chip to act as a triple endpoint device, one endpoint per camera, and each endpoint is double buffered. Being USB packets 512 byte long, the USB chip buffer is equivalent to 3KB of data, and it is very prone to buffer underruns. The FPGA monitors the state of the three endpoints, and offers 32 more slots that can be used by any device of the three endpoints, increasing the total buffer size by 16KiB. The scheduling inside the FPGA ensures that the most recent package received from any sensor will have a slot in the buffer queue, expelling the oldest package if required. This ensures that an stalled endpoint does not cause data loss on the remaining sensors.

Of the two main languages used to program FPGAs, Verilog and VHDL, we choose Verilog for our application as it is better suited for compact use cases. To program the FPGA, we modified the MachXO2 developmend board and used it to load the configuration to the FPGA through the JTAG interface.

USB Interface Chip

To interface the cameras with the CPU board we used a USB2.0 interface managed by the popular FX2 chip from Cypress. The FX2 is a very robust chip with a very low external component count that is very efficient in using high speed USB bandwidth (high speed defined at 480 Megabits/s, roughly equivalent to 48MiB/s). Although an FX3 version exists that enables USB 3.0 transfer speeds, it is based in a completely different design and is significantly more difficult to integrate.

The FX2 chip has a series of internal buffers that are controlled by a 8051-class processor running at 48Mhz.

The largest challenge we faced while programming the FX2 chip was to ensure a correct interface with the FPGA. The FX2 is usually configured as a master device, in this mode it takes care of pulling the data from an external parallel interface. However, it can also be configured as a purely slave device, in that case all communications are controlled using external signals.

In our case, it was too complex to program the FPGA to act as a pure master of the FX2 chip, but the master mode was not fast enough to manage the three endpoints that we required. We solved this by abusing the General Programmable Interface (GPIF), which is a custom programmable state machine that acts between the data bus and the internal buffers. We used the GPIF to generate the required signals for the FPGA communication, while retaining master control of the packages.

This freed enough cycles from the 8051 CPU that we could use them to update the gain and exposure of the image sensors.

Cypress suggests to use the Keil development system to program the FX2 chip. This is a windows only graphical environment, and it is cumbersome to integrate it in scripts. Therefore, we used a little known library named `fx2lib` to program the FX2 chip using sdcc. We then built custom programs to upload and update the firmware of the chips using the emergency reset procedure, this procedure allows us to update the firmware of the FX2 remotely.

3.4.4. Environmental Board

In the previous prototype, we used an external PCB to capture environmental data (see Fig. 3.23). This board was covered in a white case and designed to be glued to the ceiling and connected to the MRD using a white USB cable. The external board was used to isolate the temperature sensors from the heat generated by the MRD.

Although the solution was solid, the original PCB became obsolete by the time the SPHERE prototype was designed, and could not be acquired anymore. This forced us to design a board with equivalent functionality. We used this chance to design the board to be internally mounted in the MRD, see Fig. 3.24.

This board also has sensors for temperature, illumination and humidity, and we added a status led (bicolor red and green), and the NIR receiver for a remote control.

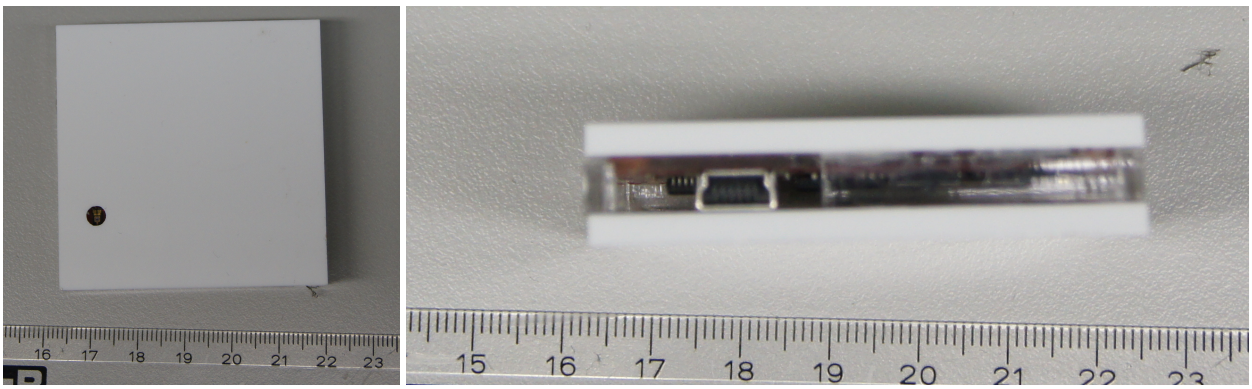


Figure 3.23.: Environmental boad used in the VIPSAFE prototype. It is mounted externally and connected to the main MRD via USB.

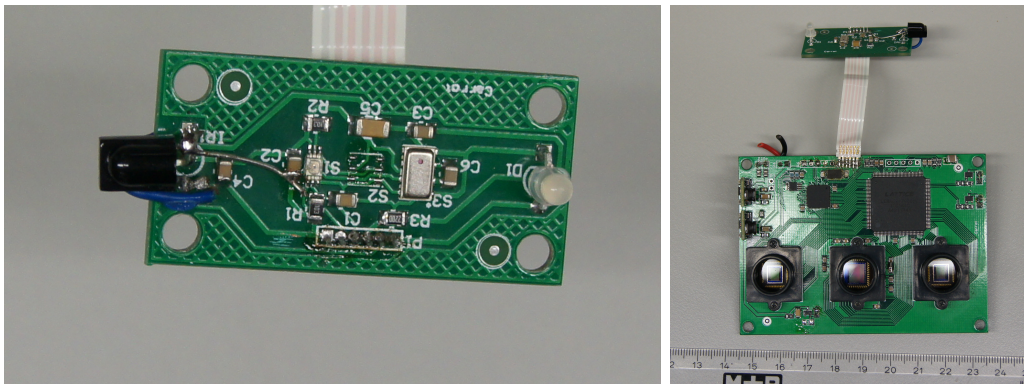


Figure 3.24.: Environmental boad used in SPHERE prototype. Has sensors for light intensity, temperature, humidity and barometric pressure. It also has a bicolor status LED and a NIR receiver for the remote control. It is connected to the FPGA on the camera board.

The environmental PCB is connected to the camera PCB by a flat 6 pin cable. The sensors are connected using a single I^2C bus and thus are controlled by the FX2 chip.

Unlike the other PCBs, which were correctly designed on the first try, we did two mistakes when designing this PCB. Namely, the pinouts of the IR receiver and the barometric sensor were reversed. However, due to the simple design of the board both mistakes could be solved during soldering.

Finally, we placed the environmental board far from heat sources and opened an air vent just above of it, therefore the sensors receive fresh air coming from outside the MRD.

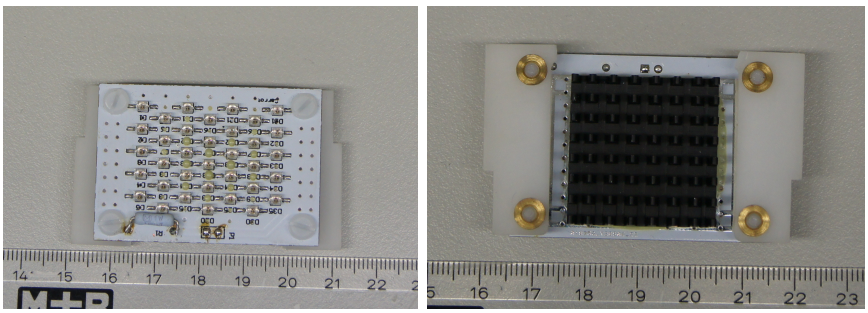


Figure 3.25.: Front and back views of the LED pcb. It is tinted in white to match with the rest of the device. As it generates a significant amount of heat, it is colled by a heatsink.

3.4.5. Illumination

NIR illumination is needed to obtain a clear image during the night. In the previous prototype we placed the NIR illuminator inside the case, but the heat generated by the led was enough to melt the plastic cover, hence we were forced to use an external NIR illuminator. This illuminator, however, did not illuminate the scene uniformly.

To solve all those problems we designed a custom PCB with 35 Vishay VSMY2853G LEDs (see Fig. 3.25). The VSMY2853G is high efficiency LED with a peak wavelength of $\lambda_p = 850nm$ and a relatively width angle of half intensity of $\phi = \pm 28°$. The VSMY2853G is designed for a maximum continuous current of 100mA, but in our PCB we use them at 90mA. This corresponds to a power usage per the entire illuminator of of 5.4W (around 50% of the entire MRD). To help in keeping the LEDs cool, we glued a heatsink to the back of the PCB, and we placed such PCB in the opposite side of the image sensors with a dedicated airflow path. This effectively solves the heat problem.

To solve the illumination uniformity problem we used the ceiling of the room as a reflector (see Fig. 3.26). As the ceiling is usually painted in white and not specular, it acts surprisingly well as a reflector and in our experience the illumination quality was significantly improved with respect to the previous prototype.

To keep temperatures in check, the illumination power is controlled by a PWM at a fixed frequency of around 10kHz, which is low enough not to generate significant Electromagnetic Compatibility (EMC) problems, but high enough to be a divisible factor of the integration time of the cameras, and thus avoid blinking effects.

3.4.6. Case Design

We designed the case using OpenSCAD with a very tight fitting for the components (see Fig. 3.27). Our idea was to use 3D printing technology to build the case (see Fig. 3.28).

Figure 3.26.: Infrared illumination window on the MRD. It is tilted 30 degrees towards the ceiling.

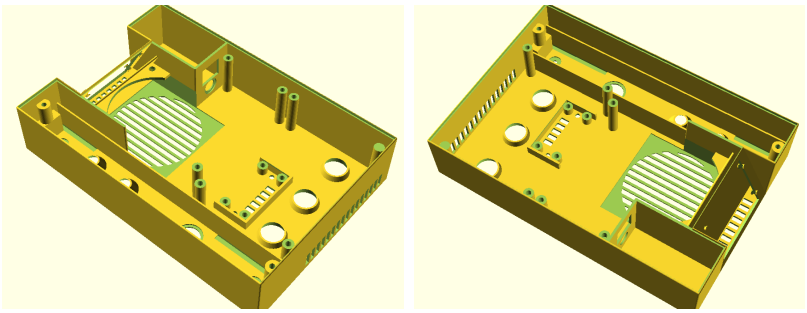


Figure 3.27.: Our SCAD design.

However, we tasked the assembly of the device to a small new company with large experience on robotics and manufacturing, and they re-created the design using Solidworks and built the cases using rapid prototyping technologies. This allowed them to use a harder plastic with better heat resistant properties and better finish while keeping it compatible with the original design (see Fig. 3.29).

The central piece of the design is the exhaust fan which expels out the heat generated in the device. The main heat generators of the device, the NIR illuminator and the CPU are very close to the fan, in this way the heat does not affect significantly the image or the environmenal sensors, each of them have a dedicated input vent, as seen in Fig. 3.30.

The design of the case is white to match with the ceiling color.

3.4.7. CE Certification

Although SPHERE was a research project, the MRD required to pass CE Certification for it to be installed in hospitals.

The CE certification process was managed by MRC Systems GmbH and the test themselves were conducted by EMV Rhein-Neckar GmbH.

The standards tested where the followng:

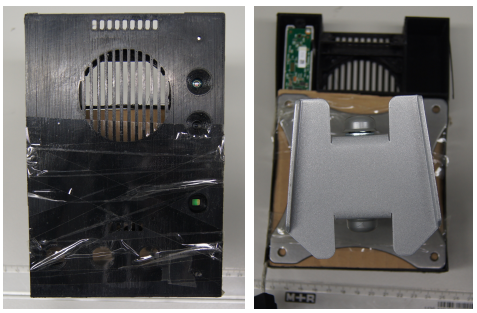


Figure 3.28.: The original 3D printed cases. They act as a frame where to mount all the components, and also have a very specific thermal flow design to keep sensors cool.

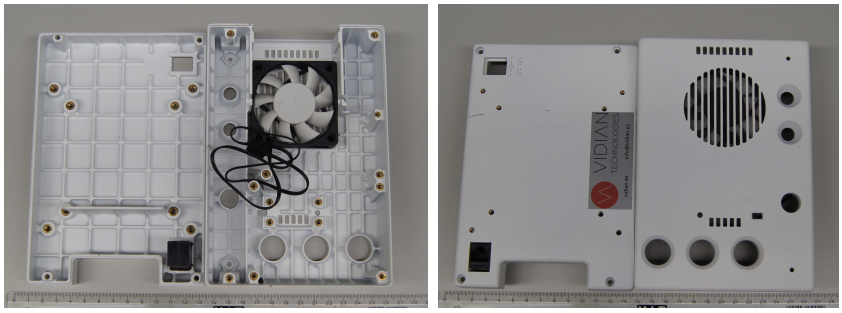


Figure 3.29.: We contacted a third party to build the case in a high quality thermal resistant plastic. The final white color matches the color of most ceilings. The cases are build using fast prototyping technologies and has metal components for the screw fixtures for increased reliability and safety.

IEC 61000-6-1: 2007 Generic standards - Immunity for residential, commercial and light industrial environments.

IEC 61000-6-2: 2005 Generic standards - Immunity for industrial environments.

IEC 61000-6-3: 2006 + A1:2010 Generic standards - Emission standard for residential, commercial and light industrial environments.

EN 55032 Electromagnetic Compatibility of Multimedia Equipment

Thus we required to perform required the follwing tests:

DIN EN 55016-1-4 Radio Interference Field Strength Emission.

DIN EN 61000-4-3 Radiated Field Immunity.

DIN EN 61000-4-6 Conducted Disturbances Immunity.

DIN EN 61000-4-4 Burst Immunity.

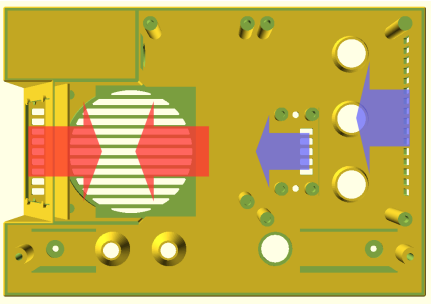


Figure 3.30.: Thermal flow design that keeps sensors cool.

Figure 3.31.: TV Receiver used to debug the electrical emissions of the MRD.

DIN EN 61000-4-11 Voltage Dips, Short Supply Voltage Interruptions.

DIN EN 61000-4-5 Surge Immunity.

DIN EN 61000-4-2 Electrostatic Discharge (ESD).

DIN EN 55016-1-4 Radio Interference Field Strength Emission / Electromagnetic Compatibility (EMC).

DIN EN 55016-1-2 Radio Interference Voltage Emission.

The MRD passed all but three of the tests on the first try. The exceptions were the voltage emission test, the ESD test, and the EMC test.

The voltage emission test measures the interferences that the system inserts back into the main electricity network through the power supply. Initially we used an inexpensive but CE compliant power supply that inserted too much interferences into the main supply. By using a better power supply we passed the test in the second try.

The ESD test measures how well the system performs after receiving an high voltage electric discharge. This test involves several subtests, but our MRD performed the worst on the air-based discharge test. The requirements of the test are to fulfill a capability level B when faced by a 8kV discharge. The capability level B implies that the system may reboot, but should resume operations independently. Instead,

Figure 3.32.: We tested the device inside this box to avoid external EMC interferences. We latter used a microwave for improved isolation.

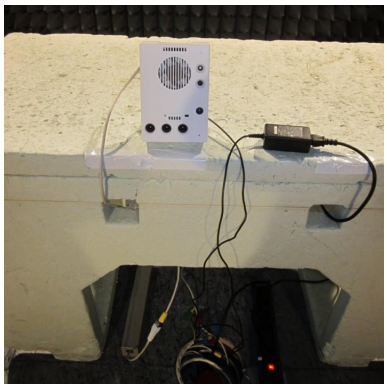


Figure 3.33.: The MRD being tested for EMC compliance.

the MRD stopped working altogether and normal operation could only be resumed after unplugging and plugging the device itself from the power supply. At the very least, the MRD was not damaged during the test.

We must note that our car keys, mobile phones, and several other nearby devices supposedly CE compliant, did stop working while we tested the MRD for electrostatic discharges.

The integrated watchdog in the CPU did not fulfill his function, and modifying it was outside our goals. Instead, we argued that the failure to recover from an electric shock does not imply any danger to the other devices in the hospital, and thus would not impact the safety rating of the device. Following this argumentation, the ESD requisite was waived.

Finally, the EMC test checks the EMC emissions from the MRD, and they were significantly off charts. This came to us as a surprise as all the parts we used were CE compliant, and thus we expected that the sum of CE compliant parts would be also CE compliant. However, this is not the case.

Finding the source of the EMC emissions was a trial and error task, which was hindered by the 200€/h price of the EMC testing facilities, and the complexity of the inner design of the MRD. Usually, EMC emission peaks can be tracked to clock

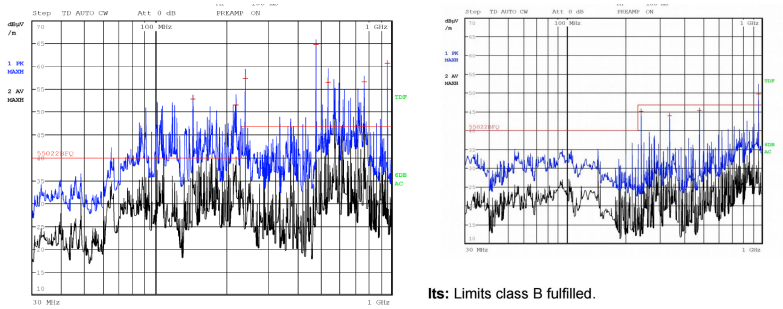


Figure 3.34.: Original and final results of the EMC test respectively.

signals, but this was problematic, as the main source of problems seemed to be related to the 480MHz frequency which was extensively used by the depth camera sensor, the camera board, and the CPU board. Furthermore, plenty of other frequencies were also above the limits, and those were more difficult to track.

The CPU board itself uses more than 8 clocks for the CPU and the memory, therefore it was the most probably source of our EMC interferences. We checked the original EMC tests performed in Korea to achieve CE certification, and those test were performed under very light loads, which could be misleading.

The camera board also used several clocks, which are generated from a 24 MHz clock connected to the FX2 chip. The FX2 chip uses a Phase-locked Loop (PLL) to generates a main 48Mhz clock and a 480MHz clock for high speed USB communication. Then the FPGA used the 48Mhz as the source of its own internal PLL and generated clocks at 24MHz, 27MHz, and 120MHz.

With so many clocks, finding the actual sources of the emissions was a daunting task. We decided to streamline the process by creating a small EMC emissions lab in our office. We used a Software Defined Radio (SDR) (see Fig. 3.31) to monitor the frequencies emitted by the MRD under different configurations.

Most SDRs can be now programmed under Linux. Their internal architecture is of a simple super-heterodine receiver where an oscillator is used to tune the receiver to any frequency between 50MHz and 2GHz, and the signal is simply acquired using a 8-bit digital-to-analog (DAC) over a 2MHz bandwidth. All postprocessing is done by the computer in software, therefore those receivers are protocol and application agnostic, and have been used to receive radio signals, television signals of all kinds, and even to receive satellite communications.

In our case, we modified the software to do a scan of all frequencies and estimate their peak power intensity. As our setup was uncalibrated, we could not provide absolute estimates, but we could compare quantitatively between different configurations. Also, we built a shield to isolate our test setup from external interferences, like radio stations (see Fig. 3.32), although we later used a microwave oven for even better isolation.

After testing more than 200 configurations, we found that there were three main sources of interferences:

- USB cables between boards.
- PLL used of the FPGA.
- Digital outputs of the image sensors.

We rebuilt all USB connections using shorter cables and added shielding to them, this effectively eliminated the 480MHz and related interferences.

We discovered that the PLL from the FPGA was the main source of our interferences. This was shocking as evaluation versions of this FPGA are sold in unshielded cases. In any case, we used 48MHz instead of 120MHz in the FPGA as the main processing frequency, and we used 24Mhz as the main frequency for all the image sensors. This required several optimizations to the FPGA code to make it more efficient, and also we needed to tweak the quality parameters of the image sensors. By using those frequencies, we could run the FPGA without PLL.

Finally, we discovered that if we tested the box in a very dark environment, the gain correction loop increases the gain of the image sensors to unreasonable levels until all images capture basically white noise. This translates to random oscillations in the 30 digital lines that connect the image sensors to the FPGA, as well as significant noise injected to the supply voltages due to the large number of transitions.

By simply testing the device in normal, non-dark operating conditions, the amount of transitions in the digital lines was reduced significantly and, correspondingly, the emissions generated were also lower.

We leveraged this knowledge to pass EMC certification on the second try, as seen in Fig. 3.34.

3.5. Software Design and Challenges

The software that runs in the MRD manages the health of the sensors, collects the data, compresses it, and sends it to a recording machine either using Ethernet or WiFi if required. The compressed stream generated by the device is optimal for storage, and no extra compression steps are required on the external storage.

The sensor reliability issue is an engineering problem that was solved without further due. Our logs show an occasional sensor malfunction, but those events were rare (*i.e.*, the logs show two failures with automated recoveries in the 800 hours recorded in the THX dataset). The only hardware reliability problem was related to a buggy CPU board where one of the three USB ports stopped working permanently, and it was solved simply by rerouting the sensor to one of the two remaining USB ports.

Figure 3.35.: We can generate virtual textured views from any required angle.

The two main challenges that the software needed to solve and required novel algorithms where:

- Achieving a reliable network connection under any circumstances.
- Compressing all incoming data using the right resources.

We will discuss those problems in the following subsections.

Finally, we use our own calibration protocol using a procedure that takes around 5 minutes per MRD. We use a single checkerboard placed in front of the device using various angles, and it is captured simultaneously by all cameras configured at maximum resolution. Even the depth camera is configured in raw mode where the original NIR image is recovered. This allows to estimate in a straightforward manner the intrinsic and extrinsic calibration parameters of all cameras. As we know the translation parameters from acNIR to depth thanks to our analysis in Appendix A, the same procedure calibrates also the depth view.

Due to the very tight tolerances used while building the MRD, which are in the order of tens of microns, the calibration parameters from all MRDs are practically identical.

Our calibration allows us to project any of the IR cameras to a 3D point cloud (in real world units) and generate as many virtual views as required, as seen in Fig. 3.35. We developed a 3D viewer to debug and analyze all information coming from the sensor.

3.5.1. Communication Protocol

The main goal of our communication protocol is to send the compressed data generated from the MRD to the recording PC. However, we also use the same format to store MRD data and for real time streaming.

Initially, we evaluated Robot Operating System (ROS) and ZeroMQ for communication, but ROS is not flexible enough for our use case, and ZeroMQ does not have intrinsically the level of robustness we need.

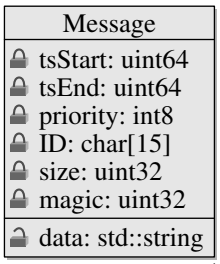
Also, ZeroMQ is optimized for low CPU usage in very high throughput scenarios (*i.e.*, sending thousands of packages per second), and the MRD does not generate that many packages. As the communication protocol, in our case, takes only a small

fraction of the CPU power, using a CPU optimized message protocol would has no observable gains.

Given the two options: modifying ZeroMQ to make it robust, or developing our own algorithm optimized for our needs, we choose the second. This also allows us to carefully avoid copying data around, as ARM processors have less available memory bandwidth than Intel ones.

The `Message`

We base our algorithmin around the `Message` structure, which follows the following format:



`tsStart` and `tsEnd`: hold timestamp information (unix time, in microseconds). If the mesage represents a sequence, `tsStart` is the timestamp of the first item and `tsEnd` is the timestamp of the last item.
`priority`: holds the priority of the package, with 0 being the default value and higher meaning higher priority.
`ID`: text field that identifies which kind of message is being sent (*i.e.*, *`"DEBUG"`*, *`"FAN"`*, *`"IR"`*, *`"LED"`*, *`"CAM0"`*, *`"CAM1"`*, *`"CAM2"`*, *`"DEPTH"`*, *`"STATUS"`*, *`"FACEROI"`*, *`"CRON"`*, *`"REMOTE"`* and *`"ENVIRONMENT"`*). The name itself does not determine the format used in the data field.
`size`: size in bytes of the data field.
`magic`: 32-bit magic fixed to: `50E8E1E8`. The magic field has two main uses. First, it identifies the blob as a message. Second, it protects against buffer underrun incidents, and for this task it is required for it to be at the end of the header.
`data`: the binary data that the message contains.

It should be noticeable the lack of error checking code (*e.g.*, CRC32). We found experimentally that the error correction codes in the TCP protocol and the Ethernet interface are sufficient to protect our data against corruption, so we did not add a third check. Furthermore, a software correction code is memory intensive and thus suboptimal for ARM architectures, by avoiding it we saved considerable processing power.

The `Queue`

The `Queue` is a fundamental part of the communication protocol and acts as a buffer of Messages. It serves several purposes:

- Its thread safe, thus it serializes the messages from several threads.
- Within the current buffer, it reorders messages chronologically.
- If the buffer grows beyond its maximum size, it drops the packages with less priority, thus ensuring a soft performance degrade.
- Can be polled for messages with a particular `ID` (*e.g.*, a particular thread that is interested only on a particular kind of message).

The `Net`

The `Net` namespace contains the `Connection`, the `Server`, and the `Client`.

The `Connection` class wraps a TCP connection around the `Message` structure and contains an input `Queue`, and an output `Queue`. We found early on that we could not trust the bandwidth throttling facilities provided by the operative system, so we implemented a simple token-based protocol that minimizes buffer underruns. The act of swapping the token between the two ends of the connection also serves as a keep-alive, which is critical for reliability. However, this also adds a delay as a Message will need to wait until the token is received to be sent, although this is limited to a maximum of 100 ms.

The `Server` object is seen by the programming model as input `Queue`, and an output `Queue`, as if it where a connection, but broadcasts the output messages to all the connected `Clients`, and aggregates the messages proceeding from all the clients into the input `Queue`. This architecture allows us to establish a second connection if the first has stopped responding but the TCP protocol has not disconnected it yet.

The `Client` simply establishes a `Connection` to the `Server`, and relaunches the `Connection` if it gets interrupted, corrupted, or lost.

3.5.2. Compression Algorithms

The main purpose of the MRD is to compress the data generated by the sensors for future use, but compressing video is not a trivial task. It generally requires a very powerful CPU or a hardware accelerated compressor, which is usually hardcoded to a specific format.

On the MRD, we use a weakly documented feature of the Exynos 4412 to accelerate h264 compression of the infrared cameras using the integrated DSP. The process is cumbersome, but it allows us to compress the data from the face camera and the stereo camera using few CPU cycles. However, it uses a significant amount of power which produces a lot of heat.

We compress the feeds from the NIR cameras using independent sequences of one second of duration. Each sequence is encoded in a single `Message`.

Audio is compressed using mp2 format. Although the mp2 codec has a smaller compression ration compared to the mp3 codec, it also respects better the original signal which is important if we want later to perform machine learning on audio data.

The depth data must be compressed in a lossless way, thus we could not use the h264 format. However, most lossless compression algorithms are incapable of compressing data at 20MiB/s on an ARM processor. We evaluated the Lagarith code (Greenwood, 2011), which is one of the fastest lossless video codecs, but the ARM implementation was not reliable. We developed our own compressor for the depth data based on range encoding.

4. Datasets

Figure 4.1.: The CV:HCI dataset was collected at the CV:HCI Lab (left). A few volunteers were also recorded at an alternate location (left), but those recordings were not used for research.

4.1. CV:HCI Dataset

This dataset was collected at the Computer Vision for Human-Computer Interaction (CV:HCI) Lab, hence its name. The CV:HCI dataset simulates an hospital room consisting of a single sized bed with bed clothing and two pillows, a night table with a cup and a pair of scissors. Simulated infusion lines were attached with duct tape to the patient's arm. Furthermore a student acting as a nurse interacts with the simulated patient at specified timestamps.

A second scenario was set up at the grounds of a project partner and was used to test different bed sizes and camera-to-bed distances, but such recordings were not used for research purposes.

A mock-up of both scenarios can be seen in Fig. 4.1.

The dataset contains one recording for each of our 23 volunteers of both genders and ages between 20 and 55 years. Two recording sessions were discarded, one for privacy reasons after we discovered that the patient's clothes were transparent to infrared light, and the second due of a technical problem that caused a synchronization loss.

Each recording session is around 20 minutes long, and each person was given the same 26 instructions which are listed in Table. 4.1. The instructions were given automatically using text-to-speech, and the timestamp of their delivery was also recorded. This allowed us to know what was the subject doing at all times.

Instruction	Action
Get into bed (eyes open)	Get Into Bed
Get into lateral right (open)	Supine To Right
Get into lateral left (open)	Right To Left
Leave the bed (eyes open)	Leave Bed
Nurse arrives and speak	Nurse Arrives
Nurse manipulates infusion lines	Nurse Manip.
Nurse leaves	Nurse Leaves
Agitate a little (supine)	Agitate Low
Agitate slightly more (supine)	Agitate Med
Agitate a lot (supine)	Agitate High
Agitate a little (fetal)	Agitate Low
Agitate slightly more (fetal)	Agitate Med
Agitate a lot (fetal)	Agitate High
Perform repetitive movements	Repetitive Mov.
Open and close fists several times	Repetitive Mov.
Shout aggressively	Shouting
Remove bed cover and cover again	Bed Cover Manip.
Manipulate the infusion lines	Inf. Lines Manip.
Touch mouth with your hand	Touch Mouth
Get the cup and drink water	Drink Water
Get a dangerous item and manipulate it	Manipulate Object
Get into bed (eyes closed)	Get Into Bed
Get into lateral right(closed)	Supine To Right
Get into lateral left (closed)	Right To Left
Leave the bed (eyes closed)	Leave Bed

Table 4.1.: Left: the instructions given to the participants. Right: corresponding action class.

The dataset uses the VIPSAFE version of the MRD (see Section 3.3.2). The VIPSAFE sensor does not perform any lossy compression, thus we acquired and saved all the data raw. Each recording is around 9GiB in size (data rate of 8 MiB/s).

Although we prefer datasets captured in real conditions, this dataset is useful to analyze actions that are rare in elderly care, (*e.g.*, shouting, aggressive behavior), dangerous (*e.g.*, manipulate the infusion lines), or difficult to label accurately (*e.g.*, agitate a little *vs.* agitate a lot).

4.2. THX Dataset

The THX dataset was collected from three hospital rooms in the sleep laboratory of the Thoraxklinik-Heidelberg GmbH, see Fig. 4.2.

It contains 99 sessions from 80 different patients, with each session corresponding to an entire night spent at the sleep laboratory. The total length of the dataset is approximately 800 hours.

Figure 4.2.: Left: one of the three rooms monitored at THX. Right: our sensor installed on the ceiling along with other sensors used at THX.

The dataset is collected using the SPHERE version of the MRD (see Section 3.3.3) and thanks to the better compression algorithms it uses only 5TiB of disk space (data rate of 1.8MiB/s).

This dataset is remarkable for three reasons:

- To the best of our knowledge, this is the largest dataset ever collected for video-based sleep monitoring.
- The subjects recorded are actual patients with a range of relevant sickness.
- The subjects were recorded simultaneously with a polysomnography and annotated by sleep medicine experts, providing us with objective reference data.

The dataset is anonymized, but contains the faces of patients, nurses, and hospital workers, and it can not be made publicly available due to ethical and data protection reasons.

For each recording we have a plethora of reference data, both curated and not curated.

We have the raw polysomnograms with more than 20 channels, including ECG, EOG, ECG, four different sensors related to respiration analysis (chest and abdomen plethystograms, a nose thermistor, and a nose barometer), one sleep position sensor, a snoring sensor, and a leg movement sensor.

We have computer estimations (provided by the polysomnography equipment) for sleep stage, and automated detection of leg movement instances, but both are considered imprecise by the doctors.

We then have the doctor corrections for sleep stage, snoring, sleep positions, and leg movement events, generally labeled at a granularity of 30 seconds (around 48000 timestamps).

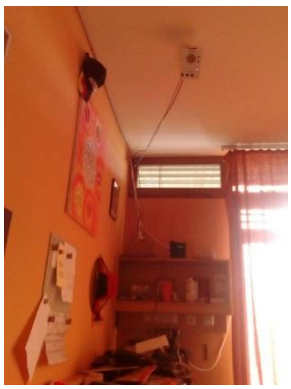


Figure 4.3.: Our MRD installed in one room from Evangelishe Heimstiftung GmbH, an organization that manages numerous nursing home.

Finally, we have a summarized report for each recorded session with more than 60 sleep quality values and several ad-hoc comments regarding diagnoses and observations.

It is very important to note that the final goal of sleep study is to a reliable diagnose sleep conditions. This means that the doctors do not need to place each of the labels with perfect accuracy if they won't change the diagnostic. In particular, most sleep quality indexes are obtained based on the amount and length of significant events, therefore many labels are not very temporally accurate. Also, the sensor setup is redundant, this way, if a sensor fails during the recording, it is not required to perform a second test (*i.e.*, 4 different signals can be used to track respiration movements). Sadly, we do not have labels corresponding to sensor failures. Still, collecting this dataset is a significant milestone of this thesis. It allowed us to develop machine learning algorithms to analyze many sleep quality indexes using real patients with relevant illnesses, and having real, objective reference data to train and evaluate our algorithms.

4.3. EHS Dataset

The EHS dataset was collected at in a nursing home managed by th Evangelishe Heimstiftung GmbH, as it can be seen in Fig. 4.3.

The dataset is also anonymized, and, like the THX dataset, can not be made publicly available.

This dataset was designed to fulfill two goals:

1. Research long term sleep patterns.

2. Evaluate our algorithms in a nursing home.

Therefore, instead of monitoring a patient for a single night, we aimed to record patients for longer periods of time, in the range of months. Up to four MRD were in use at any single time and they rotated every few months between the resident who volunteered to be monitored. The dataset contains data from 10 patients in total.

Feedback was obtained by the means of a daily form, where the participants were asked to rate the quality of their sleep in a scale from 0 to 10, being 0 the worst and 10 the best quality, and also noting any significant event that happened during the night. However, the feedback we obtained from this form was not very useful, as we are actually not very good at rating the quality of our own sleep, and many participants rated every single night with the same quality number. On the other hand, this feedback showed us that a system that can objectively rate sleep quality would be, indeed, very useful.

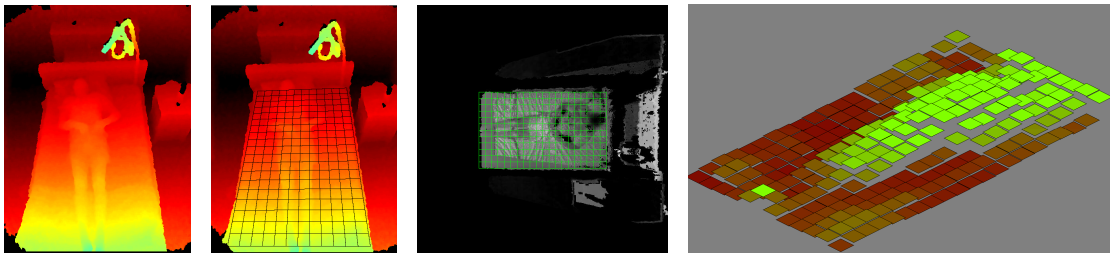
This dataset is large, very large, it contains between 5000 and 10000 hours of recordings, which is between 10 and 20 times longer than all the films released in the United States in 2016 together[1]. The dataset is so large that we have only started to grasp its full potential, and currently we have only analyzed a small subset of it to evaluate algorithms for sleep stage estimation, agitation, and bed occupancy.

The dataset was collected and curated at EHS, where they did a wonderful job to ensure that the it contains interesting individuals (*e.g.*, a wheel chair user).

It is noteworthy to mention that we obtained a very positive feedback during the installation process at EHS. Many residents volunteered with the aim of knowing if they were sleeping well or not. As we told them that, due to ethical reasons, we were not allowed to share with them the videos of their own sleep, most were disappointed, but nevertheless volunteered *for the greater good*, knowing that themselves would not benefit from it. The experience was very rewarding personally.

[1] source: en.wikipedia.org/wiki/2016_in_film

5. Bed Aligned Map



(a) (b) (c) (d)

Figure 5.1.: From a depth map (a) the bed is detected (b). A virtual top-down camera view is generated (c), and the depth points are projected into a coarse grid (d) that corresponds to the BAM descriptor.

Alignment is a major problem in computer vision systems. It relates to the fact that we usually do not know the location and orientation of the object we want to analyze with respect to the camera.

The aligment problem is usually described as a combination of translation, scale, and orientation. It is, in the general case, a 6 Degrees of Freedom (DOG) problem, and it is generally not tractable by brute-force. In those cases, algorithms based on interesting points are popular, as well as descriptors that are robust to scale and rotation changes, like SIFT (Lowe, 1999) or SURF (Bay et al., 2006).

In sleep monitoring, the alignment problem needs to be tackled in order to generate results that are consistent between different installations. In current literature, this is addressed in one of two possible ways, either the camera is rigidly fixed to the bed as in (Aoki et al., 2003, 2001; Huang et al., 2010; Lee et al., 2015; Takemura et al., 2005; Torres et al., 2016; Yu et al., 2013), or a ROI is selected manually during calibration, as in (Chase et al., 2004; Yu et al., 2013). However, neither solution is practical in many cases, as in the majority of hospitals and elderly care homes the beds have wheels and are moved frequently. And, even if the bed itself does not move, articulated beds where the user can change dynamically the shape of the bed are becoming popular. It is not practical to calibrate manually the system each time the bed is moved, or its shape is altered.

Therefore, we decided to solve the alignment problem automatically by detecting the bed and its shape. Detecting bed locations is not a new idea and it seems not to

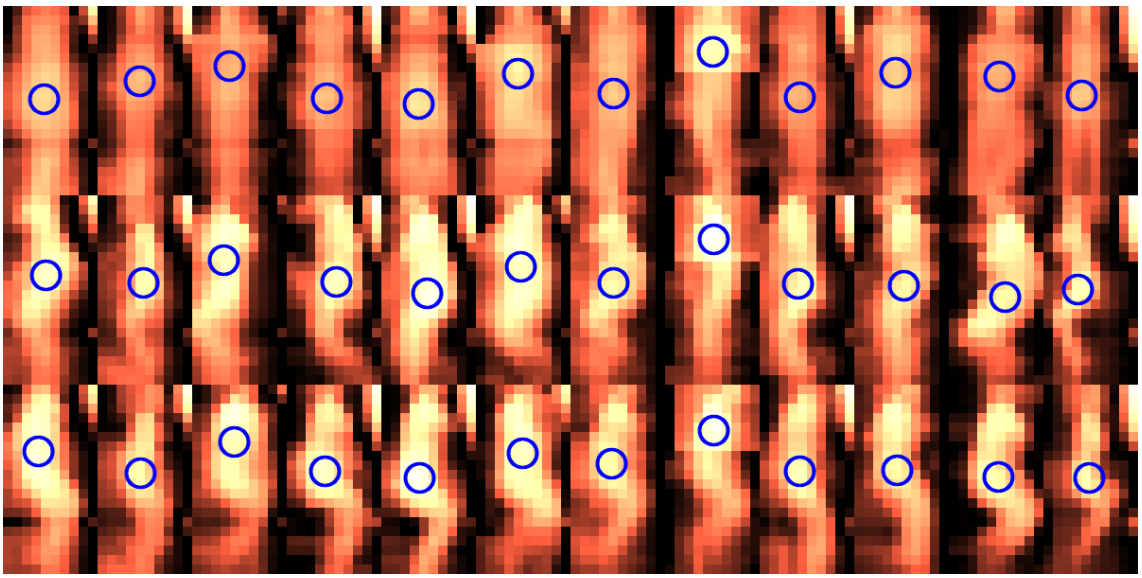


Figure 5.2.: BAM representations of subjects lying in supine position (top), on the left side (middle), and on the right side (bottom). The estimated center of gravity of the body is displayed as a circle.

be particularly challenging at first. Detecting empty beds in clean environments is simple, as seen in (Kittipanya-Ngam et al., 2012), where they solve the problem using a Hough line detector (Duda and Hart, 1972), but detecting beds in real conditions where the patient is already sleeping in it is far more complex.

We suggest an algorithm based on depth data to detect beds that we been applied successfully on the THX and the EHS datasets. Still, given the large variability of beds that exist, we expect that this algorithm will need supervision and future tuning to be employed in more diverse scenarios.

By detecting the bed in 3D world coordinates, we can project the 2.5D information we obtain from the depth camera into a virtual camera placed exactly above the bed, thus using the bed as our reference for alignment. Then, we can also project the 2D images from our NIR cameras to the virtual top-down view. This representation is translation, scale, and rotation invariant, and effectively solves the 6 DOG problem.

Furthermore, we suggest a descriptor based on this view: the Bed Aligned Map (BAM) (see Fig. 5.1). The BAM is a matrix based representation that is aligned to the bed, and in each cell contains the mean height above the mattress. Our BAM has a lower resolution compared to most common image and depth representations, as we experimented with cell sizes between 2.5x2.5 cm and 10x10 cm. The low resolution helps to store long sequences of BAMs, and also preserves the privacy of the user.

In this chapter we show how we detect the bed and how to construct the BAMs. We also evaluate BAMs in their role as a summarized representation of the a bed.

Finally, we show an approach to recognize body parts from BAMs using deep learning, that can be used, among other things, to obtain a view aligned to the user instead.

5.1. Detecting the Bed

This section explains how we detect the position of the bed with respect to the camera in real 3D world coordinates (not image coordinates). This is a 7 DOG problem where we find the translation (3 DOGs) and rotation (3 DOGs) of the bed with respect to the camera, as well as estimate the width of the bed (1 DOGs). The length of the bed is fixed at 2m, as this all the beds we ever had the chance to monitor are approximately 2m long.

The main novelty of this algorithm is that it is fully automatic, and can estimate the bed position even if the patient is currently laying on the bed.

Prerequisites. This algorithm uses the depth camera as a source, therefore needs to know its calibration parameters. The algorithm expects a 2.5D field where each pixel is labeled with its 3D position with respect to the camera.

Another requisite is that it expects the camera to be installed approximately as we instruct. This is, attached to the ceiling approximately above the feet of the patients, and with the cameras pointing to the bed. The idea is to ensure that the bed is within the field of view of the camera.

1-Tiling. Pixels that are not smooth in the disparity image are discarded. Pixels are considered smooth if the difference in disparity between them and all their 4-connected neighbors is smaller than 2. This step removes noisy pixels and pixels adjacent to edges.

Then we create tiles of 16×16 pixels. We goal is to calculate the normal of each tile for the rest of the algorithm, and also simplify the calculation of the next steps by reducing the dimensionality of the image. The 16×16 size was selected as it is small enough to offer good spatial resolution, but large enough to determine the direction of the normal vector with precision.

From now on, our representation is a matrix where each cell has translation and a normal vector.

2-Z axis and floor estimation. We estimate the distance to the floor and its normal. We use the hypothesis that the mode of the normals in the image will belong to the floor. Also we expect the camera to be installed as specified, thus normals that belong to walls can be easily discarded by thresholding.

To detect the mode we use an expectation-maximization algorithm which converges fast and is very precise. This normal becomes the Z axis of our coordinate system.

Once we found the normal of the floor, is trivial to find the actual distance to the floor: we use the point with the largest dot product with the floor norm. We use the floor as the origin of coordinates of the Z axis.

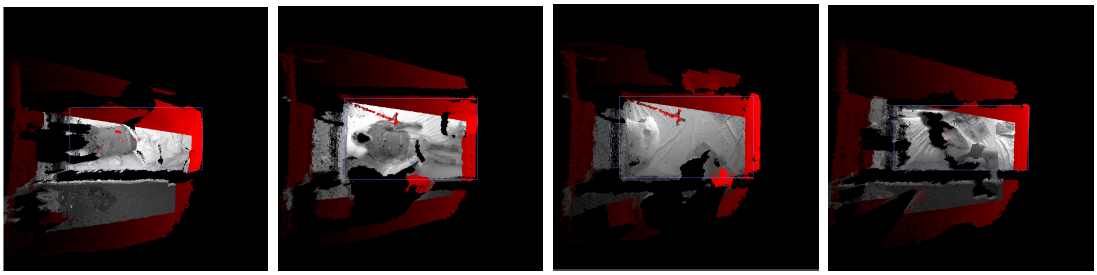


Figure 5.3.: Virtual top-down isometric views aligned to the bed in 4 different nights. Note how the infrared camera, seem to cover a different part of the image, while in reality its field of view is fixed.

This step, thus, solves 3 of the problem DOG.

From now on, we use an orthogonal projection of the ground, and it remains to find the location (x, y) of the bed, its rotation (α), its width, and, if needed, its shape.

3-Y axis, width, and rotation. This step sets the Y axis in a way that the bed will be aligned along the X axis.

We use the Hough Transform to detect the two long edges of the bed, and we also have a special check to detect beds that are next to a wall.

This steps also solves 3 of the problem DOG: the x coordinate, the rotation α and the bed's width.

3-X axis. There is only one DOG remaining, and it corresponds to the position of the bed along the x dimension. As the bed header is often occluded by pillows, we could not detect it in a reliable way. Therefore, we detect only the footer of the bed which has usually a quite simple structure, and then we use our expected length of the bed (2m) to center its position.

5.1.1. Evaluating the bed detector

To evaluate the performance of our bed detector, we estimated the bed localization 2505 times within a 7 hour long test with several volunteers acting as distractors in the bed. On 91.8% of the cases, the bed estimation was within 5cm of the ground truth, and the mean standard deviation was 13.6mm for width and 31.4mm for length.

Although this test suggested that our detector worked well on occupied beds, we must note that the results obtained are preliminary, as we only evaluated it in a few different experimental setups.

Then, we found empirically that the bed detector performed well when the MRD was deployed in the hospital and at the nursery home. In those cases, we could not acquire ground truth of the location of the bed, so we only have qualitative evaluation, as seen in Fig. 5.3.

5.2. Constructing the Bed Aligned Map

Once we have detected the bed position in the image, we can work on a orthographic top-down representation (see Fig. 5.1).

To create the BAM descriptor we first crop the view to the bed area and divide the region in cells of fixed size. In most cases we use 10×10cm cells, but we have used 5×5 cm cells on occasion, and even 2.5×2.5 cm on selected occasions. Then we project each pixel from the depth map into its corresponding cell. The BAM consists of a matrix with the average height above the bed of each cell.

We apply two ad hoc filter procedures. First, we use a very simple raytracing algorithm to remove parts that are floating above a cell, this allows us to remove unwanted artifacts (*e.g.*, the triangle shaped handle that assists the patient in waking up). We also apply a light interpolation filter to provide a depth estimate for each cell, even the occluded ones.

5.3. Evaluating the Bed Aligned Map

Qualitatively, the BAM algorithm has many desirable properties. It solves the alignment problem, and makes the task of analyzing the bed activity translation, rotation and scale independent. Furthermore, it is fully automatic to extract, and does not require any in situ calibration procedure, special clothes, or markers, and the bed is correctly detected even if its never empty. Being extracted from a depth sensor, it is robust to changes in light conditions and its performance is unaffected by the lack of texture typical of hospital settings. Finally, the BAM is an ethically conscious representation.

Quantitatively, BAMs use significantly less storage space than raw depth images. 10x10cm cell BAMs are almost exactly 500 times smaller than VGA depth images captured at 10 bits per pixel. This is important for long term sleep studies.

The expressiveness power of the BAM as a representation will be evaluated extensively in this thesis, and it has been used to analyze sleep position, sleep related actions, and to quantify agitation among others.

5.4. Finding Bodyparts in Bed Aligned Maps

We explored the alignment problem in more depth by trying to perform human-based alignment, in particular, we want to identify where are each of the body parts of the patient.

This algorithm is useful for recognizing actions, interactions, and accidents, as well as aiming the head camera, and recognizing the chest and abdomen regions for respiration analysis.

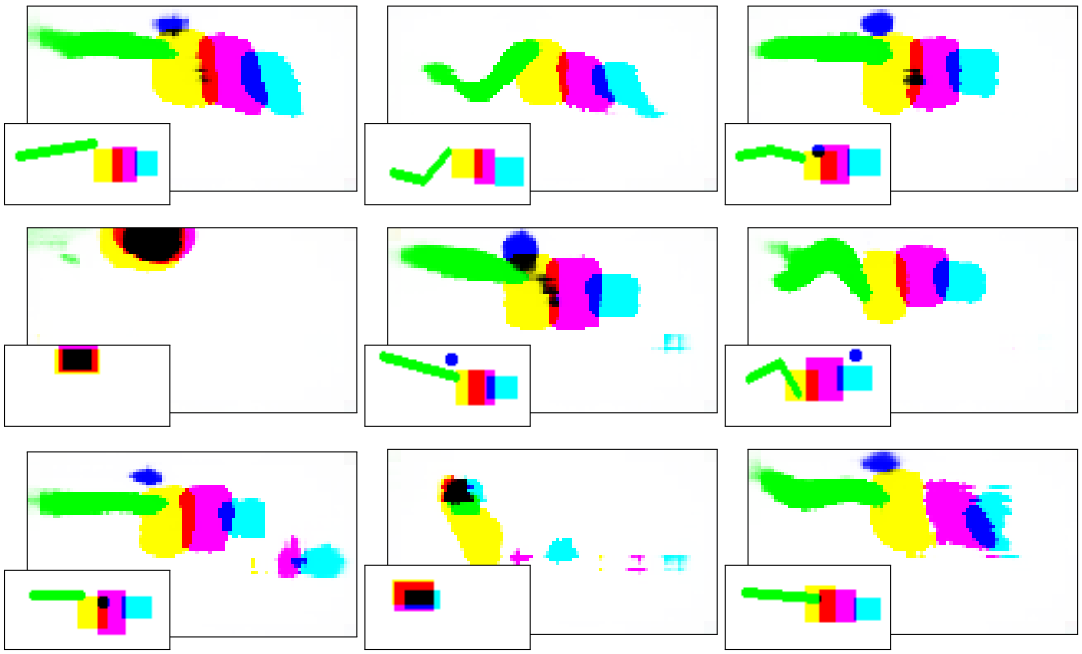


Figure 5.4.: Examples of body part detections made by our algorithm. On the bottom-left are the reference labels. Only the right leg is labeled.

Bodypart detection in depth maps has been widely explored in the path, but the seminal work in (Shotton et al., 2013) set a new standard with incredible results while using a seemingly simple machine learning algorithm based on random forests. The (Shotton et al., 2013) approach has been applied to recognize body parts in beds before (*e.g.*, Al-Naji et al. 2017), but the patient must not be covered by a blanket.

We evaluated Shotton *et al.* algorithm in our datasets, however, it performed poorly as we did not have sufficient meaningful data to train it. Therefore, we developed an approach based on Convolutional Neural Networks (CNNs) that performs better.

A preliminary analysis using the CV:HCI dataset revealed that the main problem we faced on this task is our highly unbalanced dataset, as the supine position is highly predominant. For example, if we use the mean position (the most probable body position occurring in the dataset) to predict the head's position we achieve an accuracy of 96.24% without even looking at the camera.

This is caused by the scarcity of interesting body poses in our datasets, caused, in turn, by our tendency to spend most of our sleeping time in the same reduced set of positions. Of course, one the main use of a bodypart detector is to recognize unusual positions, so the dataset bias is a strong hindrance for this application.

To counter this effect, we trained our networks using a small region of the input space instead of the whole image. The idea of this approach is to provide enough context to identify the bodypart, but not enough to find where the bed position is.

Classifier	Class	Accuracy	Recall	Precision	F-1
Forests	Head	96.28	18.73	58.31	28.11
	Chest	89.81	26.38	50.66	34.64
	Abdomen	87.52	38.37	58.62	46.21
FC (ours)	Head	97.85	72.75	75.25	73.97
	Chest	93.87	77.99	70.56	74.08
	Abdomen	91.01	80.72	65.44	72.28

Table 5.1.: Results of our bodypart classification algorithm. The forest approach is based on Shotton et al. (2013).

This approach fits perfectly well with Fully Convolutional (FC) networks, in those networks, the size of the input field and output field are not fixed, as all the layers in the network have a limited receptive field. This way, we can train the network using very small patches, and use the same unmodified network to predict the body positions of the whole bed (see Fig. 5.5).

To evaluate this algorithm, we used the THX dataset, extracted 1998 BAMs with a cell resolution of 2.5×2.5cm, and labeled the head, the chest, and the abdomen, as well as the right leg. We used a receptive field of 32×36 cells, which corresponds to 80×90cm. The problem was evaluated as a pixel- and class-wise classification problem, and we used five-fold cross-validation.

Results can be seen in Table. 5.1. For this task, our FC network outperforms significantly the approach by Shotton *et al.*. Examples of the labels and our predictions including non-mean shape scenarios can be seen in Fig. 5.4.

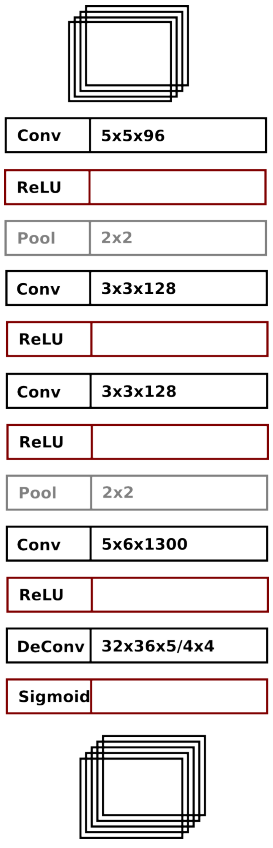
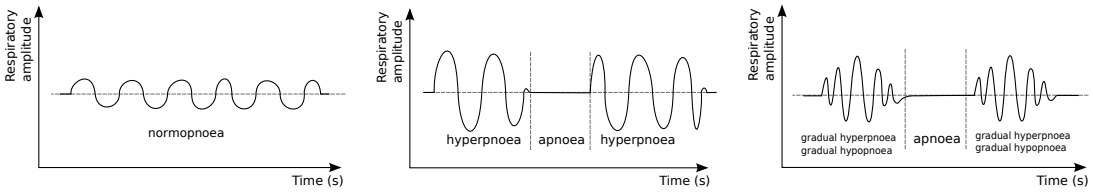


Figure 5.5.: FC Net.

6. Analysis of Respiration Patterns



(a) normal respiration (b) central apnea (c) cheyne-stokes

Source: wikimedia commons, Savvas Radević.

Figure 6.1.: Different respiration patterns.

Against the general impression, respiration is a complex signal to analyze in real conditions. The control system that manages respiration is semi-autonomic: the muscles involved can be voluntarily controlled, but the autonomous system will take care as soon the voluntary control stops. This is important as our respiration patterns change when we speak, move our body or become agitated. There are multitude of events that alter our breathing, some are very common (*e.g.*, snoring, coughing), and some are less common but important nevertheless, like obstructive apnea. In obstructive apnea the upper airways are blocked and the diaphragm moves the air from the lungs to the stomach and back, resulting in chest motion but no air exchange.

To measure respiration rate in hospitals, the nurse starts by asking the patient to remain still and relaxed, and then counts the number of chest excursions during a set period of time. If the patient coughs, talks, or becomes agitated, the nurse just repeats the test again. This procedure seems simple but it highlights the fact that we need the collaboration of the patient during the measurement.

To monitor breath rate continuously, we need to consider that the patient will eventually perform actions that disturb the breathing rate. Therefore, breath rate monitoring is more complex than breath rate measuring.

Even more complex is the analysis of respiration patterns and, in particular, sleep apnea events. It is not sufficient to detect an interruption of a normal breathing pattern, as such interruptions are common (*e.g.*, the patient going to the toilet, drinking water, coughing, talking on the phone, singing, etc.). Also, many respiration patterns do not involve an interruption of breathing pattern. Therefore, we need machine learning approaches that recognize, segment, and classify respiration patterns.

The task is, however, far from trivial. Let's consider the case of hypopnea. A hypopnea event is defined by a very shallow breathing, or a very low respiratory rate. During hypopnea, the chest movements and airflow is present, but not sufficient to maintain the oxygen levels in the blood. Even the criteria used to classify an hypopnea event in sleep medicine is fuzzy and subject to change often. In our THX dataset, hypopnea events are annotated if they show a 30% reduction in airflow lasting for 10 or more seconds.

But hypopneas are not the only complex case of apnea (see Fig. 6.1). In obstructive apneas we see quite a violent chest response as the diaphragm struggles to free the air pathway. In Cheyne-Stokes respiration the intensity of the respiratory efforts oscillates between normal to almost zero and back. In central apneas, the respiration just stops for a period of time. Although central apneas are simple to detect, they are also rare and have a poor prognosis, as they are caused by a malfunction in the respiratory center in the brain.

In this chapter, we suggest two different methods to estimate breathing rate from chest movements, one tracking the trajectories of dots that are projected into the scene, and other using a depth camera. Then, we show preliminary results on using deep learning to classify apnea events on the THX dataset. Finally, we discuss the state of the art and possible future directions.

6.1. Respiration Rate from Self-Consistent Trajectories

In this section we analyze the mathematical background of monitoring chest movements using a stereo geometry setup. This covers both stereo camera pairs, as well as depth sensors that use projected patterns like Kinect v1.

The main limitation of compter vision systems is that the small movement evoked by respiration (around 10mm as for Segars et al. 2007) is too small to measure respiration accurately. To overcome this limitation, monocular systems require the subject to wear textured clothing, like (Tan et al., 2010), and stereo systems use large baselines (Aoki et al., 2001) or just very close detection distances (Burba et al., 2012). The motivation behind our study is to establish the relationship between the camera size and the measuring distance in this particular problem.

Instead of using the MRD, which can only provide depth information and not the raw NIR image, we modified a Kinect v1 to project large NIR dots. We track each dot over 30 seconds to create a trajectory, and we combine all trajectories together using PCAs. Finally, we perform a autoregressive (AR) spectral analysis to estimate the Respiration Rate (RR). Our method is designed to maximize the information extracted from the source images while still being a realtime approach. This allows us to estimate very small movements such as the ones induced by a sleeping person

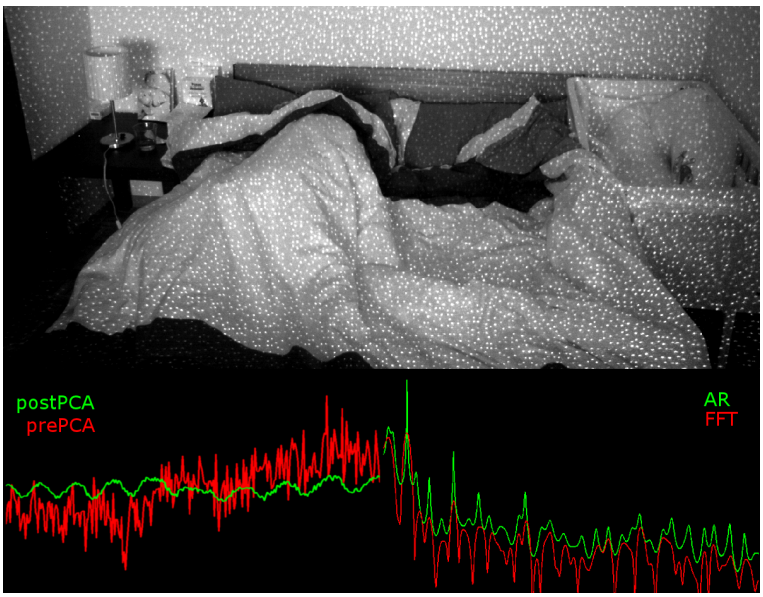


Figure 6.2.: RR estimation on a 30s window, PCA filtered signal & AR power spectrum.

from a distance of 3 meters (see Fig. 6.2), where the mean amplitude of the movement is less than 0.048 pixels, or 0.0022 degrees.

We evaluated our breath rate estimator on 9 healthy subjects and compared our results with those of an inductance plethysmograph (*i.e.*, measuring belt). Furthermore, we use an artificial system that simulates chest movements, to evaluate the distance and frequency limits of our system and compared them to those of a state-of-the-art alternative (Burba et al., 2012).

6.1.1. Image Sensor and Camera Model

Our system requires a NIR dot pattern projector and a camera with a matching NIR filter. For convenience we use a modified Kinect v1 as it includes both devices, however we use a custom driver to gain direct access to the CMOS NIR image sensor.

The main goal of modifying the Kinect v1 camera was to improve the individual detection of the dots projected by the NIR laser. We configured the NIR camera to 1280x1024 pixels and 9.1 frames per second, and added a Niko Zoom lens to the NIR projector to increase the size of the dots. Using this setup we project an average of 15000 dots within the camera Field of View (FOV) (see Fig. 6.2).

Several parameters must be taken into account in order to design the system. From the camera's point of view, when an object in the scene moves, the dots that are projected on it are displaced along their epipolar line. Using the pinhole camera model and the standard model for parallel cameras, the epipolar lines are $y = const$,

and therefore the dots move only along the horizontal dimension. This means that the trajectories we estimate take the form of a vector.

The displacement magnitude depends on the focal length of the camera f; distance between camera and projector, or baseline b; magnitude of the object displacement d; distance to the object z; and angle of the displacement respect the angle of the camera α. Then Δx is the observed horizontal displacement of a dot in the image plane in pixels:

$$\Delta x = fb(\frac{1}{z} - \frac{1}{z'}) \qquad z' = z + d\ \cos(\alpha)\ , \tag{6.1}$$

$$\Delta x = \frac{f\ b}{z^2}\ \frac{d\ \cos(\alpha)}{1 + d/z\ \cos(\alpha)}\ , \tag{6.2}$$

then, if we assume $z \gg d$ we get:

$$\Delta x = \frac{f\ b}{z^2}\ d\ \cos(\alpha)\ . \tag{6.3}$$

We approximate f by using the rectilinear lens model from the horizontal resolution r and the angular field of view β:

$$f = \frac{r}{2\ \tan(\beta/2)}\ , \tag{6.4}$$

applying it to Eqn. 6.2 we get:

$$\Delta x = \frac{r\ b}{2\ z^2\ \tan(\beta/2)}\ d\ \cos(\alpha)\ . \tag{6.5}$$

Consequently, the pixel displacement can be improved by using a better resolution, larger baseline (Aoki et al., 2001), smaller field of view or smaller distance to the object (Burba et al., 2012).

The Kinect camera has $\beta = 57.8°$ and $b = 75mm$. With the Kinect tilted 30° from the horizontal and pointing towards a person sleeping in supine position, a 10mm breathing amplitude translates to 1.74 pixels at .5 meters, 0.435 pixels at 1 meter, 0.109 pixels at 2 meters, and 0.027 pixels at 4 meters.

6.1.2. Estimating Dot Trajectories

We track each dot over 30 seconds. First we highlight the dots on the current image by convoluting it with a 2D LoG kernel of matching size, in our case 9x9 pixels. We then find the local maximum for each dot with sub-pixel precision by computing the Center of Gravity (Bailey, 2005) over a 5x5 pixel window. The tracking is performed frame by frame using a 3x3 pixel search window.

6.1.3. Feature Fusion using Principal Component Analysis

In approaches where one trajectory per pixel is obtained, it is common to fuse them simply by averaging them, as seen in (Aoki et al., 2001; Burba et al., 2012; Takano and Ohta, 2007; Tan et al., 2010). In our case, we use the dot trajectories as features, and we only take into account dots with a complete 30s record. This reduces the typical amount of features from millions to thousands, and allows us to use a more elaborate method to fuse the dots.

We experimentally found that the two major noise sources in our system are thermal noise, and mechanical vibration. The first noise shows no correlation between dots, but the second displays a strong correlation between them. However, both types of noise are uncorrelated with respect to the RR signal. In this scenario we use PCA to separate the signal component from the mechanical vibration, while reducing the level of thermal noise at the same time.

The principal components are calculated from a matrix containing the trajectories of each dot. We only need to calculate the first few components as we expect the RR signal and the correlated noise to be represented in the bases with most variance, and the thermal noise to be distributed between the remaining ones. We use the progressive Expectation-Maximization (EM) algorithm described in (Roweis, 1998) to efficiently calculate the 16th strongest components.

Then we discard noisy bases by using the Durbin-Watson (DW) test, as has been suggested in (Ryu et al., 2011):

$$d_{\text{DW}} = \frac{\sum_{t=1}^{T}(v_t - v_{t-1})^2}{\sum_{t=0}^{T} v_t^2} \ , \tag{6.6}$$

d_{DW} lies between 0 and 4, and small values indicate a positive autocorrelation. All bases with d_{DW} above average are discarded. Then we estimate the Power Spectral Density (PSD) of the remaining bases using Fast Fourier Transform (FFT), and discard the bases whose average power in the interest region (from 3 to 60 breaths per minute) is less than the average power outside the interest region.

We average the bases that remain, if any, to calculate an average trajectory with less noise, see Fig. 6.2.

6.1.4. Estimating Respiration Rate from Autoregressive Spectral Analysis

Model based methods such as autoregressive (AR) Spectral Analysis are often used in heart and breath rate monitoring (*e.g.*, Takano and Ohta 2007). An AR model can be seen as an Infinte Impulse Response filter that outputs an estimation of the data when excited by white noise. A pth order AR model is defined by:

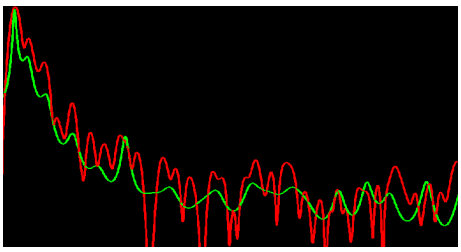


Figure 6.3.: Qualitative comparison of the PSD of a signal when estimated using AR (green) and FFT (red). Both estimations detect the maximum peak at the same frequency, but AR peaks are narrower.

$$X_t = \sum_{k=1}^{p} \varphi_k X_{k-i} + \varepsilon_t \ . \tag{6.7}$$

Once we estimate φ_k using Burg's maximum entropy method (Burg, 1968), we get the PSD of the process with:

$$S(f) = \frac{\sigma_Z^2}{|1 - \sum_{k=1}^{p} \varphi_k e^{-2\pi i k f}|^2} \ . \tag{6.8}$$

Where σ_Z^2 is the noise variance. The PSD obtained from the AR model presents narrower peaks than the PSD obtained from FFT (see Fig. 6.3). p is usually chosen empirically for each task, as higher order AR models contain more information and offer more precision, but are prone to split the main peak in two or more smaller peaks. We chose $p = 80$ based in our preliminary experiments.

Our RR estimation corresponds to the largest maxima found in our ROI of the PSD. The frequency of the detected maxima is then refined using the bisection method.

6.1.5. Evaluation

As a proof of functionality, we measured the RR of 9 healthy subjects. They were asked to rest in our test bed, and then we monitored their breathing for two minutes from a distance of 1.5 meters to the chest and an angle of 45°.

We compared the results with those of an inductance plethysmograph (*i.e.*, a chest-stretch measuring belt). We obtained a correlation value of 0.995, and the null hypothesis of no correlation is rejected with a *p-value* of 2.91^{-8}.

Using an artificial chest, we evaluated the range of operation of our system and compared it to that of a state-of-the-art alternative (Burba et al., 2012).

The artificial chest presents a surface of 30x20cm that simulates normal respiration induced movements with inhalation, exhalation and pause periods and a movement

ours	2m	3m	4m	5m
4 bpm	70	30	40	40
5.7 bpm	100	90	90	35
8.1 bpm	90	90	100	0
11.6 bpm	95	100	55	5
16.6 bpm	100	80	50	0
23.8 bpm	95	60	55	0
34.2 bpm	100	80	25	0
49.2 bpm	95	35	30	5

baseline	0.7m	0.85m	1m	1.15m
4 bpm	0	0	0	0
5.7 bpm	0	50	10	0
8.1 bpm	25	85	45	5
11.6 bpm	65	60	05	0
16.6 bpm	45	25	10	0
23.8 bpm	20	15	10	0
34.2 bpm	5	15	0	0
49.2 bpm	0	10	15	0

Table 6.1.: RR estimation success rate (in %) using an artificial chest beating different frequencies and placed at different distances from the camera.

amplitude of 10mm. In all tests, the artificial test was covered by a white textureless bed sheet.

The sensitivity of the algorithm determines the range of distances and frequencies where the RR is usable. Our test measures the capacity of the tested algorithms to discern the correct breath frequency over its harmonics and the background noise. We tested several RR and distance combinations, 20 times per bin. A test is deemed successful when the algorithm provides an estimation within ±10% of the generated RR, and a failure otherwise. To evaluate our algorithm, we placed the sensor 1m above the bed and we aimed it to the artificial chest, however, our baseline algorithm was unable to recover any signal at 1m distance, so we lowered the sensor to 50cm above the bed.

Results can be seen in Table. 6.1, where our algorithm is capable to outperform significantly (Burba et al., 2012) in both distance and frequency ranges.

6.2. Respiration Rate from Depth Maps using Early Fourier Fusion

On the previous section, we estimated RR from imagery using a custom Kinect setup that impedes its use as a depth camera. This is a significant hindrance if we need the depth information for other tasks (*e.g.*, action recognition), therefore we suggest now a method to estimate RR directly from the depth maps provided by Kinect.

Although the sensing technology used in both sections is similar, the main extra challenge that we face now is that Kinect uses a very coarse quantization step (around 25mm). This quantization is obviously problematic as we expect to capture a signal that is nominally 10mm wide. Our hypothesis here is that, by fusing the information of a large enough region of the depth field, we can recover a signal that is smaller than the quantization step (as seen in Vanderkooy and Lipshitz 1984).

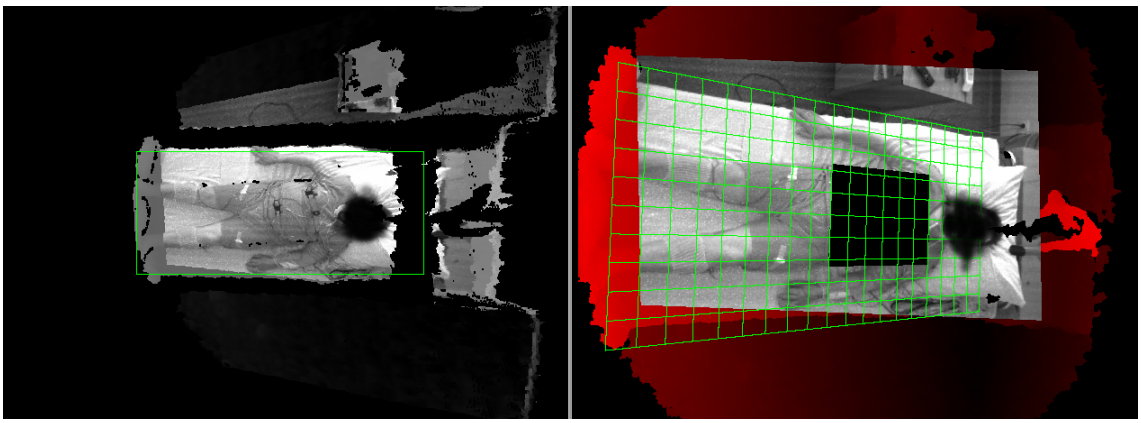


Figure 6.4.: Left: artificial top-down view generated from a BAM. The polysomnogram sensors are visibly attached to the patient, however most patients slept under a blanket. Right: raw depth map with the infrared camera superimposed. The black square is our bed aligned ROI, which is the same for all patients. Face obscured to preserve privacy.

Furthermore, we evaluate our algorithm on 3,239 segments collected from 67 sleep laboratory patients from the THX dataset. In this section introduce three novel contributions: a dynamic ROI which is aligned to the bed, a confidence metric based on patient DW, and the Early Fourier Fusion strategy. Overall, our camera based method is accurate on 85.9% of the segments. Our accuracy is similar to the one obtained from a chest sensor (88.7%). Most importantly, to the best of our knowledge, we are the first to report the performance impact of different sleep conditions, like apnea, position and staging, on RR estimation.

We took 40 samples each night (a total of 3,760) containing many challenging situations: empty beds, patients sitting, changing sleep positions, having apneas, etc. We discarded samples if at least one polysomnogram sensor was disconnected, reducing the total number to 3,239. We use a window length of 30 seconds in order to obtain results that are comparable to the ones captured manually by doctors and nurses.

We did not place any limitations on the patients actions or routines. Therefore, patients used at will: blankets of various thicknesses, a variable amount and size of pillows, and several read books, newspapers and magazines during the recording. Those variables had no discernible impact on the results.

There is no uniquely defined ground truth for the number of breaths of a sequence, precisely because the definition of what constitutes a single breath is fuzzy in several corner cases (*e.g.*, interrupted breaths, minimal expansion, coughing, etc.). Based on the recommendations of the medical doctors from the sleep laboratory, we use the estimate obtained from the thermistor placed under the nose as a reference.

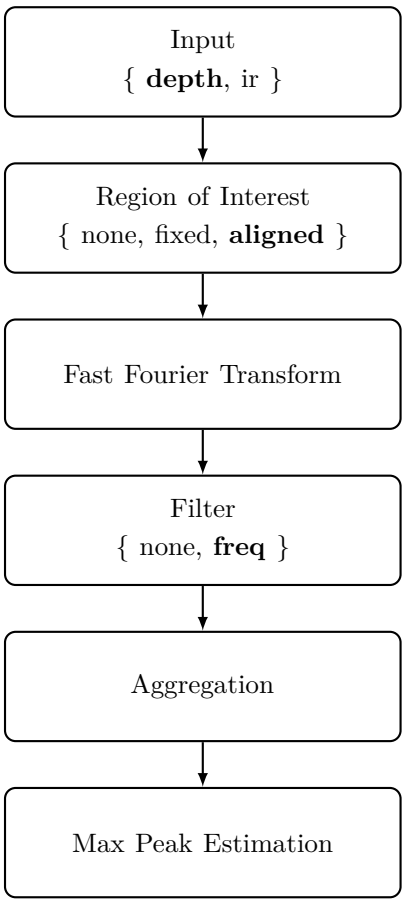


Figure 6.5.: Workflow, default settings are in **bold**.

6.2.1. Early Fourier Fusion

Our algorithm (Fig. 6.5) takes as input a sequence of images, obtains a ROI, and for each pixel within the ROI creates a trajectory in the time domain, which contains all the pixel values over the sequence.

The PSD of every trajectory is calculated using FFT, and a filter is optionally applied to each PSD.

The PSDs from all trajectories are aggregated together, and the location of its peak is found using quadratic interpolation.

Note that the information about the frequency is on the location of the peak, not its magnitude, hence the units and magnitudes actually used for the trajectories are irrelevant.

We evaluate both depth and infrared cameras as input sources. In both cases the images were compressed using a lossless algorithm, as we found out that even the slightest image degradation had a strong negative performance impact.

The depth camera provides distance readings, and thus is preferred to monitor respiration related movements. However, the resolution and noise of depth cameras

degrade easily with the distance, and at the 4-5 meter range used in our setting the PS1080 provides very noisy data. Only by considering the sequence as a whole, the main spectral component can be recovered.

Trajectories from the infrared camera represent the intensity variation of a pixel over the sequence. Although its performance depends strongly on the environment, light conditions, and available texture, it is intrinsically less affected by the distance to the chest, and has a significantly better effective resolution (for the effective resolution of PS1080 cameras see Appendix A). Furthermore, infrared cameras are less expensive than depth cameras, so it is worth to evaluate its performance.

We boost the signal-to-noise ratio by dropping out non-relevant pixels using a ROI.

We suggest to use a dynamic ROI that is anchored to the bed position. This is necessary for unattended monitoring, as our camera is fixed to the ceiling, but beds may change position often (*e.g.*, in hospitals and nursery homes).

The bed is automatically located once per night using the BAMs algorithm and the region of interest is centered on the chest area, as seen in Fig. 6.4.

We also evaluate a fixed ROI. The fixed ROI is equivalent to the dynamic ROI, only that the bed position is not updated for each recording, and thus it is not dynamically aligned to the bed.

To generate a strong signal we must fuse as many trajectories as possible. There are two main trends on how to perform the fusion. We can simply aggregate all temporal trajectories, hoping that the noise gets canceled while the main signal raises above the noise level. Or we can do some intelligent fusion using either Principal Component Analysis (Prathyusha et al., 2012) or Independent Component Analysis (Poh et al., 2011).

In both cases, it is assumed that the signal we capture is in phase on all trajectories, however this is not the case for breathing. As we breath, we move our environment, parts of the bed clothing rise, while other parts sink. Those parts would have the same frequency, but different phase.

This is the main motivation behind our Early Fourier Fusion technique. Instead of fusing temporal trajectories and estimate the PSD of the fused trajectory, we first estimate the PSD of each individual trajectory before fusing them. As the PSD effectively eliminates the phase of the signal, it ensures that there is no destructive inference between two useful signals.

Before aggregation, we apply a very conservative filter to each PSD: we keep only the trajectories whose mean power/Breaths per Minute (BPM) is larger inside the region between 6 BPM and 60 BPM than outside. This effectively limits our range of detected frequencies from 6 BPM to 60 BPM, which is an acceptable range to detect using a 30 second window.

As our system performs continuous monitoring instead of independent measurements, we need to deal with cases where the bed is empty, or the patient is coughing,

talking or changing sleep positions, etc. A nurse would not measure breathing rate in those situations where the outcome would not be considered reliable. Thus an automated monitoring system should not report the breathing rate in those situations. However, this raises the challenge of detecting unreliable situations.

We suggest a confidence test based on the agitation of the patient (explained in detail in Section 7). If we detect minimal levels of agitation, we expect the bed to be empty and thus we discard the segment. If we detect excessively high levels of agitation, we expect that a singular event happened (change in sleep position, coughing, etc.), and we discard the estimation.

6.2.2. Evaluation

We use acceptance curves for evaluation. The **x** axis is the acceptance threshold, while the **y** axis is the percentage of samples that provide an estimate within the acceptance threshold to our reference.

As a baseline, we evaluate how well can we predict our reference breathing rate, obtained from the nose thermistor, using the other polysomnogram signals as a source (see Fig. 6.6.a).

We can see that an agreement better than 1 BPM is achieved in only 87% of the sequences.

Algorithm

Sensor input

Our results show how the depth camera provides significant better performance than infrared images, as expected (see Fig. 6.6.b). In the hospital setting, where our dataset was captured, the bed clothing has very little texture where changes in the infrared image can be tracked. Therefore we use the depth camera as default.

Region of Interest

Our experiments show that using a ROI is critical to obtain good results (see Fig. 6.6.c). In our dataset there is little variation between bed positions, hence the fixed ROI, whose position is fixed relative to the camera field of view, provides a large performance boost. However, the dynamic ROI, whose position is relative to the bed localization, is even better.

These results suggest that there is still room for improvement when selecting the right ROI size. The ROI size needs to balance two properties. A large ROI will capture breathing in segments where the patient is lying on the edges of the bed, or in less common positions. But a smaller ROI would provide better signal-to-noise ratio, and thus improve the accuracy if the patient is breathing faintly. As a consequence, we expect that adapting dynamic also the ROI size would improve the results.

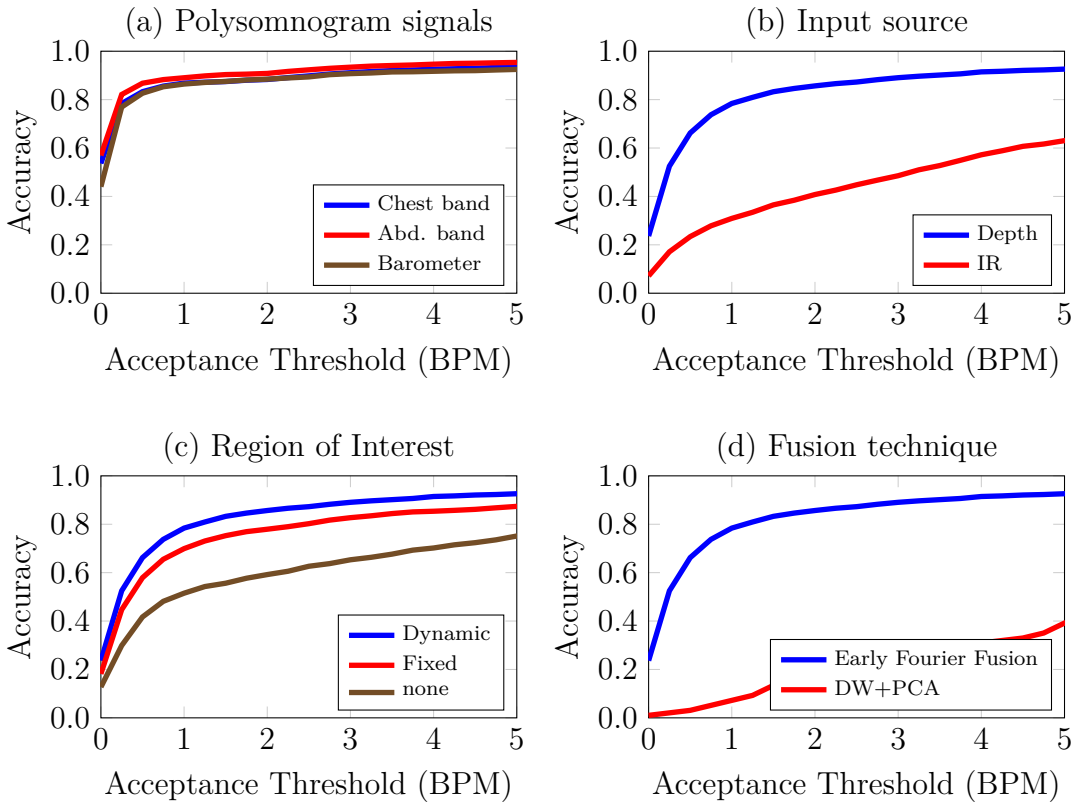


Figure 6.6.: (a) breathing rate predicted from other polysomnogram signals is accurate to ±1 BPM only in 87% of the sequences, (b) the depth camera performs significantly better than an infrared camera as a source, (c) the dynamic ROI provides a performance edge over the fixed ROI, (d) our Early Fourier Fusion outperforms DW+PCA in this case.

Merging Style

Given the distance between the camera and the patient, PCA based algorithms do not provide acceptable results, in particular, if a denoise step like Durbin-Watson (DW) is used (see Fig. 6.6.d).

The DW statistical test is meant to remove trajectories with low autocorrelation, however the minimum resolution of the depth camera at such distances is so large that the signal appears randomly dithered, and thus shows very little autocorrelation. The same applies for the correlation between different trajectories, which is minimal, hence the failure of the PCA based methods.

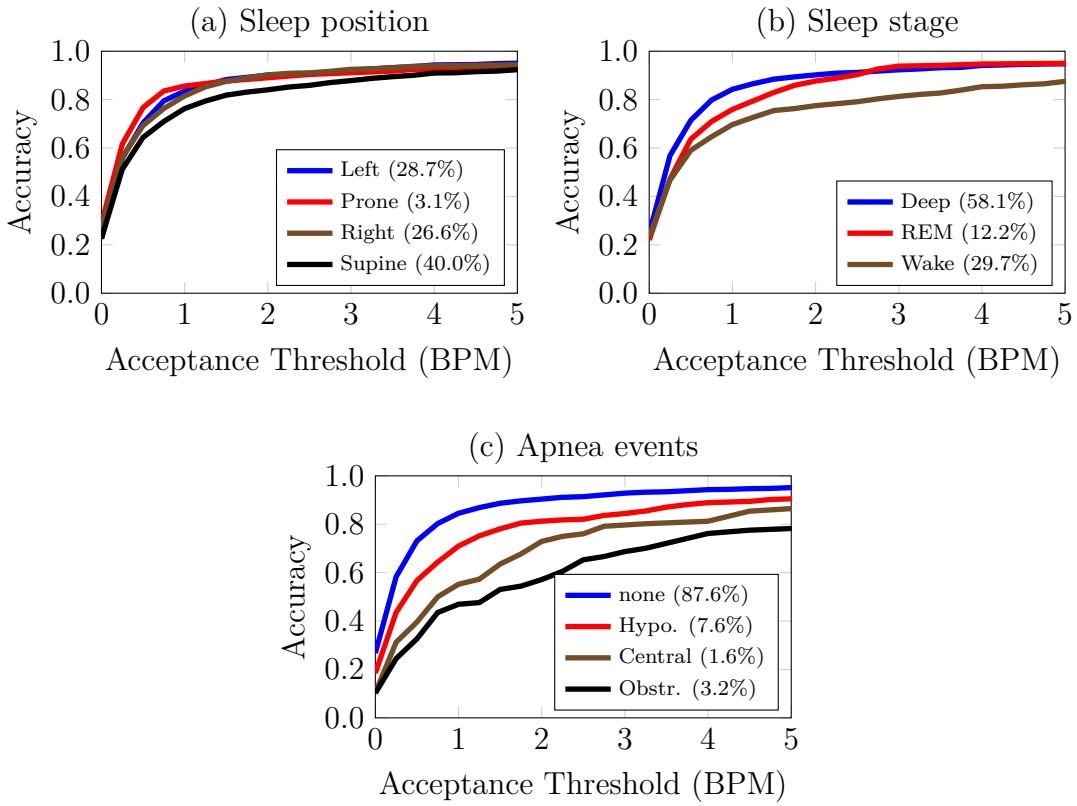


Figure 6.7.: Accuracy under different sleep conditions: (a) sleep position has no significant performance impact, (b) the algorithm performs better if the patient is sleeping, (c) the performance of the system is degraded during apnea events, in particular during obstructive apnea. The number enclosed in parentheses is the incidence rate of each class.

Sleep Conditions Analysis

Sleep Position

Although the breathing motion of the chest is not directly observable if the patient is lying in a lateral position, the motion is transferred to the surroundings of the patient (bed clothing, arms, pillow) where it can be observed. Hence, the sleep position of the patient has no significant impact on the accuracy (see Fig. 6.7.a). Albeit the supine position shows the worst performance, this happens because most apnea events occur while the patient is in supine position.

Sleep Stage

Results for sleep stage accuracy exhibit our expected behavior as the system performs better if the patient is relaxed (see Fig. 6.7.b). At the same time, patients have

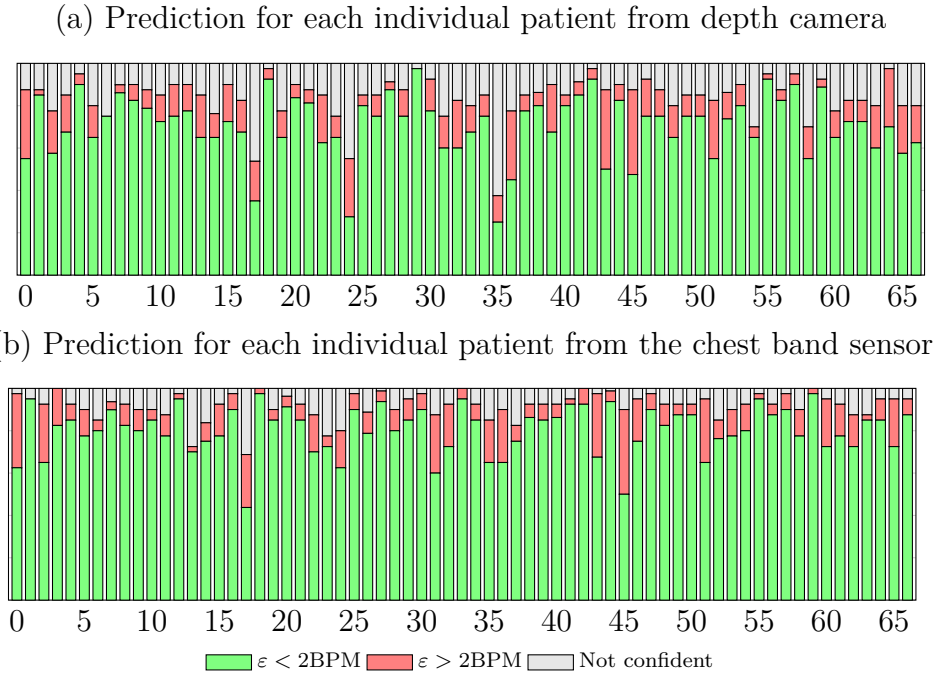


Figure 6.8.: We evaluate our system's performance on each patient. Top: results from our algorithm (accuracy 85.9%). Bottom: Results from the chest band used in the polysomnogram (accuracy 88.7%). Note the behavior similarity between the depth camera and the chest band sensor.

difficulties to sleep deep when having apneas, thus deep sleep is correlated with sequences with low rate of apneas.

Apnea Events

Apnea events have a large accuracy impact (see Fig. 6.7.c). Hypopneas are defined as a 30% reduction of air flow for more than 10 seconds, therefore breathing is usually shallow, but nonetheless existent. Central apneas represent a breathing pause larger than 10 seconds, which then resumes uneventfully. As 10 seconds are a significant part of the 30 second segment we use, the signal degradation is larger.

Obstructive apneas happen when the upper respiratory tract is obstructed: the diaphragm tries to expel the air from the lungs, but only manages to send it to the stomach, where it is sent again to the lungs. This stage is known as paradoxal breathing, and is manifested in the polysomnogram as a phase shift between chest and abdomen breathing signals. This stage is interrupted when the oxygen saturation in the blood drops enough to arouse the patient, who makes an conscious effort to unlock the airways. The breathing rate is non regular, and thus do not express a large peak on the spectra.

Per Patient evaluation

We evaluate our system predictions for each patient, and compare to the breathing rate predictions from the chest band. In our representation, each sequence is color coded. Green represents that the sequence produced a confident estimation, and it was within 2 BPM of the reference, the thermistor. Red means that the sequence produced a confident estimation, and it was not within 2 BPM of the reference. Gray indicates that the sequence did not pass the confidence test (see Fig. 6.8).

This way we can evaluate if the results are consistent among all patients. We found out that, for most patients, our system behaves like the chest band. However we have two outliers: patient 24 and 35 were significantly better recognized using the chest band than from the depth camera. Two patients out of 67 are not significant enough to extract conclusions, therefore further analysis is required.

6.3. Respiratory Event Recognition

Next, we address the task of recognizing noteworthy respiratory events in the THX dataset, and, in particular, sleep apnea events. This dataset is captured in a sleep laboratory, contains real patients with different degrees of sleep apnea syndrome, and is annotated by sleep medicine doctors.

Recognizing sleep apnea events is one of the most difficult tasks addressed in this thesis. Sleep doctors do not use any assistive technology to aid in recognizing them, and they label all events by hand using the information provided by the polysomnogram.

A complete solution for this task must have two steps: first, to find the unusual respiratory events, and second, to classify them. However, to the best of our knowledge, no method has been published that successfully addresses the problem in this terms, even if all polysomnogram sensors are available to them.

The main problem is that the definition of sleep apnea events is inherently subjective. The American Academy of Sleep Medicine (AASM) provides a guideline to identify each respiratory event. This guideline is designed to build a common framework for all the sleep laboratories in the world, but many of the parameters used in this guideline are open to multiple interpretations to the inexpert eye, and many change dramatically between guideline updates.

To highlight the diagnostic difficulties, we explore the problem of detecting hypopnea. The definition of hypopnea according to the guideline (and simplified), is a respiratory event where the respiration intensity drops by more than 30% with respect to normal levels, lasts more than 10 seconds, and affects the O_2 blood saturation levels. Some doctors use airflow pressure to measure respiration intensity, some others use plethysmography (*i.e.*, variation in the volume of the lungs), and a difference of 30% in both metrics is wildly different. Most problematic is to assess what is

"normal levels", should those "normal levels" be the usual respiration intensity in supine position?, or we should have "normal levels" for supine, lateral, and prone positions? What to do in borderline cases?

In real world conditions we have found many labels for hypopneas with a duration of less than 10 seconds. And last but not least, the majority of respiratory events do not fit in a single category, and thus doctors label them as a combination of two categories, or simply as a "mixed apnea" event.

There have been attempts to recognize specific apnea events from the chest and abdomen movements (*i.e.*, Várady et al. (2003) detects a very specific pattern that occurs only during obstructive apnea.). Others claim to recognize apnea events when in reality they simply ask volunteers to stop breathing for a few seconds (Al-Naji et al., 2017). But the problem of recognizing automatically sleep apnea events in real settings remains unsolved.

Still, detecting apnea events is not the main goal of sleep studies. This is just a means to diagnose and stage the gravity of the sleep apnea syndrome. As detecting individual events is challenging, it is common to relaxed the problem statement and simply classify patients as healthy or sick. In this case, most successful approaches use a combination of nasal airflow, ECG, and pulsoxymetry, as seen in (Várady et al., 2002; Burgos et al., 2010; Tian and Liu, 2005; Sola-Soler et al., 2007; McNames and Fraser, 2000) and apply machine learning approaches, mainly Support Vector Machines (SVM) and Artificial Neural Networks (ANNs).

Detecting apnea events is even more complex if we only have access to visual data. Not only we do not have access to vital signs, but visual information is also too complex to be trained using only one binary label per patient (healthy / unhealthy). We must use the labels for individual apnea events provided by sleep medicine doctors.

On the THX dataset we have labels for the following events:

Central Apnea. Respiration simply stops during a small period of time and then resumes normally. It has usually neurological causes.

Hypopnea. The respiration intensity is reduced by 30%.

Obstructive Apnea. The upper airflow gets blocked while the chest and abdomen move violently trying to clear the blockage. The struggle continues until a the blood O_2 saturation drops enough to wake the patient and force him to grasp air. It is usually related to overweight.

Mixed events. As we mentioned earlier, the majority of obstructive events do not fit into a single class and are thus classified as a combination of classes or as a mixed apnea event.

Others. Our dataset has a few labels for Respiratory Event Related Arousal (RERA) and Cheyne-Stokes events, but they have an combined occurrence rate of less than 0.15%. The occurrence rate of apnea events in the THX dataset can be seen in Table. 6.2.

Breathing Condition	Occurence Rate
Central Apnea	1.64%
Central Hypopnea	0.59%
Cheyne-Stokes	0.06%
Hypopnea	1.35%
Mixed Apnea	0.56%
Obstructive Apnea	3.16%
Obstructive Hypopnea	5.67%
RERA	0.09%
Normal	86.88%

Table 6.2.: Ocurrence rate of different breathing conditions in the THX dataset.

	CA	CH	H	MA	OA	OH	N
Central Apnea	13.44	12.10	11.02	3.76	11.83	45.70	2.15
Central Hypopnea	29.53	27.46	3.63	3.11	0.00	35.23	1.04
Hypopnea	8.87	18.28	28.23	4.57	2.69	36.02	1.34
Mixed Apnea	6.16	13.49	22.87	2.05	13.49	41.64	0.29
Obstructive Apnea	8.27	7.20	14.67	12.27	9.07	46.13	2.40
Obstructive Hypopnea	11.76	12.57	4.81	8.29	6.68	54.81	1.07
Normal	13.94	19.57	13.14	5.63	3.22	42.89	1.61

Table 6.3.: Confusion matrix of a LSTM+CNN based classifier on sleep apnea events.

Our first efforts to classify individual sleep apnea events obtained mixed results, as expected for a task of this level of complexity. We prepared a balanced training set where each sample is a 60s sequence of BAMs sampled at 5 fps. We first tried to classify each sequence using its Power Spectral Density (PSD), as it a popular method used in the published literature to diagnose sleep apnea from vital signs. However, our modality is more complex to analyze than single dimensional vital signs, and our PSD based approach did not achieve any meaningful results. Eventually, our effort in this area resulted in the development of a new technique to train deep learning networks using the Earth Mover's Distance (EMD) as a loss function that we present in Appendix C.

Next we analyzed the sequences using a model based on Long Short-Term Memory (LSTM) combined with Convolutional Neural Networks (CNNs), obtaining the results shown in Table. 6.3. Our model struggles to learn the concept of obstructive apnea, as its relatively violent and unstructured behavior makes it each event unique, and, in turn makes it difficult for the model to learn a general description for the problem. The same problem happens for the mixed apnea class. The network also has problems

to recognize normal breathing. Those results highlight that CNN have problems to recognize uninteresting and weakly described events.

We are in the process of reformulating the problem in a way that is simpler to provide meaningful results, but the model benefits from the individual labels we have for sleep apnea events. For example, we have trained a Gated Recurrent Unit (GRU) + CNN network that takes a sequence of 5 minutes and classifies it into two classes: *severe apnea* and *not severe apnea*, achieving an accuracy of 70.37%.

7. Analysis of Sleep Quality Cues

In this chapter, we propose novel methods to analyze meaningful cues for sleep quality assessment. We analyze in depth agitation, bed occupancy, sleep actions, and sleep position, and also show promising results for sleep staging, leg movement detection and facial stress detection.

Agitation and bed occupancy are analyzed using new but straightforward metrics. Our main novelty there is that the metrics we designed are objective, have a clear physical meaning, and are Point of View independent.

By displaying agitation, bed occupancy, and our facial stress indicator, we can provide a meaningful and compact visual summary of a night, thus compacting 8+ hours of information in a single picture.

On the other hand, actions, staging, position and leg movements are relatively more complex and must be inferred using machine learning techniques, of which we have evaluated a wide range, including Large Margin Nearest Neighbor (LMNN), Support Vector Machines (SVM), and Convolutional Neural Networks (CNNs).

To develop our machine learning models we need a large amount of data, but the low information density of our task (*e.g.*, less than 1 change in sleep position per hour) makes it impractical to train our models using raw depth data. We require the high information density of BAMs to build compact datasets of reasonable size. Therefore, most of the algorithms presented in this chapter use BAMs as the sole input feature.

7.1. Basic Sleep Quality Cues

7.1.1. Agitation

Agitation is the main indicator used in several automated sleep monitoring systems. It is reasonably straightforward to measure using a variety of methods, and can be used to infer information like sleep schedules, and total sleep time.

More critically, agitation is used in ICUs to adjust the sedation protocol to each patient's needs. The goal is to keep the patients relaxed but mentally conscious. This is, by no means, an easy task. Oversimplifying: too little sedation rises the risk of having aggressive events that can be self-damaging for the patient, while excessive sedation can induce *delirium*, which is a potentially-deadly medical condition characterized by excessive torpor. The physical and medical attributes of a patient

are used to design the initial sedation protocol, but the protocol must be adjusted continuously based on medical and behavioral cues, like agitation events.

Medical equipment is used to monitor vital signs, but behavioral monitoring depends on the notes registered by the medical staff. In contrast to vital signs, there are no objective ways to measure behavior. This results in wide disparities in the reports from different medical units.

Actigraphy, the measurement of physical activity, has been suggested as an indicator to be included in sedation scales (Sadeh and Acebo, 2002). Compared to the visual reports by nurses, actigraphy methods offer a better coverage and thus constitute a significant step in the right direction. However, actigraphy currently still depends on subjective metrics to quantify the body movement (*e.g.*, (Sadeh and Acebo, 2002) count the number of significant body movements per hour, but what does constitute a significant body movement?). This lack of a "*golden standard*" to quantify body movement has hindered the widespread use of actigraphy in ICUs (Chanques et al., 2006).

Agitation is the main indicator used in sleep monitoring systems that are based on vision. Due to the difficulty of localizing precisely the body, it is generally quantified by measuring changes between consecutive images. This approach is not robust to changes in illumination (Mansor et al., 2010b), although light invariant feature descriptors have been used to compensate global illumination changes (Reyes et al., 2010). But the main problem is that the measures obtained are only locally consistent (both temporally and spatially). Results from different beds are not comparable due to different Point of View (POV) of the camera, and even using the same camera and bed, just changing the bed clothing can completely alter the result.

To summarize, agitation is one of the most important cues in sleep monitoring, but the lack of a consistent metric to measure it hinders its widespread usage.

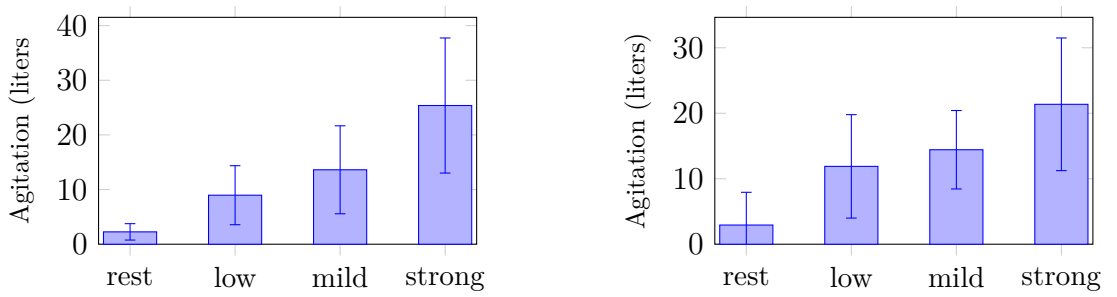
To solve this problem, we suggest a novel metric to measure agitation which is consistent both temporally and spatially and has a clear physical interpretation.

Our metric is defined as the mean difference between the maximum and the minimum volume occupied above the bed within one second. Being a volume metric, it is measured in cubic meters. We can calculate it trivially from BAMs.

Lacking a standard procedure to measure agitation, we used the CV:HCI dataset where we asked the subjects to show three different levels of distress in supine and fetal positions. To compare between agitation levels we averaged the measurements corresponding to 5 seconds.

The only instructions our subjects received were to show *low*, *mild* and *strong* distress; this results in wide disparities between subjects but the average measured agitation showed a consistent progression between intensity levels (Fig. 7.1).

We also evaluated the effectiveness of this indicator in a classification task, and the resulting confusion tables can be seen at Table. 7.1. The results not only show that



(a) Supine (b) Fetal

Figure 7.1.: Mean and standard deviation of the agitation values of subjects when instructed to rest or show a low, mild and strong distress respectively.

	rest	low	mild	strong
rest	23	0	0	0
low	0	21	2	0
mild	0	1	18	4
strong	0	1	3	19

(a) Supine

	rest	low	mild	strong
rest	21	1	1	0
low	1	16	4	2
mild	1	3	14	5
strong	0	3	4	16

(b) Fetal

Table 7.1.: Agitation classification within a subject using the suggested metric.

our metric is well correlated with our indications for *low*, *mild* and *strong* distress, but our subjects behaved in a surprisingly consistent way.

7.1.2. Bed Occupancy

Bed occupancy is a common indicator captured by sleep monitoring systems. It is possibly the simplest sleep indicator, and is usually captured from pressure sensors, although many smartphone applications estimate also bed occupancy based on accelerometer data.

Bed occupancy is used to analyze sleep patterns, and amount of sleep.

Computer vision techniques are less successful to estimate bed occupancy in a robust way due to the many possible bed occlusions and the different textures that beds present. Common methods estimate bed occupancy by detecting the patients head using skin color models like (Mansor et al., 2010a), or place markers in the scene like (Becouze et al., 2007).

On the other hand, we can use BAMs to estimate the amount of volume occupied above the bed and use it as our bed occupancy metric (see Fig. 7.2). Like our agitation metric, our bed occupancy metric is robust to light, POV, and texture changes.

7.1.3. Facial Stress

Often, the only body part that is not occluded during sleep is the face, and thus face analysis has been suggested to recognize stress and agitation.

However, all published methods are based on simulated experiments that have little resemblance with real settings. Both (Becouze et al., 2007) and (Mansor et al., 2010a) use skin color information that is only available if the face is illuminated by light in the visible range, furthermore they only test on frontal faces of very high resolution.

On the other hand, the facial images we actually obtain from a ceiling mounted camera are generally not frontal, as seen in Fig. 7.3. Also, color information is not available during the night. And finally, many elderly people with sleeping problems use a mask to improve their breathing. All those effects together make facial analysis during sleep a very challenging task.

Therefore, we suggest to measure facial stress using a simple but robust metric. In this case, we base our method on the same principles stated by (Becouze et al., 2007): stress is a dynamic feature that is related not to the aspect of a face, but to how the face changes over time.

Therefore, to measure meaningful changes on the face aspect, we use Bayesian Surprise, which has been used previously to detect salient events (Schauerte et al., 2011). Bayesian Surprise works by observing a signal over time and capturing its prior probability distribution. In this framework, the a new data observation generates a surprise value that is inversely proportional to its expected probability based on the prior model.

We resize each image from the face camera to 32×32 pixels and apply Bayesian Surprise to each pixel using a temporal Gaussian window mean of 25 frames (*i.e.*,

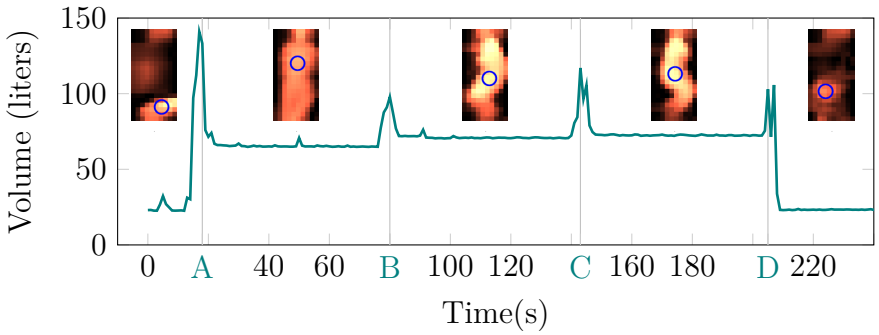


Figure 7.2.: Bed Occupancy: subject enters the bed (A), changes two times of sleeping position (B, C), and leaves the bed (D). Note how the volume never reaches zero as the pillow and the bed clothing occupy a significant amount of space.

Figure 7.3.: The face camera allows to record hi-res, hi-speed images of the face (top left).

500ms). We then aggregate the Bayesian Surprise of all pixels to create our facial stress index. Unlike our previous indexes, our facial stress index is unitless.

We have evaluated empirically our facial stress indicator in lab experiments (see Fig. 7.4), but due to the high difficultly of obtaining a meaningful reference data for facial stress, we could not evaluated it in real settings.

7.1.4. Long Term Sleep Summaries

One of our goals within the SPHERE project is to help assessing the sleep quality of nursery home residents over time.

This goal is generally achieved using only the bed occupancy metric, and current summaries are limited to statistics concerning the times of going to sleep and waking up, as well as the average quantity of sleep. We decided to go one step further and provide a visual summary based on both our bed occupancy metric and our agitation metric, this way we can observe in a single image several months at a time.

On our visual representation (see Fig. 7.5), each day is represented by two rows: red for bed occupancy and blue for agitation. Both metrics work together to show the big picture of what is happening during the nights.

Analyzing Fig. 7.5, we can see how the resident had a few very agitated days between 22.6.2015 and 25.6.2015. The red line for those days is almost constant, meaning that the resident did not leave the bed, but the blue line shows large amounts of agitation. Those days correspond to a heat wave.

On the other side, towards the end of the study the resident could not sleep for many hours at a time, even leaving the bed. This corresponds to regions with almost neither red or blue.

To the best of our knowledge, we are the first to provide those kind of summaries, and thus we still need to develop methods to evaluate them in an objective way, but both sleep medicine doctors and health care practitioners have expressed positive views about them.

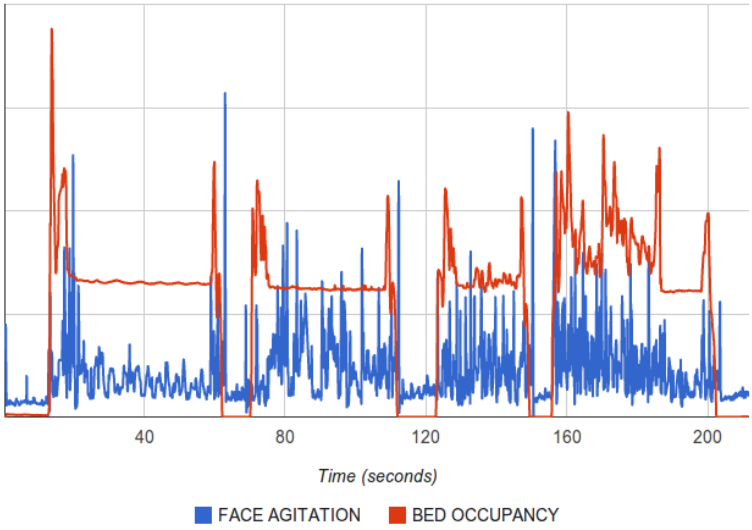


Figure 7.4.: This sequence simulates 4 scenarios. 10s-60s: Sleeping relaxed shows an almost flat bed occupancy indicator and low agitation levels in the face. 70s-110s: Sleeping with pain expressions is not reflected in the volumetric information, but it is detected by the face agitation levels. 120s-145s: Being restless in bed is reflected by an clear response in both indicators. 145s-200s: Strong compulsions ending with an accident and sudden loss of consciousness.

7.2. Actions

In this section we explore the viability of the BAM representation to perform action recognition in elderly care scenarios. In some circumstances, action recognition approaches offer a high level of scene understanding compared to the simplistic methods presented previously (see Weinland et al. 2011). Most approaches are based on motion patterns with space time interest points (Laptev et al., 2008; Wang et al., 2011), and recently, if 3D data is available, the reconstructed body pose is also used (Ohn-Bar and Trivedi, 2013; Vemulapalli et al., 2014).

We use the CV:HCI dataset for training and testing. We analyzed the instructions that the volunteers received and selected the ones that correspond to a recognizable action or activity. For those selected instructions, we segmented the 10 seconds following the delivery of the instruction. Using a rate of 10 BAMs per second each segment is 100 BAMs long. We clustered the instructions that belong to the same action, ending up with 17 different actions.

Early during our research, we observed that there is a clear hierarchical overlap between actions. For example, changing bed positions or leaving the bed implies some sort of agitation, and at the higher level, interacting with a nurse usually implied a change in the resting position. To deal with this overlap of actions, we split them in

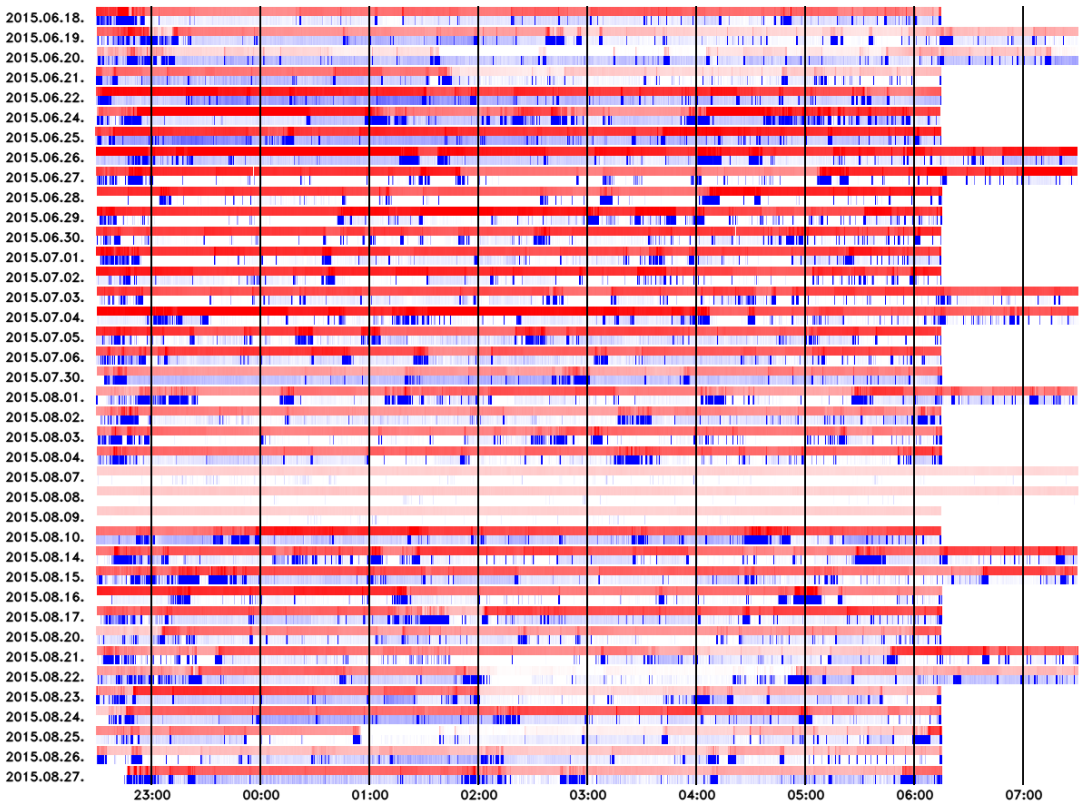


Figure 7.5.: 39 night sleep summary of a nursery home resident. Red bars correspond to bed occupancy, showing that the patient spent long hours outside the bed towards the end of the study. Blue bars indicate agitation.

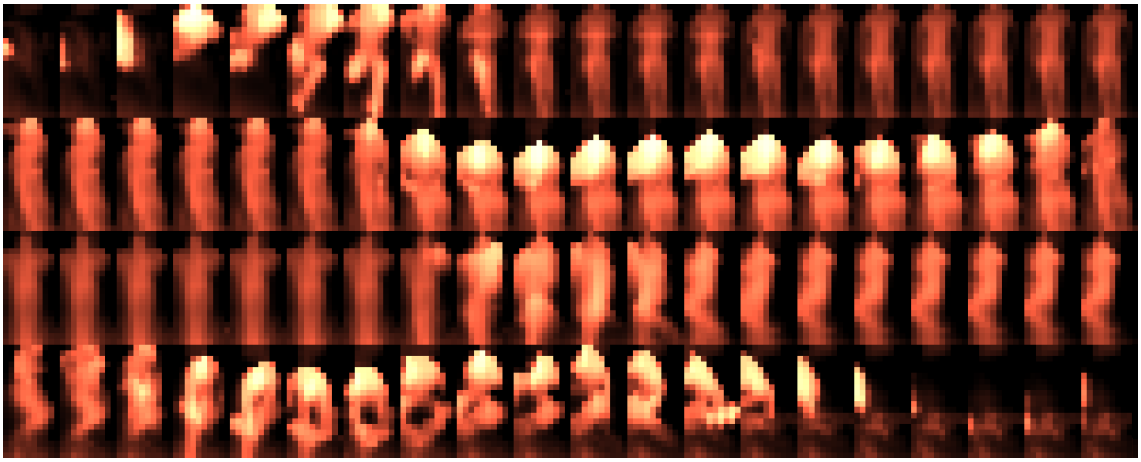


Figure 7.6.: BAMs offer a synthetic representation that protects the privacy of the user while providing enough information to recognize common actions such as: getting inside the bed (1st row), drinking water (2nd row), change sleeping position (3rd row), and leaving the bed (4th row).

Action	Sequences
Agitate Low	42
Agitate Med	42
Agitate High	42

Table 7.2.: Low Level Actions.

Action	Sequences
Get Into Bed	42
Supine To Right	42
Right To Left	42
Leave Bed	42

Table 7.3.: Mid Level Actions.

Action	Sequences
Nurse Arrives	21
Nurse Manipulates	21
Nurse Leaves	21
Repetitive Movements	42
Shouting	21
Bed Cover Manipulation	21
Infusion Lines Manipulation	21
Touch Mouth	21
Drink Water	21
ManipulateObject	21

Table 7.4.: High Level Actions.

three different problem sets: agitation (Table. 7.2), sleep position changes(Table. 7.3), and high level actions (Table. 7.4). Each of the problems are evaluated separately.

We encode each image directly using BAMs, however, as we are dealing with a problem of dynamic nature, we further calculate the difference between BAMs in subsequent frames, the result is further denoted as difference Bed Aligned Maps (dBAMs).

Whole action sequences are encoded as a Bag of Words (BoW), which has been shown to yield state-of-the-art results in many action recognition tasks (Wang et al., 2011). To this end, we first learn a 1000-word codebook via k-means clustering. Then we either apply Vector Quantization (VQ) or Locality-constrained Linear Coding (LLC) with sum-pooling to obtain a BoW representation of each sequence. Since, it has been shown that power normalization can increase the discriminative power of a feature vector (Arandjelovic and Zisserman, 2012), we first normalize

Feature	Encoding	high (10-class)	med (4-class)	low (3-class)
BAM	VQ	52.4	92.1	50.0
BAM	LLC	62.8	93.4	**57.1**
dBAM	VQ	61.9	94.5	50.0
dBAM	LLC	66.2	86.1	48.4
BAM+dBAM	VQ	62.0	94.5	50.0
BAM+dBAM	LLC	**67.5**	**97.0**	55.6

Table 7.5.: Action classification accuracy for VQ and LLC using as features BAM, dBAM, and their combination.

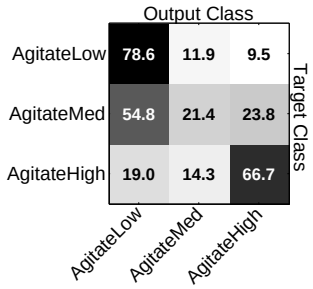


Figure 7.7.: Confusion matrix for three different agitation levels for LLC encoded BAM+dBAM (accuracy of 55.6%). The action recognition framework works better for clearly defined actions.

the features to unit length and then raise each element of the feature vector to the power of 0.3. Finally, we standardize the features to zero-mean and unit-variance before using a linear multi-class SVM for action recognition. In our experiments we observed that features based on BAMs and dBAMs appear to have complementary properties, hence we fuse BAM and dBAM features by concatenating their BoW encoding.

Our results can be seen in Table. 7.5, and show that LLC performs better than VQ, and the combination of BAM and dBAM features perform better than the individual ones. Therefore, from now on we will discuss the results obtained using LLC and the combined BAM+dBAM features.

The results on the low level actions are weak but expected (see Fig. 7.7). Our action recognition framework works better on non overlapping actions that are clearly defined, and it has difficulties to recognize actions that are based in more abstract concepts.

The mid level actions are clear and distinctive, and therefore we achieve an accuracy of 97.0% on them (see Fig. 7.8).

The results on the high level actions show two different behaviors. Reasonably unique and distinctive actions, such as nurse interactions, manipulating of the bed cover, and the infusion lines, achieve great results. However, there is not enough

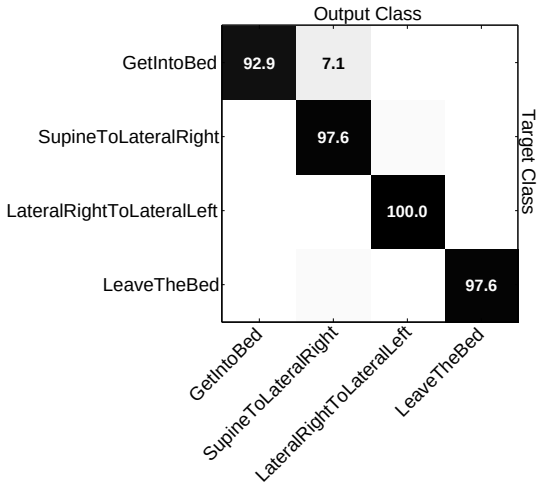


Figure 7.8.: Confusion matrix of the bed related actions for LLC encoded BAM+dBAM. Those actions are clearly defined, and easily recognized with an accuracy of 97.0%.

training data to model accurately weakly described actions such as "repetitive movements". Finally, the BAM representation is probably not strong enough to distinguish clearly between grabbing an object and manipulating it, and grabbing a cup and drinking water from it, and therefore there is significant confusion between those two classes (see Fig. 7.9).

7.3. Sleep Stage

In this section we task of sleep-wake classification, as well as distinguishing between different sleep stages.

Sleep can be divided into two very distinct types: Rapid Eye Movement (REM) and Non-Rapid Eye Movement (NREM). REM corresponds to light sleep and it is when dreams occur, conversely, NREM corresponds to deep sleep. The AASM further divides NREM into three stages: N1, N2 and N3. The task of classifying the sleep pattern into their correspondent classes is called sleep staging.

Sleep patterns are generally recognized from the very distinctive brain waves that each class produces, therefore sleep analysis is strongly based on EEG data, while EOG and EMG are also used as a complement.

However most sleep laboratories still perform sleep staging manually, as automated methods are not accurate enough for medical use, even if the EEG, EOG and EMG data is avilable to them. There is still much research being performed in automated sleep staging from EEG data, like in (Hassan and Bhuiyan, 2017).

If automated sleep staging from EEG is far from being solved, using less invasive methods is still more challenging.

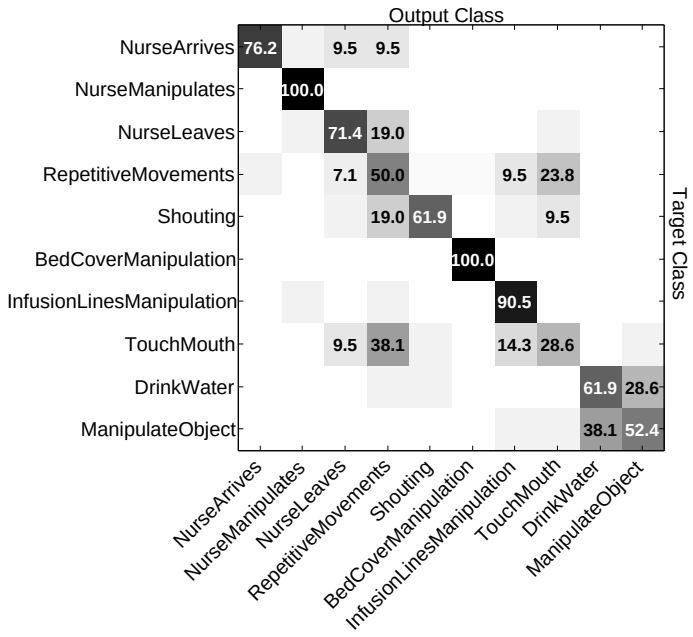


Figure 7.9.: Confusion matrix of the higher level actions for LLC encoded BAM+dBAM (overall accuracy of 67.5%). We can observe a split behavior where some actions can be detected reliably, while others are easily confused. It is noteworthy that the *drink water* and *manipulate object* actions, which use similar movements, are only confused mostly between themselves.

Non invasive methods rely in a very simple heuristic: if we are awake we tend to move more than if we are sleeping. Many approaches have been developed based on this idea, but the seminal work in the area is due to (Cole et al., 1992). Cole uses a activity monitor placed on the wrist that records the amount of movement that a subject experiences during the night (actigraphy), and classifies each minute of the recordings as sleep or wake simply by thresholding.

Actigraphy methods have reached an accuracy of 91% when compared to manually staged data, but show a very low specificity of 34%, as seen in (de Souza et al., 2003). This limits significantly the usefulness of this methods in practical settings.

Still it is useful to know how well can we stage sleep from our distance sensor. In our setup, we try to estimate sleep stage from a 1 minute sequence of BAMs by the means of an LSTM based neural network (see Fig. 7.10). LSTM are a variety of Recurrent Neural Network (RNN) that encodes well temporal models, but avoids the common problem of vanishing gradients that plague vanilla RNN.

We use our THX dataset which is labeled by sleep medicine doctors with sleep stages annotations. 60 of the patients were used for training and 16 for testing.

When using our network to classify between all labeled sleep stages, the network recognizes fairly well the awake state, but classifies all sleep stages with a strong bias

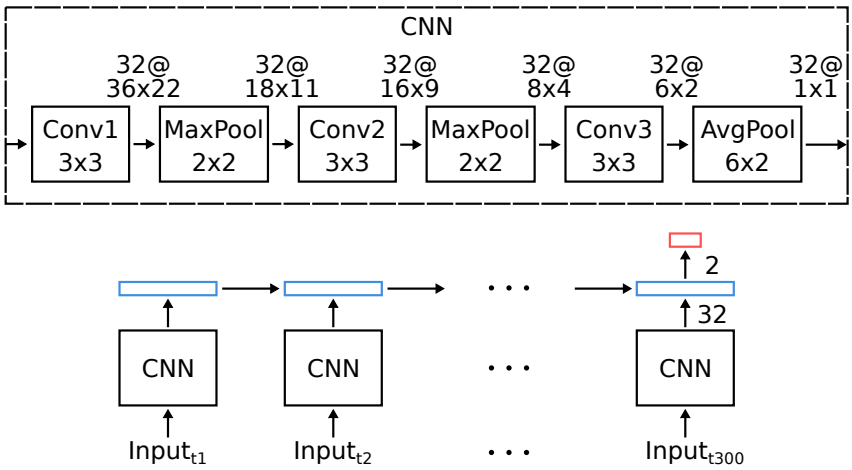


Figure 7.10.: The architecture we used to classify sleep stages. We perform the task on a temporal input size of 300 BAMs, which corresponds to 1 minute.

	N1	N2	N3	REM	Wake
N1	0.00	27.27	0.00	68.18	4.54
N2	0.00	31.91	0.00	61.70	21.28
N3	0.00	16.36	0.00	67.27	16.36
REM	0.00	6.25	0.00	85.42	8.33
Wake	0.00	12.28	0.00	8.77	78.95

Table 7.6.: Confusion matrix of our LSTM + CNN classifier on all labeled sleep stages.

	NREM	REM	Wake
NREM	53.33	24.85	21.82
REM	46.67	33.33	20.00
Wake	17.58	6.67	75.76

Table 7.7.: Confusion matrix of our LSTM + CNN classifier when collapsing all NREM stages into one.

towards REM, as seen in Table. 7.6. This result is expected due to the difficulty of distinguishing between similar sleep stages by means only of visual cues.

Then, we tested if we could, at least infer the slight differences between NREM and REM sleep stages, thus we trained our classifier after fusing N1, N2 and N3 stages as NREM. Results shown in Table. 7.7 confirm that the differences are too slim to be recovered only by visual cues.

Finally, we tested the simplest scenario by classifying only between *sleep* and *awake* classes, where we achieved an accuracy above 80% (see Table. 7.8). Due to the very challenging environment that is the sleep laboratory, we deem this result as

	Sleep	Wake
Sleep	82.43	17.57
Wake	19.31	80.69

Table 7.8.: Confusion matrix of our LSTM + CNN classifier only on sleep-wake classification.

quite satisfactory, however we can not compare against actigraphy methods as no actigraphic sensor was used during the recording.

7.4. Sleep Position

In this section we analyze the task of recognizing sleep positions, which are related to several sleep disorders. For instance, sleep apnea is observed with much higher frequency in supine position, as seen in (Cartwright, 1984; Penzel et al., 2001; Richard et al., 2006). Hence, according to Itasaka et al. (2000), shifting the body position during sleep is an effective medical treatment for sleep apnea. Nakano et al. (2003) notes that sleeping in supine position is also linked to snoring. Finally, sleep position is monitored in ICU and nursery homes to prevent pressure ulcers (Maklebust and Sieggreen, 1996).

In sleep laboratories the sleep position is commonly registered using a sensor attached to the chest that measures the direction of the gravity. This intrusive approach is valid for a one-time study but not suitable for long-term monitoring.

As with most sleep quality cues, sleep position is not uniquely defined. Several approaches, *e.g.*, (Huang et al., 2010; Lee et al., 2015; Yu et al., 2013), are based on the United Kingdom Sleep Assessment and Advisory Service (SAAS) who clustered sleep positions in six classes: fetus, log, yearner, soldier, freefaller and starfish as see in (Idzikowski, 2003). Sleep laboratories use a simpler classification: supine, left, right and prone, as seen in (Kuo et al., 2004). And the most pragmatic systems simply classify between supine and lateral positions, like (Yu et al., 2013).

The methods and algorithms used to classify between sleep positions are also varied.

Smart beds are a commonly used to monitor sleep position non intrusively. Motion sensors are placed inside the pillow (Harada et al., 2000) or onto the bed itself, as in (Hoque et al., 2010; Malakuti and Albu, 2010). (Hoque et al., 2010) uses Radio-Frequency IDentification (RFID)-based sensors equipped with accelerometers that are attached to the bed mattress.

There are camera based approaches that use color (Huang et al., 2010), infrared (Kuo et al., 2004), thermal, depth cameras((Yu et al., 2013), or a combination of them (Huang et al., 2010; Lee et al., 2015; Torres et al., 2016).

Most methods used are very crude compared to the state of the art in computer vision. (Yu et al., 2013) proposes an approach for the two class problem (supine and lateral positions) based on a depth camera attached to the bed. First head and torso are detected using ellipse fitting, then the position is classified as supine if the topmost pixel on the head is above the topmost pixel of the chest, and side-lying otherwise. Their algorithm is evaluated on 8 volunteers in a simulated experiment. (Lee et al., 2015) describes a system using an overhanging Kinect v2 sensor over the bed. They classify between SAAS positions. First they extract body joint positions using Kinect v2 own libraries, which are based on (Shotton et al., 2013) algorithm. They use the relative position of hands and knees with respect to the spine for classification using a parametric approach. The approach requires the patient to not use a blanket, and they do not provide evaluation results.

Most related to our approach, (Torres et al., 2016) uses a combination of depth and infrared cameras together with a pressure mattress to classify between SAAS positions. Only one scenario with a fixed camera above the bed is used, so alignment problems are not considered. They use Histogram of Oriented Gradients (HoG), from (Dalal and Triggs, 2005) and Modified Geometric Moments (MGM), from (Hu, 1962), as features. The descriptors are combined using coupled-constrained least squares to assign weights to each modality, and then multiclass classifiers based on SVM and Linear Discriminant Analysis (LDA) are used. Evaluation is performed on only 5 people in simulated scenarios.

Based on the data that is made available to us, we define our problem as a four-class classification task with right, left, and supine classes for sleep position, and an special case for the empty bed. Prone position is not considered as it is rare in real conditions and we did not have enough samples in our THX dataset for it to be statistically significant.

7.4.1. Analysis on the CV:HCI dataset

As a first experiment, we used our CV:HCI dataset to establish the difficulty of the problem of classifying sleep position from BAMs. Using a naive nearest neighbor approach with euclidean distance we obtain a accuracy of 85.9% using a leave-one-person-out cross-validation. This results were improved by using PCA to reduce BAM to 32 dimensions and LMNN (Kilian Q Weinberger, 2006) as a classifier. LMNN uses semidefinite programming to learn a Mahalanobis distance metric for k-Nearest Neighbors (kNN) classification. This combination achieves a 100% accuracy. Results of this experiment can be seen in Table. 7.9.

	empty	supine	left	right
empty	21	2	0	0
supine	0	20	3	0
left	1	3	18	1
right	0	0	3	20

(a) 1NN

	empty	supine	left	right
empty	23	0	0	0
supine	0	23	0	0
left	0	0	23	0
right	0	0	0	23

(b) PCA-LMNN

Table 7.9.: Confusion matrix of sleep position classification from BAMs on the CV:HCI dataset. Being a simple dataset, we obtained very high accuracies even when using simple techniques like nearest neighbor (left), and LMNN (right).

7.4.2. Analysis on the THX dataset

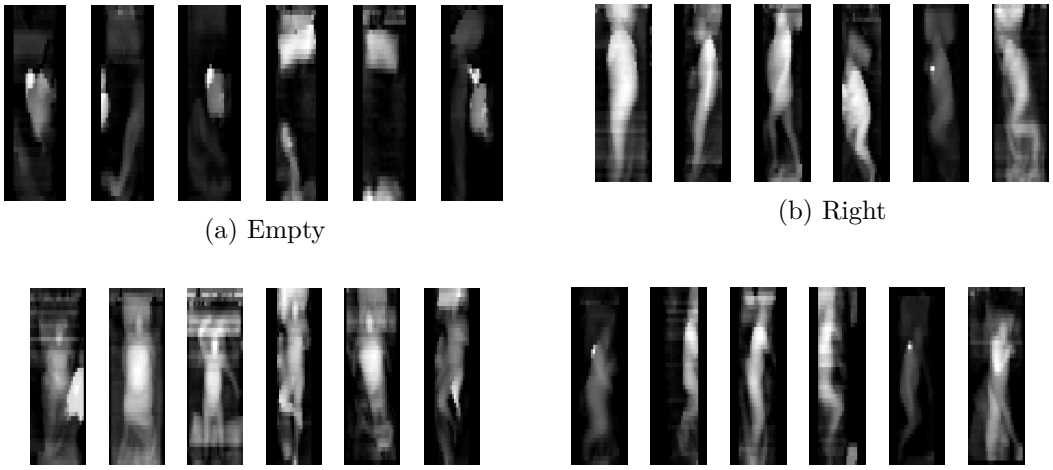
The THX contains people with significant sleep disorders, and a wide variety of ages and body types, as can be seen in Fig. 7.11. To assess the difficulty of this dataset it suffices to say that the gravity sensor attached to the chest has an accuracy of only 91.9% for sleep position classification.

Due to the lack of generalization capability of nearest neighbors approaches, the strategy we used in the CV:HCI dataset is quite unsuccessful here. Instead, we hypothesized that BAMs are still a good descriptor for the task, and we could replace the classificator by one of greater generalization capability, like CNNs. However, this has the challenge that CNNs require large amounts of data to properly generalize, and non simulated sleep position data is scarce: our 600 hours of video contain only around 1000 significantly different sleep positions. This means that we were forced to use a very small CNN model.

Our CNN classifies a single 40x26 cell BAM into one of four possible classes: empty, right, supine, and left. We use three convolutional layers with ReLU, 2x2 max pooling, and batch normalization, followed by two fully connected layers and an softmax layer (see Fig. 7.12).

As usual, training data is normalized to a zero mean and unit standard deviation. It is then resampled to balance the amount of samples on each class, and finally data augmentation is used to reduce overfitting. This includes left and right shifting of the bed as well as mirroring across the vertical axis.

Training is performed using Stochastic Gradient Descent (SGD) with a learning rate of 0.001 and a mini-batch size of 10. After each mini-batch update step t, we anneal the learning rate by a factor of $0.001/(1 + t \cdot 10^{-5})$. To compute the loss between the predicted and the target output, we use the negative log-likelihood criterion. Training stops when validation error ceases to improve, between 30 and 61 epochs.



(a) Empty (b) Right

(c) Supine (d) Left

Figure 7.11.: Sample BAMs. The dataset is diverse with respect to patient appearance (weight, age, gender, corpulence) and sleep attitudes (*e.g.* sleeping with/without blanket). It contains all the common behaviors in a hospital bed, *e.g.* a nurse visible in the leftmost image in (c). The gray scale indicates depth.

We perform a 5-fold cross validation, making sure that each fold has a similar size, but comprises a different set of patients. Subsequently, we present our experimental results averaged over all folds.

Chest Sensor: The chest sensor used in the sleep laboratory, the SleepSense from S.L.P. Ltd., uses an accelerometer to detect its orientation by means of detecting the gravity vector. Its output is discrete: left, right, supine, prone or up.

The up label means that the patient chest is in vertical position, either sitting on the bed or standing. We associate this label to our empty class.

Among all sleep positions, the sensor shows a significant bias towards the supine position (see Table. 7.10).

Accuracy across all samples is 91.9%, however it must be noted that the sensor tends to oscillate frequently between sleep positions, and often missreports the prone position. It is difficult to estimate the impact of those errors on the accuracy.

	Empty	Right	Supine	Left
Empty	98.61	0.00	0.69	0.69
Right	1.36	93.49	6.24	0.44
Supine	0.00	1.01	96.98	2.00
Left	0.86	0.45	13.24	85.45

Table 7.10.: Confusion matrix for the chest sensor from the sleep laboratory. The accuracy is 91.9%.

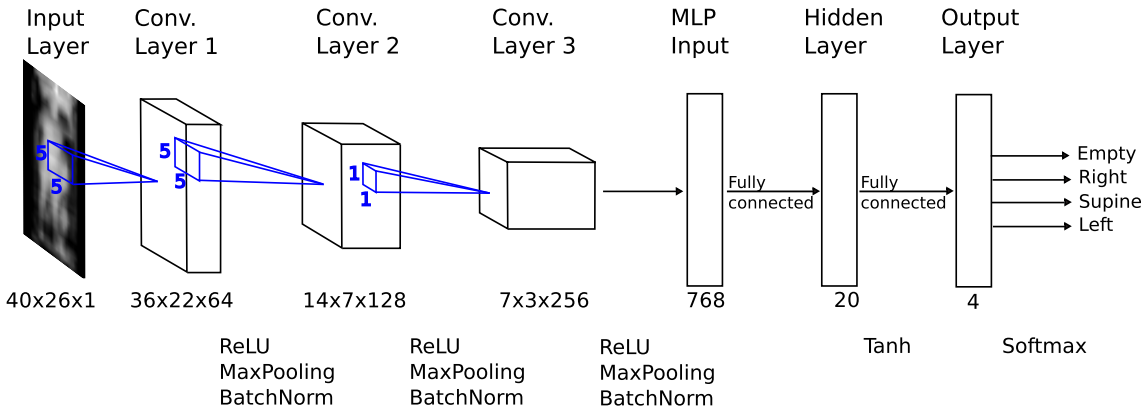


Figure 7.12.: The CNN architecture we use for sleep position recognition consists of three convolutional layers and two fully connected layers. Convolutional layers use ReLU activations, 2×2-max-pooling and batch normalization. We use *tahn* activations for the fully connected layers. The output layer uses softmax with one neuron per class.

Large Margin Nearest Neighbor: LMNNs had great success to classify sleep position from BAMs on the simulated THX dataset. The simulated dataset used was perfectly balanced with one instance per class and person, and had clearly distinct sleep positions.

When LMNN is used in the sleep laboratory dataset, results are significantly worse (see Table. 7.11). In this case we used leave-one-patient-out evaluation which is better suited to LMNN, and no rebalancing of the training set, as having repeated samples is harmful. LMNN is unable to overcome the unbalanced training set and shows a strong bias towards the most common class: supine. Overall accuracy drops to 70.8%.

	Empty	Right	Supine	Left
Empty	9.43	11.32	75.47	3.77
Right	0.00	55.88	35.43	8.69
Supine	0.15	5.54	85.83	8.47
Left	0.00	4.42	30.27	65.31

Table 7.11.: Confusion matrix of the Large Margin Nearest Neighbor, it shows a strong bias towards the supine position and an average accuracy of 70.8%.

Multilayer Perceptron: To quantize the impact of the convolutional layers, we evaluated a simpler neural network approach consisting of a Multi-Layer Perceptron (MLP).

The source BAM is flattened to a 1040 dimensional vector and fed to a MLP with 1040 input, 20 hidden and 4 output neurons. Hyperbolic tangent is used as activation function for the hidden neurons and softmax for the output layer, see Fig. 7.13.

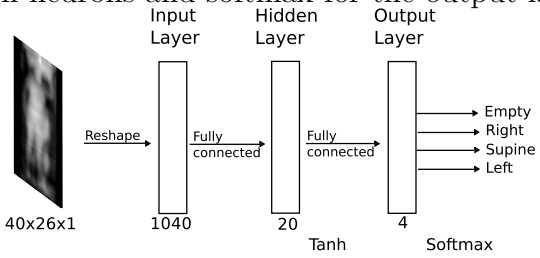


Figure 7.13.: The architecture of the MLP with 1040 inputs and 20 hidden neurons. Each 40x26 BAM is reshaped into a 1-dimensional vector before being processed.

Training is performed similarly to the CNN approach, balancing the training set and using data augmentation. We use dropout to prevent overfitting.

As expected, our MLP generalizes better than LMNN, but still shows weak performance with an accuracy of 82.2%, as seen in Table. 7.12).

	Empty	Right	Supine	Left
Empty	82.80	1.60	9.20	6.40
Right	1.60	84.40	8.40	5.60
Supine	1.60	10.40	78.80	9.20
Left	1.20	4.40	9.60	84.80

Table 7.12.: Confusion matrix of the multilayer perceptron, accuracy of 82.2%.

Histogram of Oriented Gradients and Support Vector Machines (SVM): HoG, from (Dalal and Triggs, 2005), and SVM, from (Hearst et al., 1998), are known to have a very good synergy to detect and classify human shapes.

It can be compared to a shallow version of a CNN, HoG providing the spatial structure, and SVM classifying on top. Furthermore, it has been suggested recently for sleep position by (Torres et al., 2016).

Our setup differs significantly from the one used by (Torres et al., 2016), and therefore it is not possible to exactly replicate the methods they implemented, but we can evaluate their suggested HoG+SVM on top of BAMs.

We apply HoG to the BAMs, reducing the feature space from 1040 to 280 dimensions, and apply SVM on top.

As the previous methods, data augmentation and training set balancing was applied, and empirically we found that the RBF-kernel provides the best performance.

The accuracy achieved is 88.9%, as seen in Table. 7.13). This is better our MLP, as expected due to the HoG descriptor provides structural insight, but not as accurate as the chest sensor.

	Empty	Right	Supine	Left
Empty	96.00	1.20	0.40	2.40
Right	1.60	89.20	6.40	2.80
Supine	0.80	2.80	90.80	5.60
Left	0.40	1.60	12.80	85.20

Table 7.13.: Confusion Matrix of the Histogram of Oriented Gradients and Support Vector Machines. The accuracy is 88.9%.

Convolutional Neural Network: Our CNN approach achieves an accuracy of 91.0% without data augmentation, whereas additional data augmentation leads to a further performance boost to 94.0%, as seen in Table. 7.14).

The CNN achieves superior performance compared to all other classifiers and even outperforms the chest sensor, and its behavior is similar to the HoG + SVM combination. In both cases the best classified class is the empty bed and, most importantly, the errors in sleep position are reasonable. This means that in both cases the left position is more easily confused with the supine position than with the right position, and the same happens in the opposite direction. With such accurate performance, most errors can be attributed to edge cases were the patient sleep position is between both positions.

	Empty	Right	Supine	Left
Empty	98.40	0.00	0.00	1.60
Right	0.40	93.20	4.40	2.00
Supine	0.40	1.60	94.00	4.00
Left	0.00	1.20	4.80	94.00

Table 7.14.: Confusion Matrix of the CNN. The accuracy is 94.0%. This outperforms all other classifiers and the chest sensor.

Therefore, we can conclude that our computer vision sensor is capable to recognize sleep position up to an accuracy comparable to state-of-the-art contact sensors. To improve the results, we should capture even more data in order to use a larger CNN model, and probably we need to tackle the problem as a regression problem instead of a classification problem, in order to better integrate the cases where the patient lies in a position that is between two canonical classes.

8. Conclusion

An automated and reliable sleep monitoring system would improve the sleep quality of elderly people, simplify the diagnose of sleep disorders, and monitor the treatment of already diagnosed patients with sleep disorders. Of the many modalities of sleep monitoring, computer vision has the advantage of being relatively inexpensive, easy to install, versatile (*i.e.*, can be used for multiple different tasks), and requires absolutely no interaction from the user, which is particularly important for elderly patients.

Most state-of-the-art however, does not consider sleep monitoring as a holistic task, and focuses on just one of the related subtasks (*e.g.*, breath rate recognition). Furthermore, most published approaches in this topic have weaknesses related to the scenario (*e.g.*, people sleeping without a blanket), problem definition (*e.g.*, considering apnea simply as a breathing interruption), or evaluation (*e.g.*, testing only on a few healthy volunteers).

In this thesis we have presented, to the best of our knowledge, the first system capable of addressing most of the tasks related to sleep monitoring. Furthermore, our training and evaluation is performed on large datasets collected in real world scenarios, and the tasks we addressed were defined in close collaboration with sleep medicine doctors. Therefore we expect that the research performed in this thesis has proven that computer vision can be used to analyze sleep patterns in most, if not all, the required scenarios.

We developed a customized recording device, collected large amounts of data in real world scenarios, obtained reference data for our collected data from doctors experts in sleep medicine, and applied state-of-the-art signal processing and machine learning techniques. We have produced several novel contributions on each of the steps of the pipeline.

Contributions

MRD. Our recording device, named Medical Recording Device (MRD), collects information from several sensors, compresses, transmits and stores it efficiently and reliably. Our MRDs have processed reliably more than one Petabyte of data, and thanks to them we can capture as much sleep data as we need in future projects. Furthermore, the technologies that we developed for the MRD allow us to create easily new monitoring systems for other low information density applications, like monitoring the activity of the residents in nursery homes. Also, our efforts on the MRD have derived in contributions that have had an impact in other fields: our analysis of the Kinect has been used to extend the range of the device (*e.g.*, Berger et al. 2014), and Marlin, our compression algorithm, is the first algorithm capable of compressing lossless images efficiently and decode them faster than 2 GiB/s.

Datasets. Our MRD allowed us to collect, to the best of our knowledge, the largest and most complete computer vision datasets for video-based sleep monitoring used in public research. In machine learning, size matters, and our datasets have allowed us to develop deep learning models for sleep monitoring tasks.

Bed Aligned Maps. We developed a new compact descriptor named Bed Aligned Map (BAM) to summarize sleep recordings, and used it as the input representation in most of our algorithms. Compared to images, BAMs are scale, rotation and translation invariant. Also, BAMs are 500 times smaller than the original depth maps, hence, using BAMs for both training and testing our machine learning algorithms are very efficient, and this was crucial in this thesis given the amount of video information we collected (in the order of Terabytes).

Respiration Analysis. We proposed several new algorithms to analyze respiration cues that improve the state-of-the-art. We provide a novel mathematical background that explains how the camera parameters affect the signal-to-noise ratio for the chest movement. We provide algorithms to estimate the respiration rate for two different input modalities. Compared to a sensor that measures chest expansion, our approach obtains similar results. Finally, we present a preliminary study on the viability of detecting sleep apnea events, which is a challenging task to perform automatically.

Sleep Quality Cues. We also proposed new algorithms to analyze different sleep quality cues. We suggested objective and view independent metrics to monitor bed occupancy and agitation, and use them to create visual sleep summaries. We developed a deep learning approach to estimate sleep position that outperforms the sensor used in sleep laboratories. To the best of our knowledge, we have been the first to propose computer vision algorithms that recognize sleep stages, restless legs syndrome events, and classify bed-related actions.

Discussion and Future Directions

Automated sleep monitoring is becoming a mature machine learning problem. Until now, sleep monitoring has been approached generally as a sensing problem, where the main challenge was to design relevant sensors, and those were evaluated in oversimplified scenarios. This scenario oversimplification has hindered the actual deployment of sleep monitoring devices in real world scenarios, simply because they were not reliable enough. We show that given enough context information, it is possible to analyze sleep monitoring very effectively from the machine learning perspective.

We have analyzed many different tasks related to sleep monitoring, but there are still many challenges that our medical and health care partners have identified and have not yet been addressed, like detecting nightmares, panic attacks, falling events, wetting the bed, smoking, fire hazards, etc. New datasets are required to analyze some of these tasks.

Complementary, in our datasets there are sources of information that have not yet been tapped (*e.g.*, snoring, facial expressions, etc.). Also our large data collection from EHS has not yet been entirely analyzed and can spawn new developments in summarization and analysis of long term sleep patterns.

Also, there is room to improve the performance of many of the most challenging tasks we attempted, (*e.g.*, predicting sleep apnea events), and there are many techniques that still need to be explored, like learning personalized models.

Finally, we are exploring ways to embed many different techniques into a single machine learning model, the goal behind this idea is that a holistic expert system will generalize better, improving the performance of the individual tasks, and it should deliver new, better indicators that can be used to improve our sleep quality.

A. Analysis of the Kinect Depth Camera

Depth cameras are ideal to monitor sleep, as they use near infrared (NIR) light, and thus, are able to work in the dark. But the depth camera technology is has not reached the same level of popularity that of the color cameras, and this is reflected in the very reduced number of choices available for a sensor. In fact, depth cameras were deemed too expensive to be used in non-industrial environments before the release of Microsoft Kinect in 2010.

The Kinect camera suddenly enabled depth sensing at a reasonable cost, however, this was not its primary purpose. It was meant to be used in conjunction with the XBox 360 gaming console from Microsoft as a means to capture the skeleton of the players in real time.

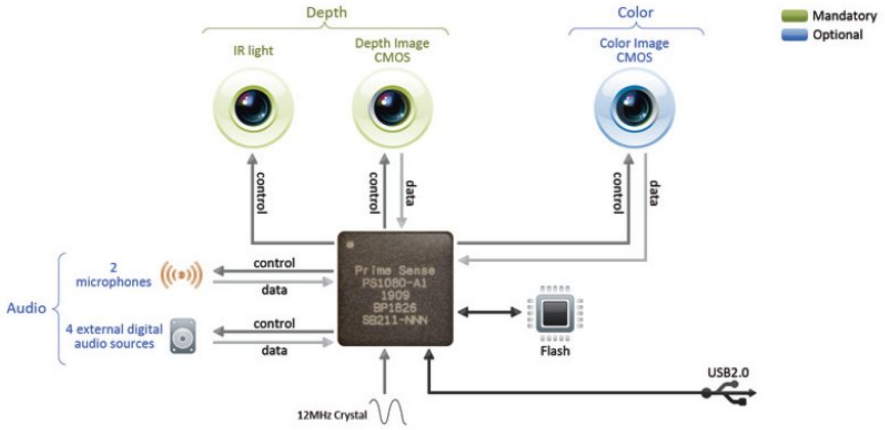
When it was released, the Kinect camera was a closed device with no means to be accessed from a computer. This condition changed within hours of the release when the communication protocol was reverse engineered by Hector Martin, leading to the development of the first usable driver: *freenect*[1]. The protocol revealed that Kinect solves internally the correspondence problem and directly provides the disparity map, which is very convenient.

This made the Kinect the standard depth camera used for most computer vision tasks. Although other depth cameras have appeared after Kinect, (most importantly, the Kinect v2), the original Kinect v1 remains one of the most popular cameras still in use, as it balances compactness, ease of use, and precision.

Being a closed design, the actual specifications of the camera were largely unknown. Several research groups analyzed the camera in many different ways. First, its precision as a range finding device was analyzed (see Khoshelham and Elberink 2012, Park et al. 2012, and particularly Andersen et al. 2012), where its precision was found to be adequate, but its repeatability is higher. The results also revealed a drift related to its operating temperature.

Those early studies only analyzed the depth map provided by Kinect, unbeknownst of the technology used. On the other hand, Kurt Konolige from WillowGarage published a very detailed technical analysis of the Kinect based on its protocol and the technology it uses (Konolige and Mihelich, 2010), which is similar to the projected texture approach he used previously in the PR2 robot (Konolige, 2010). Although

[1] http://openkinect.org



Source: `www.primesense.com/category/reference_design`

Figure A.1.: Diagram of the PS1080 chip that powers Microsoft Kinect. The PS1080 can manages both depth and color cameras. The depth information is captured by the NIR camera, while the NIR light is used to project a fixed, pattern.

Konolige's measured the focal length and the lens distortion, the algorithmic details remained at the theoretical level, as the calibration values used in Kinect were not available.

In this chapter, we build upon the Konolige's analysis, and develop a model that can accurately replicate the performance of Kinect in real and simulated settings. Our model is based in an algorithm that reconstructs the calibration information encoded within each Kinect. This model can be used, for example, to replace the depth algorithm hardcoded in Kinect, and use a software algorithm with higher resolution, as has been done in (Berger et al., 2014).

Our model for Kinect revealed that the combination of the algorithm used for Block Matching (BM) and the projected pattern produces a predictable bias in the result. This bias depends on the distribution of the dots within the block. We quantize this bias, and suggest ways to correct it.

In the end, we are able to model the Kinect camera, which was optimized for cost, not for precision, and suggest alternative algorithms that improve the quality of its depth maps.

A.1. The PS1080 Chip from PrimeSense

The Kinect system contains three independent devices, a 4-microphone array, a tilt mechanism, and the depth camera itself. The depth camera can be used independently from the other devices, and it was developed by a company named PrimeSense.

The camera hardware is very efficient, but the easiest way to explain its workings is to start with a normal stereo camera. In stereo cameras, we have two image sensors with a fixed position one to another, we name one of the cameras the master and the other as the slave.

If we want to know the depth value of a pixel from the master camera, we need to search for the corresponding pixel on the slave camera, and then triangulate its position. This is known as the correspondence problem, and it is not easy to solve.

One way to make the correspondence problem trivial, is to project a texture to the image (the projected texture approach). This creates a rich set of unique characteristics which simplify enormously the problem of finding correspondences between the cameras.

Kinect goes one step further by realizing that if a projector and a camera were on the same place, the camera would see exactly the texture that the projector is showing. In other words, the camera becomes unnecessary. Thus this is what Kinect does: it is a stereo camera system where the slave camera has been replaced by a inexpensive projector Thus its bill of materials is not significantly larger than a webcam.

The Kinect is powered by a chip named PS1080 whose reference design can be seen in Fig. A.1. This chip also powers other camera systems (namely ASUS Wavi Xtation) which are almost 100% compatible with Kinect. The other key components are an NIR laser at 830nm and a normal monochrome camera with a 10nm bandpass tuned to 830nm. The PS1080 receives the NIR information from the monochrome camera and applies BM with a reference image (the virtual view of the slave camera), which can be done efficiently in parallel. The disparity map is then transmitted by USB2.0 using a very simple protocol.

A.2. Our PS1080 Model

To create our suggested PS1080 model, we were inspired by previous implementations of BM algorithms in FPGAs (Strother, 2011). Then we incorporated the knowledge provided by previous analysis of Kinect, mainly from Konolige's technical analysis (Konolige and Mihelich, 2010). Furthermore we know that Kinect is designed for low cost and depth processing is done in hardware. This implies a certain amount of constrains (*e.g.*, it is unpractical to store the whole image within the chip, and there is no external memory).

Thus we based our model on the following premises:

- NIR view from the camera is not rectified, as it is expensive to do it in hardware. A carefully crafted low-distortion lens is used instead.
- BM Sum of Absolute Differences algorithm solves the correspondence problem in a very cost efficient way.

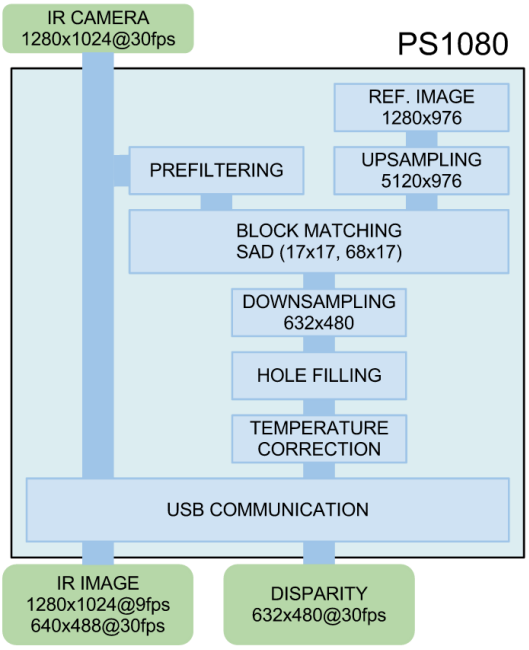


Figure A.2.: Our suggested architecture to emulate PS1080. The system acts as a stereo camera where one of the sensors has been replaced by a fixed pattern stored in the flash. Using a fixed pattern can create large shift in albedo which is compensated by a pre-filtering step, and a post-filtering step to compensate temperature changes.

- To eliminate memory constrains, epipolar lines are approximated by the horizontal scanlines.

We construct our model accordingly, and after a few refinement steps, we end up with the block diagram shown in Fig. A.2.

A.2.1. Spatial Resolution

The NIR sensor used in Kinect has a native resolution of 1280x1024 pixels (Super Extended Graphics Array (SXGA)). From several sources we know that a reference image exists and its line size is also 1280 pixels. Moreover, the speed of the NIR Image sensor is coherent with 1280x1024@30fps.

Kinect outputs disparity maps at 640x480 pixels (Video Graphics Array (VGA)), the result of a 2-to-1 subsampling. However, 8 pixels per line are zero, therefore the actual resolution is 632x480. This is the result of the BM algorithm, which requires some padding. If we assume a square block, and the padding is 16 bits at SXGA, then we assume that the original SXGA is cropped to 1280x976 pixels before entering the depth pipeline.

A.2.2. Depth Resolution

Kinect's VGA depth maps have 3 bits of subpixel precision, or, equivalently, 2 bit subpixel precision at SXGA. Usually subpixel precision is calculated in software using quadratic interpolation, but this is expensive to do in hardware. However, BM is so hardware efficient that it is worth simply to upsample the SXGA image 4 times and perform standard BM per pixel. We evaluated both approaches and the Kinect results are consistent with upsampling.

A.2.3. Reference Image

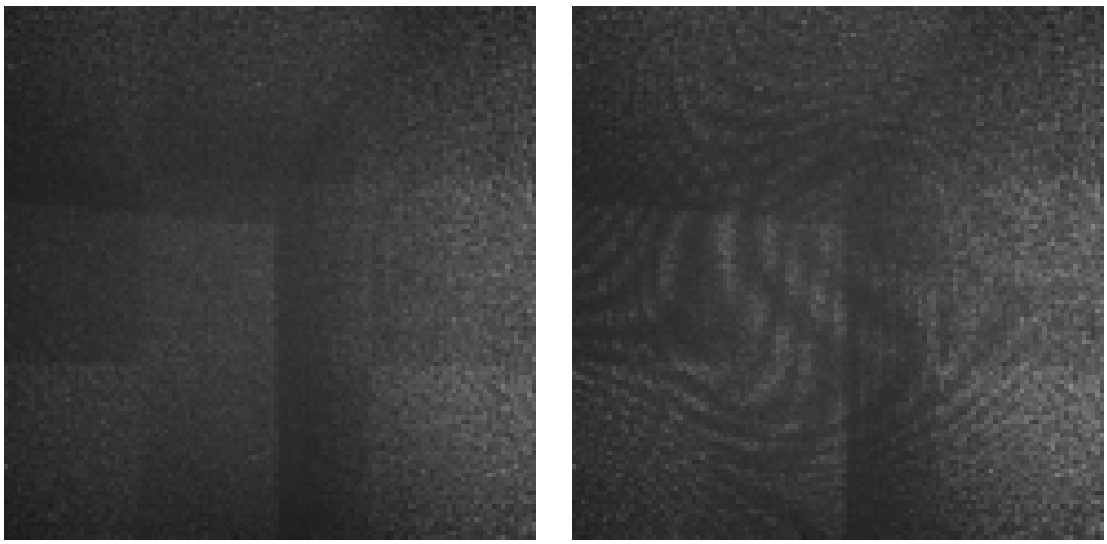


Figure A.3.: The projected NIR pattern from Kinect as seen from the NIR camera. Left: sensor is configured at SXGA. Right: sensor is configured at VGA, and a moire pattern is present.

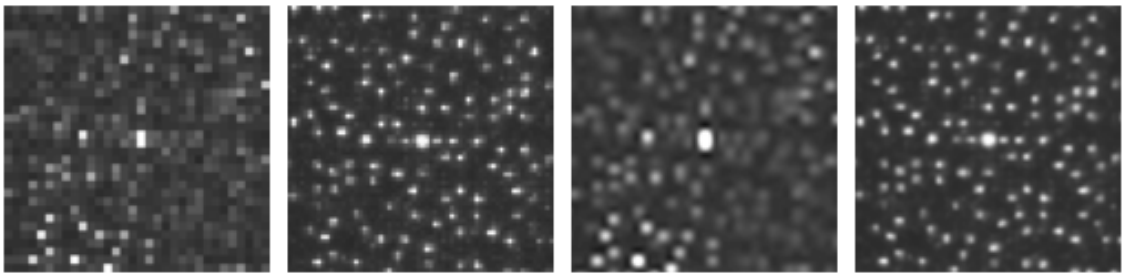


Figure A.4.: Detail of the reference image recovered from different sources. From left to right: VGA, SXGA, VGA+superresolution, SXGA+superresolution.

Te projector used in Kinect is conceptually simple, based on an idea in (Aoki et al., 2001). A temperature stabilized NIR laser goes through a first diffraction grating where it is divided in a 3x3 beams. Then a second diffraction grating splits each beam to form a pattern of 211x165 uncorrelated quasi-periodic dots. The pattern is

the same for all Kinects. The density of the dot pattern is enough to create a moiré pattern when recorded at VGA resolution (see Fig. A.3).

To recover the pattern from the point of view of the projector, we reverse the stereo camera model:

$$D(x,y) = \arg\min_k |I(x+k,y) - R(x,y)| \tag{A.1}$$

turns into into:

$$R'(x,y) = I(x + D(x,y), y) \ . \tag{A.2}$$

Therefore, if we obtain simultaneously the NIR image I and the disparity field D, we can obtain an approximation of the reference image R' by displacing the pixels from I the amount noted by D.

In order for this to work, we assume that, for the optimal k, $I(x+k,y) \approx R(x,y)$. A more detailed model would assume:

$$I(x+k,y) \approx A(x+k,y)\ R(x,y) + S(x+k,y) \ , \tag{A.3}$$

where A is the albedo of the scene, and S is the actual image the sensor would capture without the projected pattern.

However thanks to the bandpass NIR filter, scene information unrelated to the reference pattern R is filtered out so we can assume $S \approx 0$.

By averaging a large amount of captures from different orientations, we assume that all pixels have been exposed to a similar range of albedos, and we can assume $A \approx 1$.

To improve the quality of the recovered pattern, we capture many images and fuse them using superresolution techniques (see Fig. A.4).

A.2.4. Prefiltering

In stereo camera systems, gain and exposure settings are synchronized between the cameras, and therefore the illumination from both images is comparable. This is not the case of Kinect, where the reference image does not depend on the actual environment, the brightness is not consistent between views. It is known that this problem is addressed by a prefiltering step, but the actual filter used is not known.

The common way to address this problem is by normalizing both blocks before matching. We call it the *normalizing* filter, but in hardware it might be more efficient to use an horizontal Sobel filter as seen in (Hirschmuller and Gehrig, 2009).

A.2.5. Hole Filling and Temperature Correction

Kinect implements two post-processing steps to the depth map. First a hole filling pass fills small regions where the BM algorithm fails to provide a reliable output.

Second, a temperature correction seems to be applied after the hole filling step, and allows the Kinect to be used before it reaches its optimal working temperature. This effect was noted by (Andersen et al., 2012) and explored in depth in (Fiedler and Müller, 2013). As it needs no further analysis, all our experiments were performed when the Kinect temperature had stabilized (1 hour).

A.3. Model Tuning

We tune the model by comparing the disparities provided by our model and Kinect. As evaluation metrics, we use mean error, mean bias and variance, all in VGA pixels. Unless stated otherwise, we use a 17x17 block size (in SXGA pixels), the SXGA pattern,the SXGA source image, and the normalization as a prefiltering step.

block size	mean error	mean bias	variance
13x13	0.0337	-0.001	0.0047
15x15	0.0219	0.0025	0.0028
17x17	0.0201	0.0066	0.0025
19x19	0.0271	0.0089	0.0037

Table A.1.: We evaluate different block sizes for the BM algorithms. We compare our disparity to Kinect's, and units are in VGA pixels. We see that in all cases our error is significantly smaller than the resolution (0.125 pixels).

We evaluate different block sizes in Table. A.1. All block sizes perform offer compelling performance, with errors well below the resolution limit of 0.125 pixels. Kinect may use any block size between 15x15 and 17x17, but for our model we choose 17x17 as it has the lowest error and variance.

prefiltering	mean error	mean bias	variance
unfiltered	7.6075	7.2096	213.157
normalized	0.0201	0.0066	0.0025
Sobel	0.0201	0.0059	0.0026

Table A.2.: We evaluate different options as a prefiltering step. Again, we compare our disparity to Kinect's, and units are in VGA pixels. Both normalization and Sobel have almost the same performance.

On the filter evaluation, the unfiltered system performed the worst, while Normalized and Sobel filters provide similar results (see Table. A.2). Those results agree with the findings in (Hirschmuller and Gehrig, 2009).

Finally, we measured the performance impact of using VGA resolution as an input instead of SXGA. This is interesting in real time applications as VGA can be obtained at 30fps from the Kinect camera, while SXGA can only be obtained at 9fps.

Ref. Image	mean error	mean bias	variance
SXGA	0.0201	0.0066	0.0025
VGA	0.2402	-0.24	0.0054

Table A.3.: We evaluate the performance of our model when using a VGA image as an input instead of a SXGA. Although there is a significant drop in performance, the errors are still below 1 pixel.

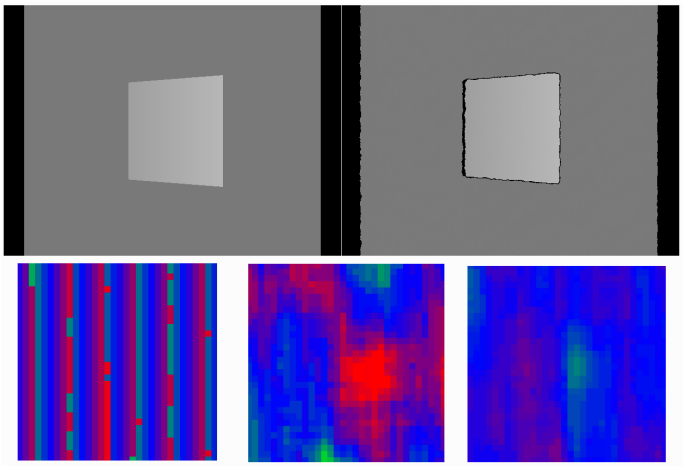


Figure A.5.: We developed a simulator that allows us to evaluate Kinect's algorithms in artificial scenarios. Top left: Depth ground truth. Top right: BM depth estimation. Bottom row: accuracy of different BM approaches on the central 32x32 pixels. Red represents error by excess, green represents error by defect with blue being neutral. From left to right: Kinect's algorithm, OpenCV's algorithm, OpenCV's at 4x resolution.

Our experiments show that he performance impact is severe, and we observe a mean error around 12 times larger on VGA than on SXGA, although it is still well below 1 pixel.

A.4. Kinect Simulator

We developed a Kinect simulator based on previous model to analyze the properties of the suggested depth algorithms. Given a triangle mesh based scene definition, we use raytracing to simulate the NIR view that Kinect would perceive. The simulator provides ground truth of the depth, and simulates the BM algorithm performed by Kinect (see Fig. A.5).

From the OpenNI driver, we know that the reference image is a calibration image taken at $p = 1200mm$ from the camera with a 100 pixel bias. Knowing the focal

length $fx = 580$ (at VGA resolution) and the baseline $b = 75mm$. And the disparity is provided at a resolution of 8x of the VGA output, the depth d can be calculated as:

$$d = b * \frac{fx}{100 + b * fx/p - (1/8) * disp} \ . \tag{A.4}$$

A.5. Analysis of the Dot Pattern

We use a SXGA version of the reference image to localize and annotate each individual dot in the calibration image with a great degree of precision.

Due to a strong vignetting effect, Kinect chooses a brightness setting that causes saturation on the dots close to the center of the image. It is not possible to alter the brightness setting, so we apply a Gaussian filtering with $\sigma = 3px$ to allow us to find dots as local maxima. A simple threshold filter is used to eliminate noise and small sparkles that appear as side effects of the holographic grating. Finally we use the 3x3 neighborhood around the maxima to estimate its peak with subpixel precision. A total of 28117 dots are detected.

Assuming a block size of 17x17 pixels, we analyze the density of the dots/block, the block bias, and the resolution (see Fig. A.7).

A.5.1. Dot Density

We measure the density of the dots by counting, for each pixel on the output image, how many dots would be included in its 17x17 block. The number of dots per block varies between 1 and 14. It has a mean of 7.62±1.23 dots per block on the center of the pattern and drops to 3.07±1.18 dots per block at the edges.

A.5.2. Block Bias

We define the block bias as the mean position of the dots included within a block with respect to the its center. This measure provides us with an indication of the precision of the Kinect spatial information. If we approximate the local neighborhood of a pixel as a flat surface, and the output of the BM algorithm as the mean disparity of each individual dot in the block, then the distance reported will be that of the mean position the dots within the block, which may not be the actual center of the block (*i.e.*, the pixel we are testing). This is the source of the non-sharp edges present in the depth field (see Fig. A.7). The block bias at the center of the pattern is 0.64 ±0.34 pixels and 1.60 ±1.02 pixels at the edges. In fact, less than 40% of the pixels in the center of the pattern present a bias smaller than 0.5 pixels, and this number drops to less than 10% at the edges. This means that the majority of the pixels report a disparity value that would fit better a neighboring pixel.

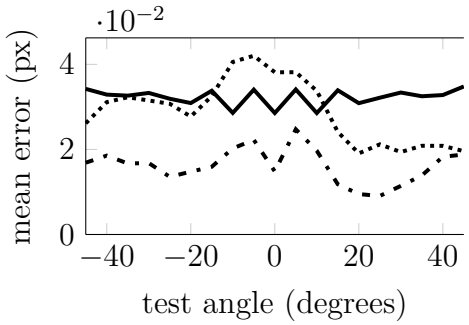


Figure A.6.: Mean disparity error on the central part of a tilted plane in pixels. Solid: Kinect's algorithm. Dotted: BM at SXGA resolution. Dash-dotted: BM at QSXGA resolution.

A.5.3. Resolution

We analyze the minimum radius that a circular feature needs to have for the BM algorithm to detect it.

First we test an ideal scenario: we assume that we have an oracle that allows us to perfectly solve the correspondence problem and, in consequence, our circular test object only requires to be be illuminated by a single dot of the pattern to be detected. For each pixel, the distance to its closest dot will indicate the minimum detectable radius, and ranges from 1.18 $\pm$0.50 to 1.98 $\pm$0.89 pixels.

However, due to the BM, an object needs to be covered by more than half of the dots of a block to be detected. In this case the minimum detectable radius for a circular object is fairly constant, and ranges from 3.24 $\pm$0.44 to 3.38 $\pm$0.75 pixelss.

A.6. Improved Block Matching Algorithms

Kinect's BM algorithm uses a fixed point resolution of 1/8th of a VGA pixel.

As the NIR image can be captured at a SXGA, we have evaluated floating point BM algorithms directly applied to the higher resolution NIR image expecting better depth resolution.

We have evaluated Konolige's optimized BM implementations from OpenCV (Bradski, 2000), first on SXGA resolution and later on QSXGA (2x2 SXGA).

SXGA takes 220ms to process an image and obtains a similar performance that of Kinect's algorithm, while BM on QSXGA takes more than 12 seconds to process a single image although obtains a significantly better performance (see Fig. A.6).

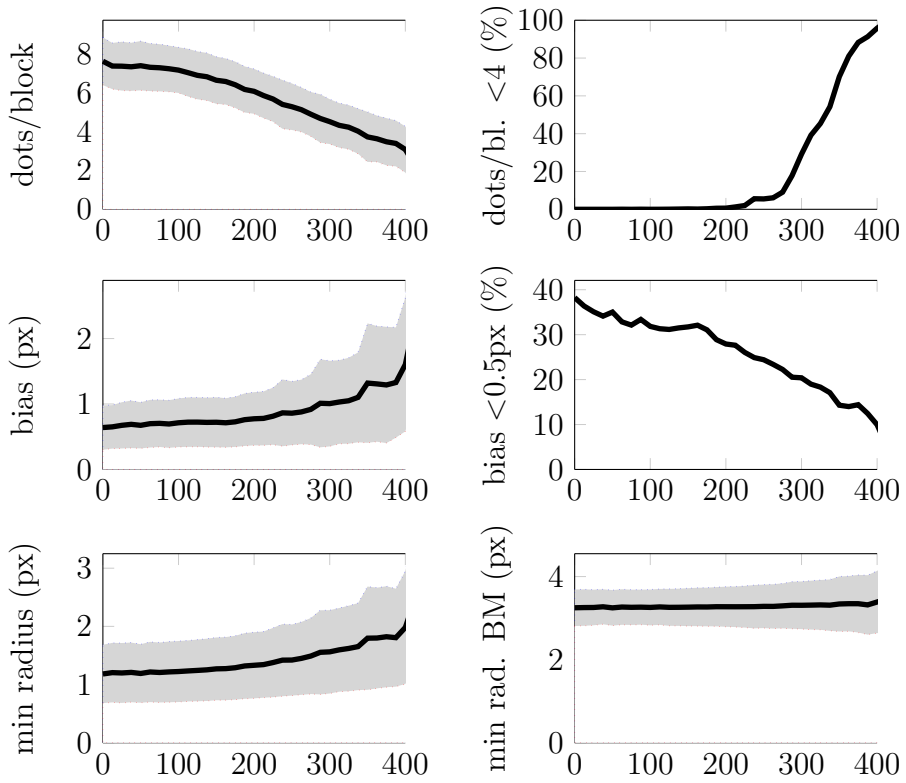


Figure A.7.: Analysis of the pattern. X axis represents the distance in pixels from the central point of the pattern. The gray corridor indicates the standard deviation. Block indicates the 8x8 neighborhood used in the BM algorithm. Top left: density of the dots per block. Bottom left: proportion of blocks that are represented by less than 4 dots. Top middle: spatial bias of the block. Bottom middle: proportion of blocks with a bias smaller than 0.5 pixels. Top right: minimum radius that a circular object must have to be illuminated by the pattern. Bottom right: minimum radius that a circular object must have to be detected by the BM algorithm.

A.7. Dot-based Point Cloud

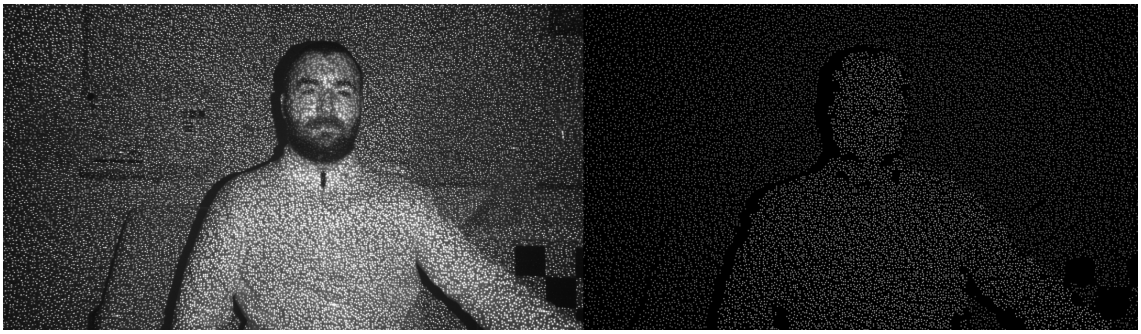


Figure A.8.: Our dot-based algorithm provides a sparse 3D point cloud (right) from the NIR image (left). Each dot in the projected pattern produces a 3D point.

We have already discussed how Kinect can only obtain depth information from the dots of the projected pattern. Ideally, a system that assigned a disparity value to each visible dot would be optimal, however the problem is hard as the dots are indistinguishable between them.

We suggest to use a simple BM algorithm to solve the correspondence problem. This provides us a gross estimate of the pattern dots' position within the NIR image, which will be refined.

As a prerequisite, we localize with precision the dots from the reference image and create a table that lists, for each pixel of the reference image, all the dots in its 8x8 neighborhood.

The online algorithm is as follows:

1. Apply BM between the NIR and the reference image.

2. For each NIR pixel (i, j) with disparity d, calculate its counterpart on the reference image $R'(i, j + d)$, and add d as a possible disparity to all pattern dots in its 8x8 neighborhood.

3. For each dot in the pattern, cluster all disparity candidates (mean-shift $\sigma = 1$). Each disparity candidate is projected back to the NIR image and the brightest one is selected.

Unlike Kinect's, this algorithm does not output a regular grid. Instead it provides a 3D point cloud where the z coordinate is calculated from the disparity between a known dot in the pattern and its localization in the NIR image. Then the x and y coordinates are calculated from the localization of the dot in the NIR image and its depth.

We have evaluated this algorithm on synthetic and realistic scenarios (see Fig. A.8) and found a performance of around 1.75 frames per second on an Intel i5 760 @ 2.80GHz. In our C++ implementation, the BM has a fixed cost of 290ms, pixel

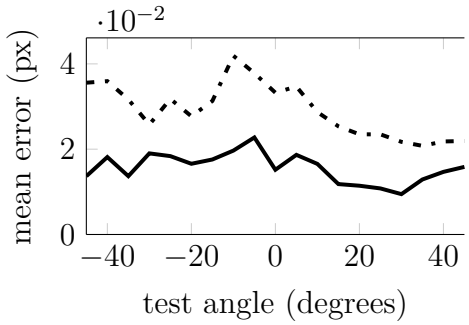


Figure A.9.: Mean disparity error on the central part of a tilted plane. Solid: our proposed algorithm. Dotted: BM at SXGA.

processing takes 50ms, mean-shift clustering 220ms and the final sub-pixel estimation 100ms. However the algorithm has ample margin for parallelization and could be implemented in a Graphical Processing Unit (GPU) for increased performance.

This algorithm achieves better depth precision than BM algorithms (see Fig. A.9).

A.8. Discussion

We have analyzed Kinect's BM algorithm quantifying important characteristics such as the minimum size of a detectable feature, and the spatial bias induced by the unbalanced distribution of dots within a block. Using this analysis we propose a better BM algorithm that improves depth precision. However, BM algorithms provide blurred depth maps and do not compensate for the spatial bias. Therefore, we have proposed an efficient method to provide a sparse point cloud by estimating the disparity of each individual dot in the projected pattern. This method avoids unnecessary interpolation, therefore it provides an unbiased point cloud with higher spatial resolution than the original algorithm using only one tenth of the 3d points.

B. Marlin: a High Throughput Entropy Compression Algorithm

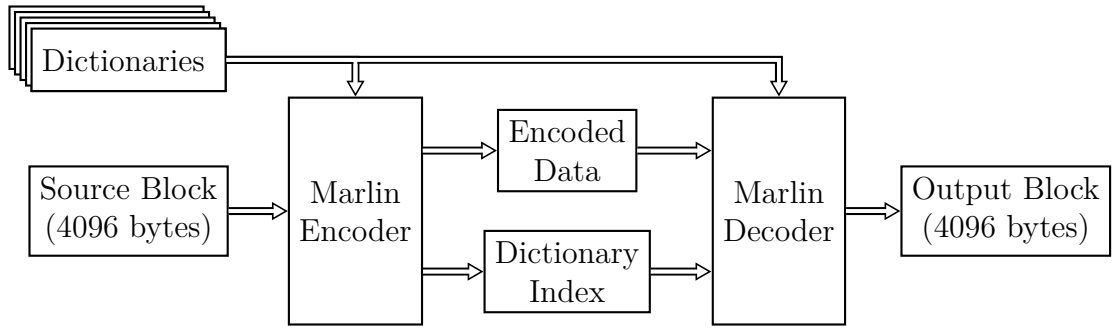


Figure B.1.: Marlin encodes blocks of 4096 bytes. A dictionary set is known to both encoder and decoder. The encoder inspects the block, selects the best dictionary, compresses the block, and emits the compressed blob and the index of the dictionary. The decoder recovers the original block using the selected dictionary.

During the development of our Medical Recording Device (MRD), we developed several new compression schemes designed to deal with the large amount of raw data we collected. Of these, we want to highlight Marlin, an entropy codec optimized for High Throughput (HT) applications. In typical workloads, Marlin is 4 times faster than current state-of-the-art entropy codecs.

The goal of HT compression is not to reduce storage requirements, but to better leverage the available bandwidth. In this scenario, the maximum throughput is a balance of compression ratio and coding/decoding speed. A number of algorithms exploit different sweet spots, depending on the required application.

All current HT algorithms are derived from LZ77 (Ziv and Lempel, 1977) using hash tables (Harnik et al., 2014; Williams, 1991) for encoding, and memory copies for decoding, *e.g.*, Snappy (Tarantov and Gunderson, 2011), LZ4 (Collet, 2011), or LZO (Oberhumer, 1996). The entropy stage is either dropped, *e.g.*, Snappy, LZ4, or greatly simplified, *e.g.*, LZO, Gipfeli (Lenhardt and Alakuijala, 2012), since adaptive entropy codes like Huffman (Huffman et al., 1952) and arithmetic (Rissanen, 1976)/range encoding (Martin and Martin, 1979) are generally slow, despite of the current efforts

to make them faster, *e.g.*, Huff0 (Collet, 2013b) for Huffman and FiniteStateEntropy (Collet, 2013a; Duda, 2009) for range encoding.

Dropping or simplifying the entropy coding stage is a reasonable trade-off for text based sources, but has a steep penalty mainly when compressing memoryless sources like noisy sensor data, *e.g.*, raw images (Howard and Vitter, 1993; de Vaan, 2007; Weinberger et al., 2000). As existing HT compression algorithms were not optimal for the kind of data that our MRD was recording, we developed Marlin, an entropy codec that can be decoded as fast as Snappy and LZO. Marlin follows a Variable to Fixed (VF) coding approach. VF codes are similar in essence to LZ77 and can be decoded equally fast. In VF coding, a memoryless digital source is described as a sequence of variable sized words from a given dictionary. The code emitted consists of the indexes that point to such words. The dictionaries are generally built to be uniquely parsable, which requires that no word is a prefix of any other word in the dictionary. Such dictionaries can be build using Tunstall (Tunstall, 1967) algorithm, but they tend to be large.

Plurally (*i.e.*, not uniquely) parsable dictionaries (Savari, 1999) have been shown to outperform Tunstall dictionaries of equal size (Al-Rababa'a and Dubé, 2015; Yamamoto and Yokoo, 2001; Yoshida and Kida, 2010). We suggest a novel way of constructing plurally parsable dictionaries for arbitrary alphabets using a constrained Markov process of only 255 states (for 8-bit sources). To optimize the dictionary to an input source, we apply an iterative algorithm with two steps: the first step calculates the Markov process for a given dictionary, the second step creates an optimized dictionary for a given Markov process. Our algorithm alternates between the two steps. Our proposed HT encoding/decoding technique, Marlin[1], builds upon this dictionary to implement an algorithm that is extremely fast to decode.

On a lossless image coding experiment, Marlin achieves a compression ratio of 1.94 at 2494MiB/s. This makes Marlin is as fast as state-of-the-art high-throughput codecs (*e.g.*, Snappy, 1.24 at 2643MiB/s), and almost as efficient as best entropy codecs (*e.g.*, FiniteStateEntropy, 2.06 at 523MiB/s). Therefore, Marlin enables efficient and HT encoding for memoryless sources, which was not possible until now.

B.1. Proposed Encoding/Decoding Technique

High throughput codes are rarely used to compress files, therefore Marlin, like LZ4 (Collet, 2011), Snappy (Tarantov and Gunderson, 2011), Huff0 (Collet, 2013b) and FSE (Collet, 2013a), does not define a file format. Marlin is designed to compress data blocks of a fixed size (see Fig. B.1). Each block is encoded using a dictionary chosen from a known set of dictionaries. This set should cover all expected probability

[1] We choose the name for the Black Marlin, the fastest fish known to man. Like fish, which are not known for having great memory, our algorithm does not remember the state of the input.

distributions that the source may present. The encoder generates a compressed data blob, and provides the index of the dictionary used to compress the block. The decoder, which knows the same dictionaries used for compression, receives the compressed data blob and the dictionary index, and generates the original uncompressed data.

B.1.1. Implementation Details

Block size. Small block sizes provide better compression efficiency, but large blocks improve decompression performance. We fix the block size to 4096 bytes, which corresponds to the size of a memory page in x86 processors.

Dictionary set. The dictionary building process is time consuming, and the generated dictionaries are too large to be sent with the data. This makes it unfeasible to create a custom dictionary per block (as Huff0 and FSE do). Marlin uses a fixed set of dictionaries with common probability distributions, and selects the best fitting one to compress each block. For efficiency reasons, the dictionary set must be small, as switching dictionaries trashes the cache. We employ 11 dictionaries per distribution.

Alphabet size. Marlin currently supports 8-bit alphabets.

Dictionary size. Marlin dictionaries can contain between 2^9 and 2^{16} entries. Larger dictionaries provide better compression efficiency, but are slower to decode. We use 2^{12} entries per dictionary as it fits within most L1 caches.

Special cases. Like Huff0 and FSE, Marlin implements two special cases. One for blocks that contain only zeros, and one for blocks which can not be compressed at all. Both cases are indicated by magic values in the dictionary index.

B.1.2. Encoding

The first step of the encoding process is to select a dictionary to encode the current block. A naive way to find the most efficient dictionary is to encode the block with all available dictionaries, and choose the one that generates the smaller blob, but this is very inefficient. Instead, we use the block entropy to select which dictionary to use.

Given a dictionary, the encoder needs to find the largest word in the dictionary that matches the input source. We have implemented this task efficiently using a trie (*i.e.*, also known as prefix tree). The performance is bound by memory latency, therefore using smaller dictionaries boost encoding speed. This happens because a larger proportion of the trie fits within the cache, increasing the hit ratio. The compression speed of our current implementation ranges between 100MB/s and 300MB/s, however there are still many tradeoffs to be explored in this area.

B.1.3. Decoding

The decoder receives a compressed blob representing a single data block, and the index of the dictionary used to compress it.

For each dictionary, the decoder prepares a table structure that contains a word per entry (see Fig. B.2.b). Each entry from the table has the same length, which is a power of two. The word is at the beginning of the entry, and the word length is placed on the last byte.

The decoder loop does:

1. extract a word index from the input stream,
2. fetch the corresponding entry from the decoding table,
3. copy the entire entry to the output stream,
4. increase the output pointer accordingly.

We analyze now in deeper detail the decoding process. In the first step we extract a word index from the input stream, as we are coding 8-bit streams, our dictionary needs to have more than 2^8 entries to achieve any compression at all. Actually, the larger the dictionary is, the better compression will be achieved, but we found empirically that there is little to gain for dictionaries larger than 2^{16} entries. Constraining the dictionary sizes to power of two values ensures that we can extract indexes from the input stream using only bit mangling operations. In our experiments we select 2^{12}-sized dictionaries since extracting 2 indexes from 3 bytes is efficient.

For each index, we fetch its corresponding entry from the decoding table. The whole entry is fetched from memory, even if the actual word is smaller than the maximum entry size. To avoid a performance penalty, we ensure that the table entries are aligned to the size of a cache line (64 bytes, usually).

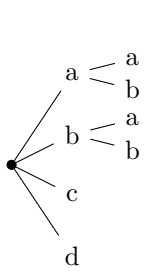
Next, we leverage the recently added ability of current processors of performing unaligned writes with almost no penalty, by copying the entire entry to the output stream. Again, copying the whole entry instead of only the relevant symbols of the word is performed to avoid checks. Finally, we increase the output pointer by the value stored in the last byte of the entry.

This algorithm has two key advantages. First, it has a deterministic input pipeline allowing parallel memory fetches; secondly, both word and word length are retrieved from a single memory fetch.

B.2. Dictionary Generation

We define $A = \{a_1, a_2, \cdots, a_N\}$ as an alphabet of input symbols sorted in order of non-increasing probability (*i.e.*, $P(a_n) \geq P(a_{n+1}), \forall n$). We generate a dictionary $\mathcal{W}$ in the form of a tree where each node corresponds to an input symbol.

We build $\mathcal{W}$ as follows: First, we initialize the tree with a leaf for each input symbol. Then, while $|\mathcal{W}|$ is smaller than desired, we add a single child to the most



(a)

a	*	*	1	(000)
a	a	*	2	(001)
a	b	*	2	(010)
b	*	*	1	(011)
b	a	*	2	(100)
b	b	*	2	(101)
c	*	*	1	(110)
d	*	*	1	(111)

(b)

Decoding **001**011110:

a	a	*	2	*	*	*	···

Decoding 001**011**110:

a	a	b	*	*	1	*	···

Decoding 001011**110**:

a	a	b	c	*	*	1	···

(c)

Figure B.2.: Decoding example where *aabc* is coded with the dictionary (a) to generate *001-011-110*. The decoder generates the table (b) from (a), storing the word length in the last byte. The decoding process (c) involves copying the whole entry to the output stream, but advance the pointer only the actual word length.

probable node. The new node will contain the symbol a_{i+1} where i is the current number of leaves of the parent node. As a special case, if a node has only one remaining leaf to grow, this leaf is grown automatically. See Fig. B.3 for an example.

B.2.1. Calculating Word Probabilities for a given Markov Process

As we generate plurally parsable dictionaries, the compression process is not steady and we model it as a Markov process with $N-1$ possible states $S = s_1, s_2, \cdots, s_{N-1}$. On s_i, words can not start with symbols with an index smaller than i.

We define the probability of the word w given that we are in state s_i as:

$$P(w|s_i) = \frac{P(w_1)}{\Sigma_{n=i}^{N} P(a_n)} \cdot \prod_{n=2}^{|w|} P(w_n) \cdot \sum_{n=1+c(w)}^{N} P(a_n), \tag{B.1}$$

where w_n is the n'th symbol of w, and $c(w)$ is the number of children of the last node of w. Thus the non-conditioned probability of w is:

$$P(w) = \sum_{i}^{N-1} P(s_i) \cdot P(w|s_i), \tag{B.2}$$

where $P(s_i)$ is the steady probability of being in state s_i. Therefore, we need to know the steady probabilities of the Markov process to calculate word probabilities.

B.2.2. Calculating the Steady State Probabilities of a Markov Process for a given Dictionary

We calculate now the transition matrix for the Markov process T:

$$T_{ij} = \sum_{w \in \mathcal{W}_j} P(w|s_i), \tag{B.3}$$

where T_{ij} is transition probability from s_i to s_j, and $\mathcal{W}_j$ are the words that end at s_j.

Then, the steady probabilities of being in each state $P(s_i)$ correspond to the first row of T^n for $n \to \infty$ (we must note that T^n converges after only a few iterations).

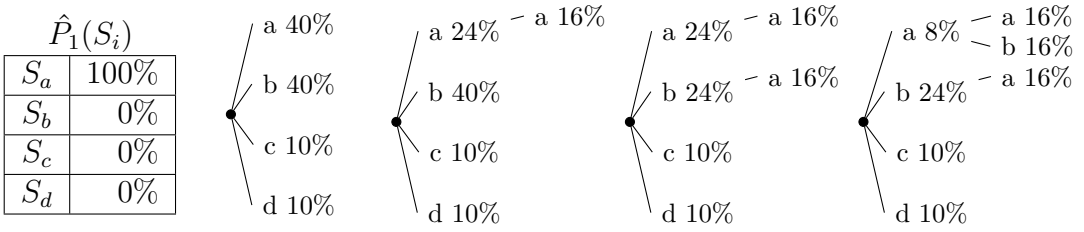
B.2.3. Building the Dictionary

We initialize the dictionary building process with a trivial Markov process corresponding to a unique parsable dictionary (see Fig. B.3.a). Then we build a full dictionary as previously stated (Fig. B.3.e). We use this dictionary to estimate a new Markov process (Fig. B.4.a), and build a new dictionary (Fig. B.4.e).

This process is repeated until convergence (Fig. B.5.b), which we found empirically to happen in a few iterations.

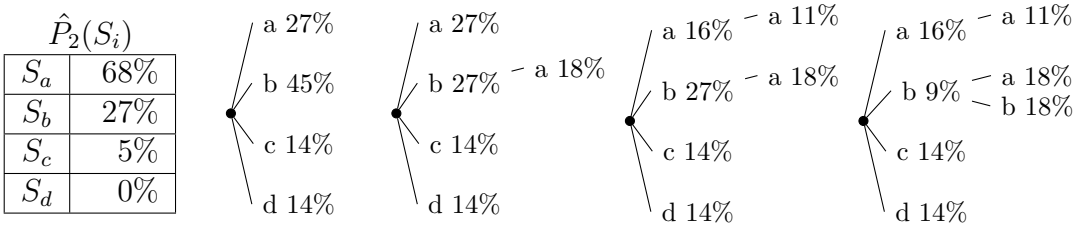
The average bit rate of the dictionary in bits is:

$$\mathrm{ABR}(\mathcal{W}) = \frac{log_2(|\mathcal{W}|)}{\sum_{w \in \mathcal{W}} P(w) \cdot |w|}. \tag{B.4}$$



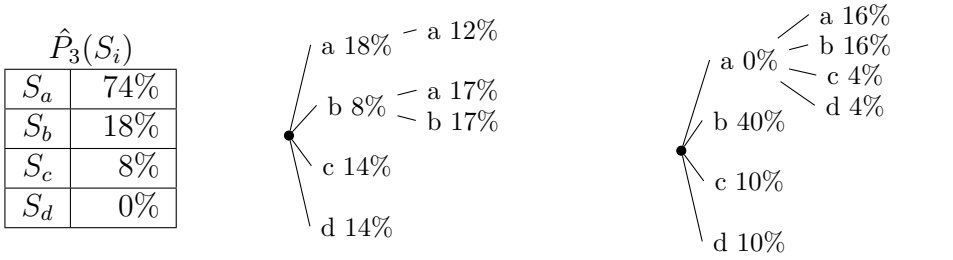
(a) Process (b) Initial (c) Step 1 (d) Step 2 (e) Step 3

Figure B.3.: $A = \{a, b, c, d\}$ with probabilities $\{40\%, 40\%, 10\%, 10\%\}$, (a) naive initialization of the Markov process steady probabilities, $\hat{P}_1(S_i)$, (b) tree initialization, (c)-(e) growing steps.



(a) Process (b) Initial (c) Step 1 (d) Step 2 (e) Step 3

Figure B.4.: Second stage of the dictionary creation. (a) second stage steady probabilities of the Markov process, $\hat{P}_2(S_i)$, which are estimated from the dictionary obtained in the previous stage (Fig. B.3.e), (b)-(e) the tree is grown again.



(a) Process (b) Marlin ABR($\mathcal{W}$) = 1.45 (c) Tunstall ABR($\mathcal{W}$) = 1.4

Figure B.5.: Third stage of the dictionary creation. (a) $\hat{P}_3(S_i)$ are estimated from the previous stage dictionary (Fig. B.4.e), (b) the resulting tree is identical to Fig. B.4.e, no more iterations are needed, (c) equivalent Tunstall dictionary, which is less efficient.

B.3. Evaluation

We compare Marlin to the following algorithms:

Tunstall (Tunstall, 1967): we use Marlin code with Tunstall dictionaries.

Rice (Rice and Plaunt, 1971): our implementation of Rice-Golomb codes (using the Rice subset) which is a popular entropy codec for Laplace and Exponential distributions (Greenwood, 2011).

Nibble (Cantero, 2010): used by Kinect and the fastest entropy encoder we are aware of. Values in $[-7, 7]$ are encoded in a nibble (4-bit word), larger values are emitted raw.

Snappy (Tarantov and Gunderson, 2011): Google's high throughput codec based on LZ77 without entropy stage.

LZO (Oberhumer, 1996): first high throughput algorithm that saw widespread use. It is used in the linux kernel and the btrfs file system. We evaluate the Lzo1-15 variety.

LZ4 (Collet, 2011): state-of-the-art high throughput algorithm developed by Yann Collet. Its current implementation is faster than Snappy due to improved hashing (Collet, 2013c).

Gipfeli (Lenhardt and Alakuijala, 2012): algorithm developed from Snappy that adds a simple entropy stage.

FiniteStateEntropy (Collet, 2013a): state-of-the-art implementation by Yann Collet of Jarek Duda's Asymmetric Numerical Systems theory (Duda, 2009). It is related to range encoding, but requires only one value to keep the state, making it faster.

Zstandard (Collet, 2015): a compression algorithm developed by Yann Collet that builds upon LZ4 and FiniteStateEntropy. We evaluate its fastest setting.

Gzip (Gailly, 1992): is often used as the reference to evaluate high throughout algorithms. It uses the deflate algorithm (LZ77 + Huffman).

CharLS (de Vaan, 2007): fast JPEG-LS implementation for lossless image compression.

Synthetic results. We evaluate the performance of Marlin on a circular Laplace distribution. This distribution arises often when dealing with differential data (Li et al., 2008). We report compression efficiency, encoding and decoding speed at entropy levels from 1% to 99%. Being a memoryless source, its compression ratio is limited by its entropy (Shannon and Weaver, 1949), therefore the compression efficiency is the ratio between the entropy and the average bit rate achieved: $\eta_X = H(X)/\mathrm{ABR}(X)$. Fig. B.6 shows that Marlin achieves compression efficiencies well above 80% for the entire entropy range. Only FiniteStateEntropy and Rice are able to compress better, but are significantly slower.

Results on real data. A typical application for entropy codecs is to compress noisy sensor data (*e.g.*, lossless image compression). We use Marlin to compress the

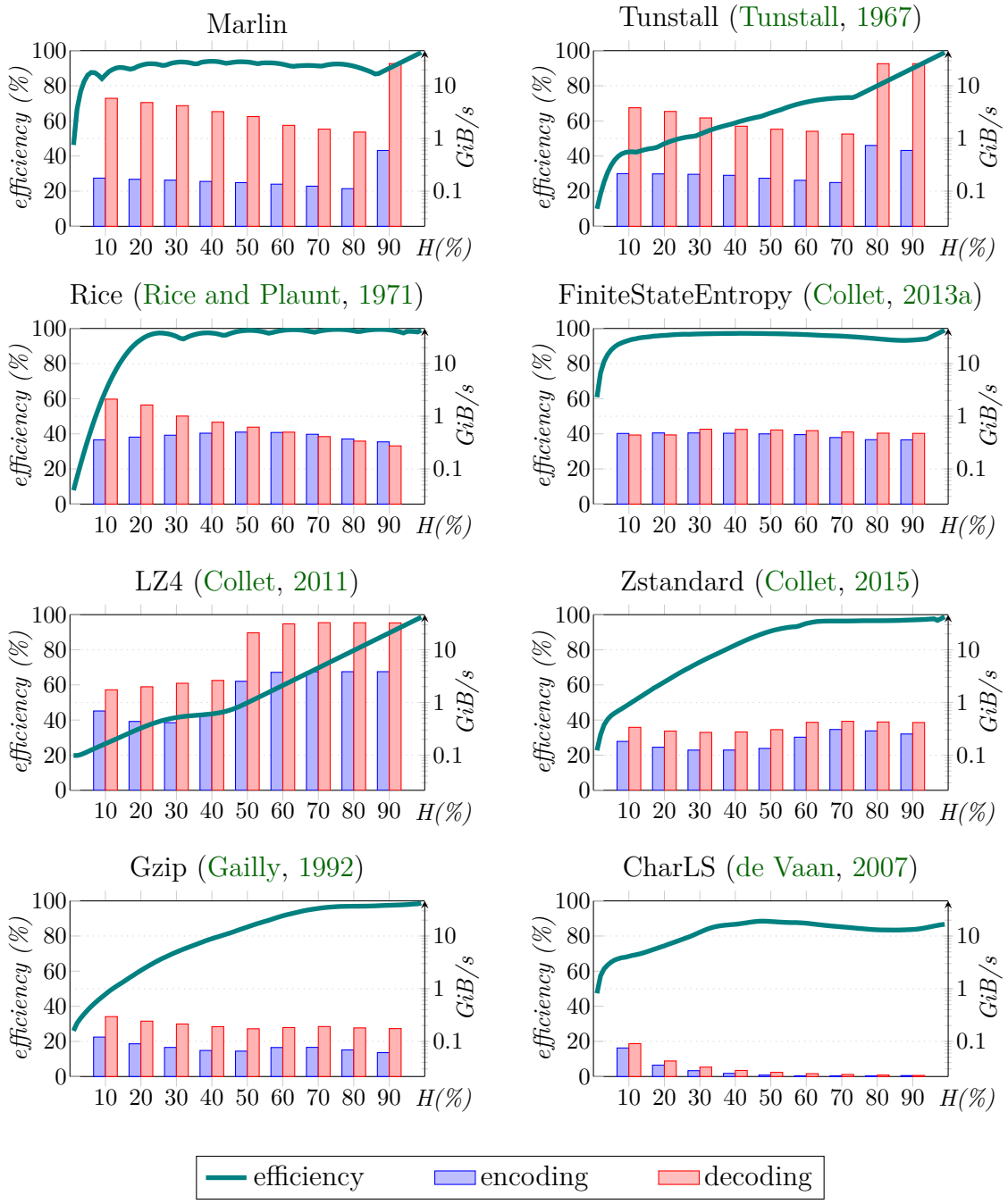


Figure B.6.: Evaluation on synthetic memoryless Laplace sources of varying entropies. Marlin's compression efficiency is above 80% and its decoding speed is above 1GiB/s. Tunstall, with equally sized dictionaries, is significantly less efficient. FiniteStateEntropy and Rice compress better, but are slower. LZ4 speed is comparable, but has poor efficiency. Note that speeds above 10GiB/s correspond to uncompressed data.

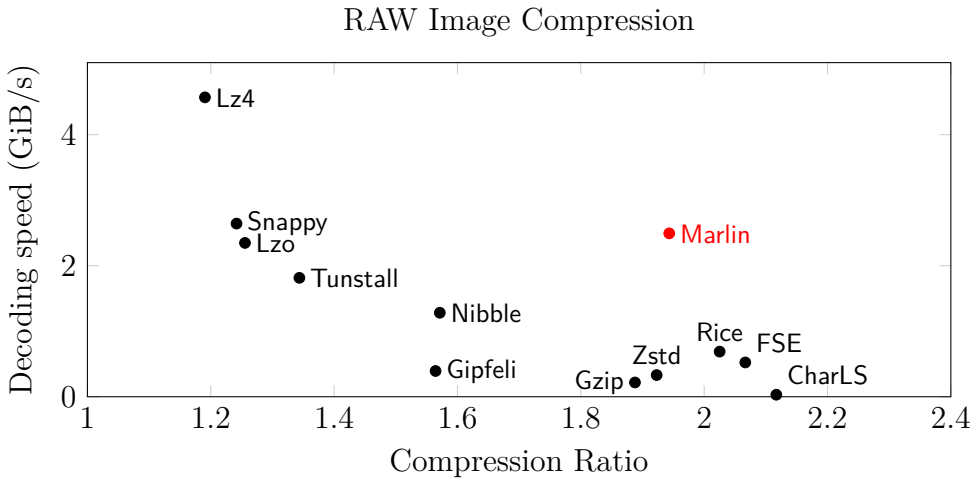


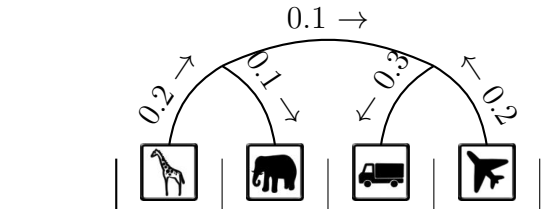
Figure B.7.: Evaluation on the Rawzor image compression dataset. The current state-of-the-art algorithm for high throughput entropy coding, FiniteStateEntropy (FSE) achieves a compression ratio of 2.07 at 524 MiB/s. Our algorithm, Marlin, achieves a compression ratio of 1.94 at 2494 MiB/s, which is 4.76 times faster.

Rawzor (Garg, 2011) image set using blocks of 64×64 pixels (4096 bytes) which are processed independently. The prediction model uses the pixel above, and we compress the residuals. Fig. B.7 shows that Marlin achieves decompression speeds in the same range as Snappy, while the compression ratio is on the same range as Zstd and Rice, and close to FiniteStateEntropy.

B.4. Discussion

Marlin is a High Throughput compression algorithm that achieves competitive compression efficiency for memoryless sources. Marlin builds upon a novel Variable to Fixed code using plurally parsable dictionaries, and a very optimized branchless decoding algorithm. As a consequence, Marlin achieves decoding speeds above the GiB/s range. However, the current implementation is to be understood as a proof of concept, and we expect to improve the decoding rate using processor intrinsics and new optimization techniques.

C. Earth Mover's Distance as a Loss Function



$f\left(\text{  }\right)$	0.2	0.4	0.2	0.2
$y_{\text{  }}$	0.0	0.5	0.5	0.0
∇EMD	1.5	−0.5	−1.5	0.5
∇EMD^2	0.5	−0.1	−0.7	0.3

Figure C.1.: We train a deep learning model for *Ellyvan*. The output of the forward pass is presented in the first row, and the label in the second. We present a method to train the network using gradients from the Earth Mover's Distance (∇EMD) in tree conected spaces, and we introduce a regularized form that converges faster (∇EMD^2).

When studying breathing signals in the frequency space, we found out that there was not an efficient way to compare histograms within a deep learning framework. The available criterions to compare histograms were Mean Squared Error (MSE) and Küllback-Leibler divergence (KLD), and both criterions compare bins independently one to each other.

On the other hand, the Earth Mover's Distance (EMD) is widely accepted as the best way to compare histograms, and is widely considered to be the native ℓ_1 distance between histograms (Rubner et al., 1998). The name is derived from a visual analogy of the data distributions as two piles of *dirt* (earth). EMD is defined as the minimum amount of effort required to make both distributions look alike. The EMD is widely used to compare histograms and probability distributions (Marinai et al., 2011; Peleg et al., 1989; Rolet et al., 2016; Rubner et al., 2000, 1998).

However, calculating the EMD for the general case is known to be computationally expensive. Furthermore, to use the EMD as a loss function, we need not the distance itself but its gradient, which is even more time consuming to calculate.

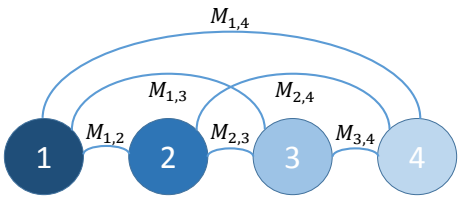


Figure C.2.: We illustrate the general case of the EMD, where *dirt* can be moved from and to all bins. Computing the amount of *dirt* that goes (flows) through each path is expensive.

In this section we suggest a closed form solution for the EMD gradient for all output graphs that are tree-connected, this includes 1-dimensional histograms and all hierarchic trees (see Fig. C.1). We do this by focusing on the structure of the output space instead of simply considering the node-to-node distance.

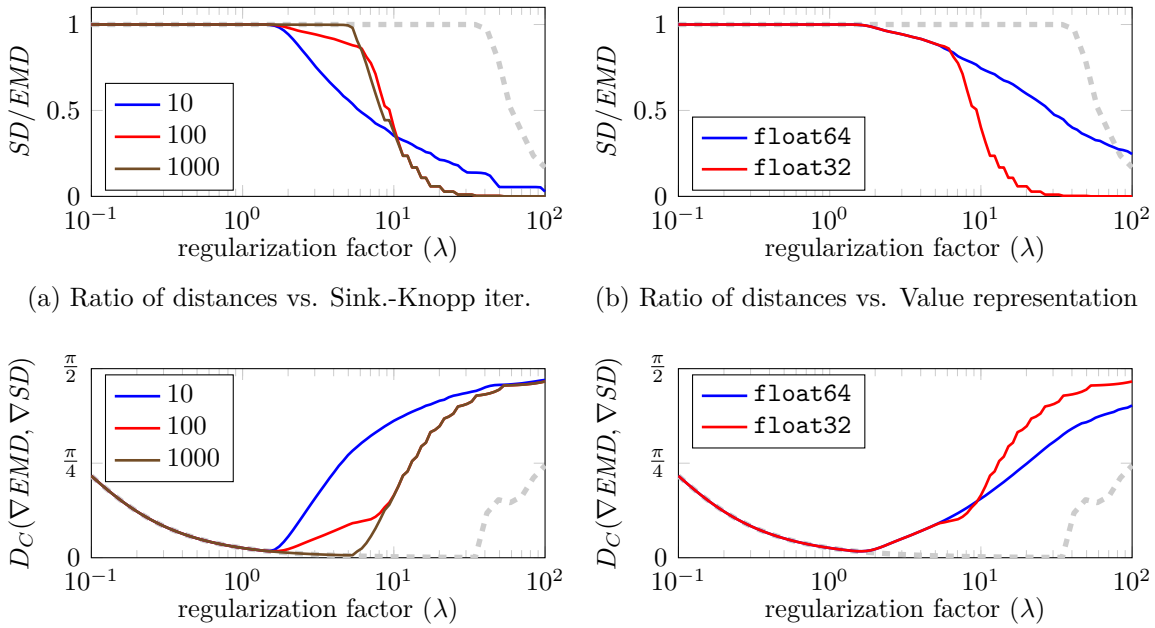
Finally, we propose a squared version of the EMD gradient that exhibits similar structure but converges faster due to its ℓ_2 behavior.

C.1. Related Work

Solving EMD optimization problems often require *lasso* optimization techniques (*e.g.*, mirror descent, Bregman projections, etc.). This represents a significant drawback for current deep learning approaches that strongly favor gradient-based methods such as Stochastic Gradient Descent, Momentum (Sutskever et al., 2013), and Adam (Kingma and Ba, 2015), that provide several small updates to the model parameters.

Using the EMD within an iterative optimization scheme was made feasible by Cuturi (Cuturi, 2013), who realized that an entropically regularized EMD can be efficiently calculated using the Sinkhorn-Knopp (Sinkhorn, 1967) algorithm. The resulting is referred to as Sinkhorn Distance (SD), and has achieved wide popularity within a number of learning frameworks (Benamou et al., 2015; Bonneel et al., 2015; Cuturi et al., 2016; Frogner et al., 2015; Montavon et al., 2015; Rabin and Papadakis, 2015; Rolet et al., 2016; Solomon et al., 2015, 2014a,b). The SD approximates EMD effectively, and provides a subgradient for the EMD as a side result of the estimation. This SD subgradient has been used to train deep learning models (Frogner et al., 2015) and is implemented as a loss criterion in popular deep frameworks such as Caffe (Jia et al., 2014) and Mocha (Zhang, 2015). However, as SD is an ℓ_1 norm, Frogner *et al.* (Frogner et al., 2015) need to combine SD with the KLD and use an exceedingly small learning rate for it to converge.

Other relaxed versions of the EMD exist for cases where speed is critical, *e.g.*, when comparing feature vectors (Ling and Okada, 2007; Pele and Werman, 2009; Rabin et al., 2008). Also, chain-connected distributions (see Fig. C.4) have a well-known closed-form solution (Vallender, 1974), which we use as our starting point.



(a) Ratio of distances vs. Sink.-Knopp iter.

(b) Ratio of distances vs. Value representation

(c) Angle between gradients vs. Sink.-Knopp iter.

(d) Angle between gradients vs. Value representation

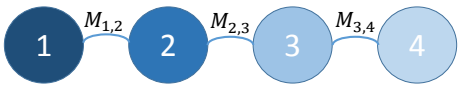
Figure C.3.: We show how implementation and hyper-parameter choices affect *SD* with an experiment on the WordNet hierarchy for 1000 ImageNet classes. (a,b): Ratio of *SD* to the real *EMD*. (c,d): Angle of the cosine distance ($D_C(\cdot,\cdot)$) between *EMD* and *SD* gradients. (a,c): Impact of limiting Sinkhorn-Knopp iterations. We see that larger values of λ require a large amount of iterations to converge. (b,d): `float32` (GPU) has a reduced dynamic range compared to `float64` (CPU), which degenerates for large λ values. The gray dashed line shown as a reference, corresponds to 10000 iterations using `float64` representation. The most commonly cited value of λ is 10.

C.2. Earth Mover's Distance

The EMD is defined for discrete distributions, hence, the probability mass (or *dirt*) is distributed in discrete piles or bins. The effort of moving a mound of *dirt* between two bins is a non-negative cost which is linearly proportional to the amount of *dirt* and distance between the bins.

Within this discrete domain, the general form of EMD between two distributions $\mathbf{p}, \mathbf{q} \in \mathbb{R}_+^N$ with $\|\mathbf{p}\|_1 = \|\mathbf{q}\|_1$ is

$$EMD(\mathbf{p}, \mathbf{q}) = \inf_{T \in U(\mathbf{p},\mathbf{q})} \langle M, T \rangle \,, \tag{C.1}$$



(a) Chain *EMD*

Figure C.4.: One dimensional probability distributions are an example of chain output spaces. In chains, *dirt* must travel through all intermediate bins in its path. This simplifies the computation of the EMD, as it only requires to eliminate one node per step.

where $\langle \cdot, \cdot \rangle$ is the Frobenius inner product and $M \in \mathbb{R}_+^{N \times N}$ defines the generalized distance between bins (see Fig. C.2). $U(\mathbf{p}, \mathbf{q})$ is the set of valid transport plans between $\mathbf{p}$ and $\mathbf{q}$,

$$U(\mathbf{p}, \mathbf{q}) = \{T \in \mathbb{R}_+^{N \times N} : T \cdot \mathbf{1}_N = \mathbf{p},\, T^\top \cdot \mathbf{1}_N = \mathbf{q}\}\,. \tag{C.2}$$

$\mathbf{1}_N$ is an N dimensional vector of all ones, and T is constrained such that its row sum corresponds to the distribution $\mathbf{p}$ and column sum to $\mathbf{q}$.

Without loss of generality (a simple scalar normalization), in the rest of the paper we assume $\|\mathbf{p}\|_1 = \|\mathbf{q}\|_1 = 1$.

C.2.1. Sinkhorn Distance

The general formulation of the EMD (Eq. C.1) is solved using linear programming which is computationally expensive. However, this problem was greatly alleviated by Cuturi (Cuturi, 2013) who suggested a smoothing term for the EMD in the form of

$$SD_\lambda(\mathbf{p}, \mathbf{q}) = \inf_{T \in U(\mathbf{p}, \mathbf{q})} \langle M, T \rangle - \frac{1}{\lambda} \langle T, \log T \rangle\,, \tag{C.3}$$

which allows to use the Sinkhorn-Knopp algorithm (Sinkhorn, 1967) to obtain an iterative and efficient solution. The Sinkhorn-Knopp algorithm is notable as it converges fast and produces a subgradient for the SD without extra cost.

However, the SD is not always numerically stable, as we show in Fig. C.3. There, SD diverges significantly from EMD, making it unfit for deep learning.

C.3. *EMD* in Chain-Connected Spaces

We analyze the scenario where bins in distributions, like histograms and probabilities, are situated in a one dimensional space. Here, moving *dirt* from a source to target bin requires an ordered visit to every bin in between (see Fig. C.4).

The bin distance M can be defined recursively to ensure that only consecutive bin distances are considered.

$$M_{i,j} = \begin{cases} 0 & \text{if } i = j\,, \\ M_{i-1,j} + d_{i-1,i} & \text{if } i > j\,, \\ M_{j,i} & \text{if } i < j\,. \end{cases} \tag{C.4}$$

Here, $d_{i-1,i}$ is the distance between two consecutive bins (often being all equal to 1).

The above choice of bin distances facilitates a simple solution to calculate the EMD using a recursion. Essentially, each bin either receives all the excess *dirt* that results from leveling previous bins, or in case of a deficit, tries to provide for it. Note that the cost of going left-to-right ($M_{i,j}$) or right-to-left ($M_{j,i}$) is symmetric. The closed form recursive formulation for the EMD between two one-dimensional distributions is:

$$\overrightarrow{EMD}(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{N-1} M_{i,i+1} \cdot |\varphi_i|\,, \tag{C.5}$$

where φ_i represents the excess *dirt* that needs to be moved ahead or deficit in *dirt* that needs to be filled up to bin i.

$$\varphi_i = \sum_{j=1}^{i} (p_j - q_j)\ . \tag{C.6}$$

For notational brevity, we will refer to $M_{i,i+1}$ as $\hat{M}_i$. The above expression can be rewritten with the sign function as

$$\overrightarrow{EMD}(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{N-1} \hat{M}_i \cdot \operatorname{sgn}(\varphi_i) \cdot \varphi_i\,. \tag{C.7}$$

Note that as both distributions have the same amount of total mass, and we progressively level out the *dirt* over all bins, when we arrive to the last bin all *dirt* will have been leveled (*i.e.* $\varphi_N = 0$). Therefore, we compute the outer sum only up to $N - 1$.

C.3.1. Gradient of the Earth Mover's Distance

To integrate EMD as a loss function in an iterative gradient-based optimization approach, we need to compute the analytical form of the gradient. However, we must ensure that the gradient obeys the law of *dirt* conservation. The gradient should not create new or destroy existing *dirt* to avoid changing the total mass ($\|\mathbf{p}\|_1 \neq 1$ after updates).

We use the trick of projected gradients, and define $\mathbf{e}_k$ as a vector of length N whose value at entry k is $1 - 1/N$, and $-1/N$ elsewhere. Note that $\mathbf{e}_k$ sums to 0.

For a small value h, we compute the distance between the perturbed distribution $\mathbf{p} + h\mathbf{e}_k$ and $\mathbf{q}$ as:

$$\overrightarrow{EMD}(\mathbf{p} + h\mathbf{e}_k, \mathbf{q}) \simeq \sum_{i=1}^{N-1} \hat{M}_i \cdot \operatorname{sgn}(\varphi_i) \sum_{j=1}^{i} (p_j + h(\delta_{jk} - 1/N) - q_j) \ , \tag{C.8}$$

where $\delta_{jk} = 1$ iff $j = k$. Note that by choosing h small enough, $\operatorname{sgn}(\varphi_i)$ can be assumed to remain unchanged. The corresponding partial derivative for the $\overrightarrow{EMD}$ is:

$$\frac{\partial \overrightarrow{EMD}(\mathbf{p}, \mathbf{q})}{\partial p_k} \simeq \sum_{i=1}^{N-1} \hat{M}_i \cdot \operatorname{sgn}(\varphi_i) \sum_{j=1}^{i} (\delta_{jk} - 1/N) \ . \tag{C.9}$$

C.3.2. Relaxed Earth Mover's Distance

The proposed gradient (Eq. C.9) is numerically stable and avoids erosion or addition of new dirt. This is an important step forward to use EMD in learning frameworks.

However, as the distance contains an absolute value function ($|\varphi_i|$) we observe difficulties converging to a solution, similar to ℓ_1 optimization. In particular, when $\hat{M}_i \in \mathbb{N}$, it is easy to see that the terms of ∇EMD are integer fractions of N (see Fig. C.1, multiples of 0.25). Furthermore, as small changes to the distribution do not change the gradient, the Hessian is zero except at a few discrete set of points (where the sign of φ_i changes) making optimization hard.

To solve these issues, we suggest a relaxed form of the EMD where the cost is calculated proportional to a power of the excess/deficit of *dirt*.

$$\overrightarrow{EMD}^{\rho}(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{N-1} \hat{M}_i \cdot |\varphi_i|^{\rho} \, . \tag{C.10}$$

whose gradient is:

$$\nabla \overrightarrow{EMD}^{\rho} \simeq \rho \sum_{i=1}^{N-1} \hat{M}_i \cdot \varphi_i \cdot |\varphi_i|^{\rho-2} \sum_{j=1}^{i} (\delta_{jk} - 1/N) \ . \tag{C.11}$$

For $\rho = 1$ we have the normal EMD distance, which behaves like ℓ_1. During gradient descend we suggest to use the case with $\rho = 2$ that bears similarity with the popular MSE loss. EMD^2 preserves the nice properties of conserving dirt, while having real valued gradients (see an example in Fig. C.1). In addition, we see that $\overrightarrow{EMD}^2$ also exhibits non-zero Hessians.

$$\frac{\partial^2 \overrightarrow{EMD}^2(\mathbf{p}, \mathbf{q})}{\partial p_k \partial p_l} = 2 \sum_{i=1}^{N-1} \hat{M}_i \cdot (N \cdot H(i-l) - i) \cdot (N \cdot H(i-k) - i) \, , \tag{C.12}$$

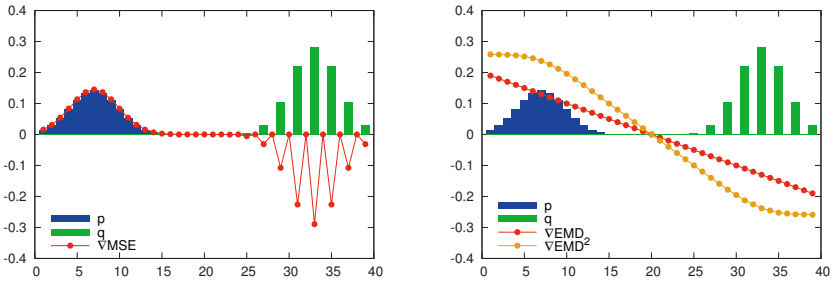
where $H : \mathbb{R} \rightarrow \{0, 1\}$ is the Heaviside step function defined as $H(n) = \{1 \text{ if } n \geq 0, 0 \text{ otherwise}\}$.

C.3.3. Discussion: comparing *EMD*, *SD* and *MSE*

Plotting the gradients provides a good impression of the actual behavior of the EMD. It also shows how EMD provides holistic optimization over the whole output space (full distribution).

In Fig. C.5 we show the gradients corresponding to several different loss criteria for the transformation between two unit-norm distributions: a smooth one $\mathbf{p}$, and a spiky one $\mathbf{q}$. We present the gradient of MSE in Fig. C.5 (a). Note how MSE optimizes each bin independently, and results in a non-smooth gradient.

In Fig. C.5 (b) we show the gradients for EMD and EMD^2. In both cases the gradient is holistic and affects the whole output space. Furthermore, the regularization effect induced by EMD^2 results in a smoother gradient.



(a) Mean Squared Error (b) Earth Mover's Distance

Figure C.5.: Gradient flow to convert a source distribution $\mathbf{p}$ to a target $\mathbf{q}$. (a) MSE gradient is not smooth, and does not affect the whole output space. (b) EMD and EMD^2 gradients affect the whole space. EMD has ℓ_1 behavior, while EMD^2 behaves like ℓ_2.

C.4. *EMD* in Tree-Connected Spaces

We extend here EMD to output spaces with a tree structure. Our formulation expects that all observed bins correspond to the leaves of the tree, and the remaining latent nodes and have no *dirt*. As we can link a tree to any non-leaf node with a zero-cost connection, this formulation allows us to express any tree structure. We refer to this analysis as the Hierarchical Earth Mover's Distance (HEMD),and we can see one example in Fig. C.1. Note that, this is still compatible with all our previous developments (as chains are a sub-class of trees). As such, we do not distinguish between EMD and HEMD while presenting the evaluation.

We define HEMD$^\rho$ as:

$$HEMD^\rho(\mathbf{p},\mathbf{q}) = \sum_{i\in G} M_{i,\mathbf{p}(i)} \cdot |\tilde{\varphi}_i|^\rho, \text{ where} \tag{C.13}$$

$$\tilde{\varphi}_i = \sum_{j\in \mathbf{l}(i)} (p_j - q_j) \ , \tag{C.14}$$

and $M_{i,\mathbf{p}(i)}$ is the cost of transporting *dirt* from node i to its parent (abbreviated as $\tilde{M}_i$), G is the set of all nodes in the tree, and $\mathbf{l}(i)$ is the set of all leaves in the subtree that has i as a root. N is the total number of leaves (and bins) in the tree. The intuition behind this formula is that we can reduce the tree one leave at a time as if it were the tail of a chain.

Then the gradient of HEMD$^\rho$ is:

$$\nabla HEMD^\rho \simeq \rho \sum_{i\in G} \tilde{M}_i \cdot \tilde{\varphi}_i \cdot |\tilde{\varphi}_i|^{\rho-2} \sum_{j=1}^{i} (\delta_{jk} - 1/N) \ . \tag{C.15}$$

All equations can be solved efficiently by a post-order traversal of the tree nodes.

A tree structured graph does not generally define the parent-child relationship between nodes. As the HEMD$^\rho$ definition is based on the parent-child relationship, there are multiple ways in which the HEMD$^\rho$ can be calculated for a given tree. However, we prove that HEMD$^\rho$ and by extension, its gradient, is uniquely defined for any given tree.

Proposition 1. *Let T and U be two identical undirected trees with different nodes selected as root. Then, the HEMD$^\rho$ between two distributions $\mathbf{p}$ and $\mathbf{q}$, is identical:*

$$HEMD^\rho_T(\mathbf{p},\mathbf{q}) = HEMD^\rho_U(\mathbf{p},\mathbf{q}).$$

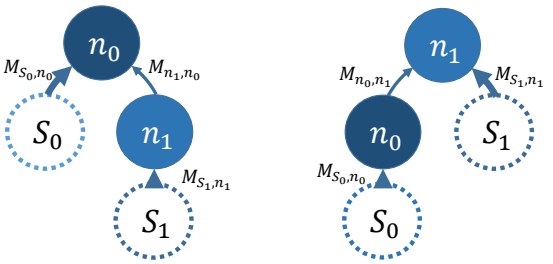
Proof. Let $\mathbf{p}$ and $\mathbf{q}$ be two distributions with $\|\mathbf{p}\|_1 = \|\mathbf{q}\|_1 = 1$. Let T and U be two undirected trees with nodes G and transportation cost function $M : G \to \mathbb{R}^+$. Let n_0 be the root node of T and n_1 be the root node of U. We prove that our proposition holds for the case where n_0 is adjacent to n_1. As a consequence, this proves by induction that our proposition is correct for every two root nodes.

We first simplify the tree structure (see Fig. C.6). Without any loss of generality, we group all adjacent nodes of n_0 except n_1 as a virtual subtree S_0 with M_{S_0,n_0} as the virtual transportation cost function from S_0 to n_0. Similarly, we group the adjacent nodes of n_1 except n_0 as S_1.

In this setting we have:

$$HEMD^\rho_T(\mathbf{p},\mathbf{q}) = M_{S_0,n_0}|\tilde{\varphi}^T_{S_0}|^\rho + M_{S_1,n_1}|\tilde{\varphi}^T_{S_1}|^\rho + M_{n_1,n_0}|\tilde{\varphi}^T_{n_1}|^\rho \text{, and} \tag{C.16}$$

$$HEMD^\rho_U(\mathbf{p},\mathbf{q}) = M_{S_0,n_0}|\tilde{\varphi}^U_{S_0}|^\rho + M_{S_1,n_1}|\tilde{\varphi}^U_{S_1}|^\rho + M_{n_0,n_1}|\tilde{\varphi}^U_{n_0}|^\rho \ . \tag{C.17}$$



(a) T (b) U

Figure C.6.: Identical trees T and U with different root nodes. S_0 and S_1 represent a bunch of nodes and corresponding subtrees connected to n_0 and n_1 respectively.

As S_0 and S_1 are identical in both trees, $\tilde{\varphi}^T_{S_0} = \tilde{\varphi}^U_{S_0}$ and $\tilde{\varphi}^T_{S_1} = \tilde{\varphi}^U_{S_1}$. Therefore, the difference of $HEMD^\rho$ between T and U is:

$$HEMD^\rho_T(\mathbf{p},\mathbf{q}) - HEMD^\rho_U(\mathbf{p},\mathbf{q}) = M_{n_1,n_0}|\tilde{\varphi}^T_{n_1}|^\rho - M_{n_0,n_1}|\tilde{\varphi}^U_{n_0}|^\rho\,. \tag{C.18}$$

Furthermore, M_{n_1,n_0} and M_{n_0,n_1} refer to the same edge, and as the graph is undirected, they are equal. Thus, to prove $HEMD^\rho_T(\mathbf{p},\mathbf{q}) = HEMD^\rho_U(\mathbf{p},\mathbf{q})$ we only need to show that $|\tilde{\varphi}^T_{n_1}|^\rho = |\tilde{\varphi}^U_{n_0}|^\rho$.

If n_0 is a leaf node of the tree, then $\tilde{\varphi}^U_{n_0} = p_0 - q_0$ and S_0 is empty. On the other hand, if n_0 is an intermediate node, then $\tilde{\varphi}^U_{n_0} = \tilde{\varphi}^U_{S_0}$ as $\mathbf{p}$ and $\mathbf{q}$ are not defined on non-leaf nodes. To cope with both cases, we define $\tilde{\varphi}^U_{n_0}$ as follows:

$$\tilde{\varphi}^U_{n_0} \;=\; p_0 - q_0 + \tilde{\varphi}^U_{S_0}\,, \tag{C.19}$$

and equivalently

$$\tilde{\varphi}^T_{n_0} \;=\; p_0 - q_0 + \tilde{\varphi}^T_{S_0} + \tilde{\varphi}^T_{n_1}\,. \tag{C.20}$$

As before, since the trees identical, $\tilde{\varphi}^T_{S_0} = \tilde{\varphi}^U_{S_0}$. This gives

$$\tilde{\varphi}^T_{n_0} \;=\; \tilde{\varphi}^U_{n_0} + \tilde{\varphi}^T_{n_1}\,, \tag{C.21}$$

Finally, as $\mathbf{p},\mathbf{q} \in \mathbb{R}^+$ and have the same ℓ_1 norm, we know that $\tilde{\varphi}$ is zero for any root node (as $\tilde{\varphi}$ is the sum of differences of all children, and the total amount of *dirt* is the same). In this case, since n_0 is the root of T, $\tilde{\varphi}^T_{n_0} = 0$. Applied to Eqn. C.21, we show that $\tilde{\varphi}^T_{n_1} = -\tilde{\varphi}^U_{n_0}$, and $|\tilde{\varphi}^T_{n_1}|^\rho = |\tilde{\varphi}^U_{n_0}|^\rho$ follows. Thus, $HEMD^\rho_T(\mathbf{p},\mathbf{q}) = HEMD^\rho_U(\mathbf{p},\mathbf{q})$.

□

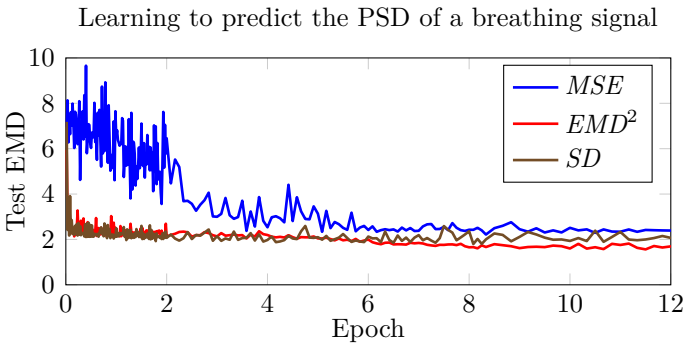


Figure C.7.: We train a regressor to estimate the PSD of real breathing signals obtained from chest excursions. Both EMD^2 and SD learn the transformation significantly faster than the MSE. Over a longer period EMD^2 achieves better accuracy than SD.

C.5. Experimental Analysis

Evaluation is performed on an i5-6600K CPU at 3.5GHz with 64GB of DDR4-2133 RAM and a GTX1080 GPU running Ubuntu 16.04, CUDA 8.0 and cuDNN 5.1.3. The SD hyper-parameters are $\lambda = 3$, the iteration limit 100, and using CUDA (*i.e.*, `float32` type).

C.5.1. Timing Analysis for *SD* vs. EMD^2

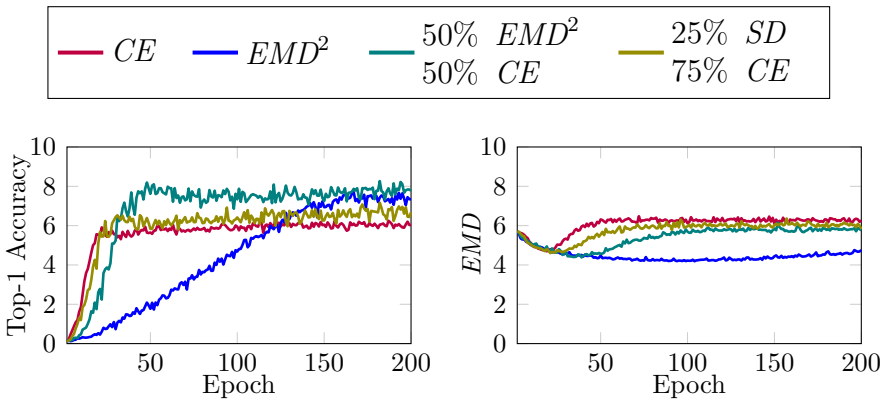
We evaluate the computational efficiency of EMD^2 and SD. The Sinkhorn-Knopp algorithm is very efficient and demonstrates fast GPU performance. However, being an iterative procedure, SD is significantly slower than EMD^2 (see Table. C.1). Furthermore, it is not practical for large output spaces, especially if we require the use of `float64` precision which is only available on CPUs.

	CPU	GPU
SD	7.51s	88.9ms
EMD^2	126ms	25ms

Table C.1.: Computation time for the gradients of SDand EMD^2 on WordNet for one minibatch of size 512. Our closed form solution for EMD^2 is 60x faster than SD on a CPU, and 3.5x faster on a GPU. Our unoptimized CUDA code is 3.5x faster than SD.

C.5.2. EMD^2 on Chain Spaces

We evaluate the use of EMD to learn PSD. This task is not only well-suited to use EMD as a loss criterion, but EMD also serves as the evaluation metric.



(a) Top-1 Accuracy (b) Holistic Error

Figure C.8.: We evaluate EMD on the 1000-class ImageNet Challenge. (a) On Top-1 precision, CE converges first, but the EMD metrics eventually surpass them. (b) On Hollistic precision, the EMD^2 outperforms others and its combinations by a large gap.

Our task is to predict the PSD of a breathing signal obtained from a patient using chest excursion signals (a nose thermistor acts as reference). Our data is recorded from 75 real patients from the Thoraxklinik-Heidelberg GmbH (THX) Dataset. We extract 200 one-minute clips for each patient, providing us with a dataset of 15,000 samples. We use data from 60 patients to train our models, and the remaining 15 as test subjects.

Noise levels depend on the activity of the patient, and are negligible when he/she is relaxed. On the other hand, when the patient moves, sits, or talks during the 1 minute segment, the correlation between chest movement and respiration disappears.

We adopt a two layer network for this experiment. The first layer consists of 16 temporal convolution filters with a receptive field of 11. We apply the *tanh* nonlinearity, and stack a fully connected layer on top. To ensure positive outputs (as we predict signal power) we apply the *square* function $(\cdot)^2$ to the output layer. We use the *Adam* optimizer (Kingma and Ba, 2015).

On this simple task *Adam* performs very well, and the model converges with all criteria (see Fig. C.7). Nevertheless, both EMD^2 and SD outperform MSE, converging in a fraction of the first epoch.

This highlights the benefits of using the EMD criterion in cases where it can be hard to obtain several training samples and the output space has a suitable structure.

C.5.3. *EMD*^{2} on Tree Spaces

We develop an experiment based on the well known 1000-class ImageNet object recognition challenge (Russakovsky et al., 2015).

We train a model based on Alexnet (Krizhevsky et al., 2012) with batch normalization (Ioffe and Szegedy, 2015) after Rectified Linear Unit (ReLU), using a minibatch size of 512, a learning rate of 0.05 with a decay of 10^{-5} and a ℓ_2 weight penalty of 10^{-4}. We use Stochastic Gradient Descent (SGD) as optimizer with momentum of 0.9. The input image is downsized to 112×112 pixels, and horizontal flipping and cropping is used for data augmentation while training.

The output space hierarchy tree is obtained from WordNet (Miller, 1995) and has a total of 1374 nodes and a maximum distance between nodes of 26. We set all edge costs to 1. By our definition, the output labels correspond to the leaves of the tree. Thus, the minimum hierarchical distance between a pair of output labels is 2.

Results are shown in Fig. C.8. The CE strongly favors Top-1 accuracy and converges fast on this metric, but the EMD based metrics catch up eventually. On the holistic accuracy, the EMD^2 shows the best performance.

C.6. Discussion

We derived closed-form solutions for EMD and its *dirt* conserving gradient on chain (*e.g.*, histograms) and tree (*e.g.*, hierarchies) output spaces. We also propose a relaxed version, EMD^2, of the original distance and compute its analytical form. Our EMD^2 exhibits better properties regarding numerical stability and convergence.

We hope that our contributions will help promote a wider adaption EMD as a loss criterion within deep learning frameworks.

Own Publications

Manuel Martinez, Monica Haurilet, Rainer Stiefelhagen, and Joan Serra-Sagristà. Marlin: A high throughput variable-to-fixed codec using plurally parsable dictionaries. In *Data Compression Conference (DCC)*, 2017a. 8

Manuel Martinez and Rainer Stiefelhagen. Breath rate monitoring during sleep from a depth camera under real-life conditions. In *Winter Conference on Applications of Computer Vision (WACV)*, 2017b. 8, 14

Manuel Martinez and Rainer Stiefelhagen. The SPHERE project: Sleep monitoring using computer vision. In *Forum Bildverarbeitung 2016*, page 221, 2016a. 8, 14

Manuel Martinez, Monica Haurilet, Ziad Al-Halah, et al. Relaxed earth mover's distances for chain-and tree-connected spaces and their use as a loss function in deep learning. *arXiv:1611.07573*, 2016b. 8

Timo Grimm, Manuel Martinez, Andreas Benz, and Rainer Stiefelhagen. Sleep position classification from a depth camera using bed aligned maps. In *International Conference on Pattern Recognition (ICPR)*, 2016. 8, 14

Andreas Benz, Manuel Martinez, Timo Grimm, and Rainer Stiefelhagen. Cordless sleep monitoring from a single depth camera. In *German Sleep Society (DGSM)*, 2016. 8

Manuel Martinez, Makarand Tapaswi, and Rainer Stiefelhagen. A closed-form gradient for the 1d earth mover's distance for spectral deep learning on biological data. In *ICML Workshop on Computational Biology (CompBio)*, 2016c. 8

Manuel Martinez, Lukas Rybok, and Rainer Stiefelhagen. Action recognition in bed using BAMs for assisted living and elderly care. In *International Conference on Machine Vision Applications (MVA)*, 2015. 14

Manuel Martinez and Rainer Stiefelhagen. Kinect unbiased. In *Int. Conference on Image Processing (ICIP)*, 2014a. 8

Manuel Martinez, Boris Schauerte, and Rainer Stiefelhagen. "BAM!" depth-based body analysis in critical care. In *Computer Analysis of Images and Patterns (CAIP)*, 2013a. 8, 14

Manuel Martinez and Rainer Stiefelhagen. Automated multi-camera system for long term behavioral monitoring in intensive care units. In *International Conference on Machine Vision Applications (MVA)*, 2013b. 8, 14

Manuel Martinez and Rainer Stiefelhagen. Kinect unleashed: Getting control over high resolution depth maps. In *International Conference on Machine Vision Applications (MVA)*, 2013c. 8

Manuel Martinez and Rainer Stiefelhagen. Breath rate monitoring during sleep using near-IR imagery and PCA. In *International Conference on Pattern Recognition (ICPR)*, 2012. 8, 14

Bibliography

Ali Al-Naji, Kim Gibson, Sang-Heon Lee, and Javaan Chahl. Real time apnoea monitoring of children using the microsoft kinect sensor: A pilot study. *Sensors*, 2017. 13, 14, 62, 80

Ahmad Al-Rababa'a and Danny Dubé. Using bit recycling to reduce the redundancy in plurally parsable dictionaries. In *Canadian Workshop on Information Theory*, 2015. 122

Michael Riis Andersen, Thomas Jensen, Pavel Lisouski, Anders Krogh Mortensen, Mikkel Kragh Hansen, Torben Gregersen, and Peter Ahrendt. Kinect depth sensor evaluation for computer vision applications. *Electrical and Computer Engineering Technical Report ECE-TR-6*, 2012. 107, 113

Hirooki Aoki, Masaki Miyazaki, Hidetoshi Nakamura, Ryo Furukawa, Ryusuke Sagawa, and Hiroshi Kawasaki. Non-contact respiration measurement using structured light 3-d sensor. In *SICE Annual Conference (SICE)*, 2012. 10, 14

Hirooki Aoki, Yasuhiro Takemura, Kazuhiro Mimura, Hiroichi Aoki, and Masato Nakajima. A non-contact and non-restricting respiration monitoring method for a sleeping person with a fiber-grating optical sensor. *Sleep and Biological Rhythms*, 1(3):249–250, 2003. 10, 14, 57

Hirooki Aoki, Yasuhiro Takemura, Kazuhiro. Mimura, and Masato Nakajima. Development of non-restrictive sensing system for sleeping person using fiber grating vision sensor. In *Micromechatronics and Human Science*, 2001. 10, 14, 57, 66, 68, 69, 111

Relja Arandjelovic and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 90

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 132, 141

Donald G Bailey. Sub-pixel profiling. In *Information, Communications and Signal Processing*, 2005. 68

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision (ECCV)*, 2006. 57

Pierrick Becouze, Christopher E Hann, J Geoffrey Chase, and Geoffrey M Shaw. Measuring facial grimacing for quantifying patient agitation in critical care. In *Computer Methods and Programs in Biomedicine*, 2007. 85, 86

Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. Iterative bregman projections for regularized transportation problems. *SIAM Journal on Scientific Computing*, 37(2):A1111–A1138, 2015. 132

Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and radon wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51(1):22–45, 2015. 132

Gary Bradski. The opencv library. In *Dr. Dobb's Journal of Software Tools*, 2000. 116

Nathan Burba, Mark Bolas, David M. Krum, and Evan A. Suma. Unobtrusive measurement of subtle nonverbal behaviors with the Microsoft Kinect. In *International Workshop on Ambient Information Technologies*, 2012. 66, 67, 68, 69, 70, 71

Hector Martin Cantero. the openkinect project. `openkinect.org`, 2010. [Online; accessed 15-February-2014]. 128

Rosalind D Cartwright. Effect of sleep position on sleep apnea severity. *Sleep: Journal of Sleep Research & Sleep Medicine*, 1984. 95

Fabio Centonze, Martin Schätz, Aleš Procházka, Jiří Kuchyňka, Oldřich Vyšata, Pavel Cejnar, and Martin Vališ. Feature extraction using ms kinect and data fusion in analysis of sleep disorders. In *International Workshop on Computational Intelligence for Multimedia Understanding (IWCIM)*, 2015. 14

Gerald Chanques, Samir Jaber, Eric Barbotte, Sophie Violet, Mustapha Sebbane, Pierre-François Perrigault, Claude Mann, Jean-Yves Lefrant, and Jean-Jacques Eledjam. Impact of systematic evaluation of pain and agitation in an intensive care unit. *Critical Care Medicine*, 2006. 84

Zhenyu Chen, Mu Lin, Fanglin Chen, Nicholas D Lane, Giuseppe Cardone, Rui Wang, Tianxing Li, Yiqiang Chen, Tanzeem Choudhury, and Andrew T Campbell. Unobtrusive sleep monitoring using smartphones. In *International Conference on Pervasive Computing Technologies for Healthcare*, 2013. 11, 14

Roger J Cole, Daniel F Kripke, William Gruen, Daniel J Mullaney, and J Christian Gillin. Automatic sleep/wake identification from wrist activity. *Journal of Sleep and Sleep Disorders Research (SLEEP)*, 15(5):461–469, 1992. 11, 14, 93

Yann Collet. Zstandard. facebook.github.io/zstd/, 2015. [Accessed 20-October-2016]. 128, 129

Yann Collet. Finitestateentropy. github.com/Cyan4973/FiniteStateEntropy, 2013a. [Accessed 20-October-2016]. 122, 128, 129

Yann Collet. Huff0. fastcompression.blogspot.de/p/huff0-range0-entropy-coders.html, 2013b. [Accessed 20-October-2016]. 122

Yann Collet. xxhash. cyan4973.github.io/xxHash/, 2013c. [Accessed 20-October-2016]. 128

Yann Collet. LZ4. lz4.org, 2011. [Accessed 20-October-2016]. 121, 122, 128, 129

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems (NIPS)*, 2013. 132, 134

Marco Cuturi, Gabriel Peyré, and Antoine Rolet. A smoothed dual approach for variational Wasserstein problems. *SIAM Journal on Imaging Sciences*, 9(1): 320–343, 2016. 132

Maria Cvach. Monitor alarm fatigue: an integrative review. *Biomedical instrumentation & technology*, 46(4):268–277, 2012. 4

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 96, 100

Sandrine Devot, Reimund Dratwa, and Elke Naujokat. Sleep/wake detection based on cardiorespiratory signals and actigraphy. In *International Conference of Engineering in Medicine and Biology Society (EMBC)*, 2010. 11, 14

Jarek Duda. Asymmetric numeral systems. *CoRR*, abs/0902.0271, 2009. 122, 128

Donna M Fick, Sharon K Inouye, Jamey Guess, Long H Ngo, Richard N Jones, Jane S Saczynski, and Edward R Marcantonio. Preliminary development of an ultrabrief two-item bedside test for delirium. *Journal of hospital medicine*, 10(10): 645–650, 2015. 4

Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya, and Tomaso A Poggio. Learning with a Wasserstein loss. In *Advances in Neural Information Processing Systems (NIPS)*, 2015. 132

Jean-loup Gailly. Gzip. www.gnu.org/software/gzip/, 1992. [Accessed 20-October-2016]. 128, 129

Sachin Garg. Rawzor: the new test images. www.imagecompression.info/test_images, 2011. [Accessed 20-October-2016]. 130

J Geoffrey Chase, Franck Agogue, Christina Starfinger, ZhuHui Lam, Geoffrey M Shaw, Andrew D Rudge, and Harsha Sirisena. Quantifying agitation in sedated ICU patients using digital imaging. In *Computer Methods and Programs in Biomedicine*, 2004. 12, 14, 57

Stefan Gradl, Heike Leutheuser, Patrick Kugler, Teresa Biermann, Sebastian Kreil, Johannes Kornhuber, Matthias Bergner, and B Eskofier. Somnography using unobtrusive motion sensors and android-based mobile phones. In *International Conference of Engineering in Medicine and Biology Society (EMBC)*, 2013. 11, 14

Mary Jo Grap, Virginia A. Hamilton, Ann McNallen, Jessica M. Ketchum, Al M. Best, Nyimas Y. Isti Arief, and Paul A. Wetzel. Actigraphy: Analyzing patient movement. *Heart and Lung: Journal of Acute and Critical Care*, 40(3), 2011. 4, 11, 14

Ben Greenwood. Lagarith lossless video codec. lags.leetcode.net/codec.html, 2011. [Accessed 20-October-2016]. 50, 128

Weixi Gu, Zheng Yang, Longfei Shangguan, Wei Sun, Kun Jin, and Yunhao Liu. Intelligent sleep stage mining service with smartphones. In *International Joint Conference on Pervasive and Ubiquitous Computing*, pages 649–660, 2014. 11, 14

Tian Hao, Guoliang Xing, and Gang Zhou. isleep: unobtrusive sleep quality monitoring using smartphones. In *ACM Conference on Embedded Networked Sensor Systems*, page 4, 2013. 11, 14

Tatsuya Harada, Akiko Sakata, Taketoshi Mori, and Tomomasa Sato. Sensor pillow system: monitoring respiration and body movement in sleep. In *International Conference on Intelligent Robots and Systems (IROS)*, 2000. 10, 14, 95

Danny Harnik, Ety Khaitzin, Dmitry Sotnikov, and Shai Taharlev. A fast implementation of Deflate. In *Data Compression Conference (DCC)*, 2014. 121

James M Harte, Christopher K Golby, Johanna Acosta, Edward F Nash, Ercihan Kiraci, Mark A Williams, Theodoros N Arvanitis, and Babu Naidu. Chest wall motion analysis in healthy volunteers and adults with cystic fibrosis using a novel kinect-based motion tracking system. *Medical & biological engineering & computing*, 54(11):1631–1640, 2016. 14

Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *Intelligent Systems*, 13:18–28, 1998. 100

Heiko Hirschmuller and Stefan Gehrig. Stereo matching in the presence of sub-pixel calibration errors. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 112, 113

Enamul Hoque, Robert F Dickerson, and John A Stankovic. Monitoring body positions and movements during sleep using wisps. In *Wireless Health*, pages 44–53, 2010. 10, 14, 95

Paul G Howard and Jeffrey Scott Vitter. Fast and efficient lossless image compression. In *Data Compression Conference (DCC)*, 1993. 122

Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Transactions on Information Theory*, 8(2):179–187, 1962. 96

Weimin Huang, Aung Aung Phyo Wai, Siang Fook Foo, Jit Biswas, Chi-Chun Hsia, and Koujuch Liou. Multimodal sleeping posture classification. In *International Conference on Pattern Recognition (ICPR)*, 2010. 11, 14, 57, 95

David A Huffman et al. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952. 121

Chris Idzikowski. Sleep position gives personality clue. *BBC News*, 2003. 95

Sharon K Inouye, Christopher H van Dyck, Cathy A Alessi, Sharyl Balkin, Alan P Siegal, and Ralph I Horwitz. Clarifying confusion: the confusion assessment methoda new method for detection of delirium. *Annals of internal medicine*, 113 (12):941–948, 1990. 4

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 142

Yoshiaki Itasaka, Soichiro Miyazaki, Kazuo Ishikawa, and Kiyoshi Togawa. The influence of sleep position and obesity on sleep apnea. *Psychiatry and clinical neurosciences*, pages 340–341, 2000. 95

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia (MM)*, 2014. 132

Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 2012. 107

Lawrence K Saul Kilian Q Weinberger, John Blitzer. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems (NIPS)*, 2006. 96

Panachit Kittipanya-Ngam, Ong Soh Guat, and Eng How Lung. Computer vision applications for patients monitoring system. In *Information Fusion*, 2012. 13, 58

Kurt Konolige. Projected texture stereo. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010. 19, 107

Kurt Konolige and Patrick Mihelich. Technical description of kinect calibration. www.ros.org/wiki/kinect_calibration/technical, 2010. [Online; accessed 15-December-2012]. 107, 109

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012. 5, 142

Shinobu Kumagai, Ryohei Uemura, Toru Ishibashi, Susumu Nakabayashi, Norikazu Arai, Takenori Kobayashi, Jun'ichi Kotoku, et al. Markerless respiratory motion tracking using single depth camera. *Open Journal of Medical Imaging*, 6(01):20, 2016. 14

Chung-Hsien Kuo, Fang-Chung Yang, Ming-Yuan Tsai, and Ming-Yih Lee. Artificial neural networks based sleep motion recognition using night vision cameras. *Biomedical Engineering: Applications, Basis and Communications*, 2004. 95

Ivan Laptev, Marcin Marszałek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 88

Jaehoon Lee, Min Hong, and Sungyong Ryu. Sleep monitoring system using kinect sensor. *International Journal of Distributed Sensor Networks*, 2015. 13, 14, 57, 95, 96

Yee Siong Lee, Pubudu N Pathirana, Christopher Louis Steinfort, and Terry Caelli. Monitoring and analysis of respiratory patterns using microwave doppler radar. *Translational Engineering in Health and Medicine*, 2:1–12, 2014. 12, 14

Rastislav Lenhardt and Jyrki Alakuijala. Gipfeli-high speed compression algorithm. In *Data Compression Conference (DCC)*, 2012. 121, 128

Jin Li, Moncef Gabbouj, Jarmo Takala, and Hexin Chen. Laplacian modeling of DCT coefficients for real-time encoding. In *IEEE International Conference on Multimedia and Expo*, 2008. 128

Wen-Hung Liao Wen-Hung Liao and Chien-Ming Yang Chien-Ming Yang. Video-based activity and movement pattern analysis in overnight sleep studies. *International Conference on Pattern Recognition (ICPR)*, 2008. 12, 14

Feng Lin, Yan Zhuang, Chen Song, Aosen Wang, Yiran Li, Changzhan Gu, Changzhi Li, and Wenyao Xu. Sleepsense: A noncontact and cost-effective sleep monitoring system. *IEEE Transactions on Biomedical Circuits and Systems*, 2016. 12, 14

Haibin Ling and Kazunori Okada. An efficient earth mover's distance algorithm for robust histogram comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(5):840–853, 2007. 132

Jian Liu, Yan Wang, Yingying Chen, Jie Yang, Xu Chen, and Jerry Cheng. Tracking vital signs during sleep leveraging off-the-shelf wifi. In *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 267–276, 2015. 12

Jason J Liu, Wenyao Xu, Ming-Chun Huang, Nabil Alshurafa, Majid Sarrafzadeh, Nitin Raut, and Behrooz Yadegar. A dense pressure sensitive bedsheet design for unobtrusive sleep posture monitoring. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 207–215, 2013. 10, 14

JoAnn Maklebust and Mary Sieggreen. *Pressure ulcers: guidelines for prevention and nursing management*. Springhouse Corporation, 1996. 4, 95

Kaveh Malakuti and Alexandra Branzan Albu. Towards an intelligent bed sensor: Non-intrusive monitoring of sleep irregularities with computer vision techniques. In *International Conference on Pattern Recognition (ICPR)*, 2010. 95

Muhammad N Mansor, Sazali Yaacob, Ramachandran Nagarajan, Lim S Che, Muthusamy Hariharan, and Muhd Ezanuddin. Detection of facial changes for ICU patients using KNN classifier. In *Intelligent and Advanced Systems (ICIAS)*, 2010a. 13, 14, 85, 86

Simone Marinai, Beatrice Miotti, and Giovanni Soda. Using earth mover's distance in the bag-of-visual-words model for mathematical symbol retrieval. In *International Conference on Document Analysis and Recognition (ICDAR)*, 2011. 131

G. Martin and N. Martin. Range encoding: an algorithm for removing redundancy from a digitised message. In *Institution of Electronic and Radio Engineers International Conference on Video and Data Recording*, 1979. 121

George A Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. 142

Grégoire Montavon, Klaus-Robert Müller, and Marco Cuturi. Wasserstein training of Boltzmann machines. *arXiv preprint arXiv:1507.01972*, 2015. 132

Kazuki Nakajima, Yoshiaki Matsumoto, and Toshiyo Tamura. Development of real-time image sequence analysis for evaluating posture change and respiratory rate of a subject in bed. *Physiological Measurements*, 22(3), 2001. 12

Hiroshi Nakano, Togo Ikeda, Makito Hayashi, Etsuko Ohshima, and Akihiro Onizuka. Effects of body position on snoring in apneic and nonapneic snorers. *Journal of Sleep and Sleep Disorders Research (SLEEP)*, 26(2):169–176, 2003. 95

Rajalakshmi Nandakumar, Shyamnath Gollakota, and Nathaniel Watson. Contactless sleep apnea detection on smartphones. In *International Conference on Mobile Systems, Applications, and Services*, 2015. 11, 14

Muhammad Naufal Bin Mansor, Sazali Yaacob, Ramachandran Nagarajan, and Muthusamy Hariharan. Patient monitoring in ICU under unstructured lighting condition. In *Industrial Electronics & Applications (ISIEA)*, 2010b. 13, 14, 84

Yoshifumi Nishida and Tashio Hori. Non-invasive and unrestrained monitoring of human respiratory system by sensorized environment. In *Sensors*, volume 1, pages 705–710, 2002. 10, 14

Markus Oberhumer. LZO: Lempel Zip Oberhumer. `www.oberhumer.com/opensource/lzo`, 1996. [Accessed 20-October-2016]. 121, 128

Eshed Ohn-Bar and Mohan M. Trivedi. Joint angles similarities and HOG2 for action recognition. In *Workshop on Computer Vision and Pattern Recognition (CVPRw)*, 2013. 88

J Ortmüller, T Gauer, M Wilms, H Handels, and R Werner. Respiratory surface motion measurement by microsoft kinect. *Current Directions in Biomedical Engineering*, 1(1):270–273, 2015. 14

Jae-Han Park, Yong-Deuk Shin, Ji-Hun Bae, and Moon-Hong Baeg. Spatial uncertainty model for visual features using a kinect™ sensor. *Sensors*, 2012. 107

Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *International Conference on Computer Vision (ICCV)*, 2009. 132

Shmuel Peleg, Michael Werman, and Hillel Rom. A unified approach to the change of resolution: Space and gray-level. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 11(7):739–742, 1989. 131

Thomas Penzel, MARION Moller, Heinrich F Becker, Lennart Knaack, and Jörg-Hermann Peter. Effect of sleep position and sleep stage on the collapsibility of the upper airways in patients with sleep apnea. *Journal of Sleep and Sleep Disorders Research (SLEEP)*, 24(1):90–95, 2001. 95

Ming-Zher Poh, Daniel J McDuff, and Rosalind W Picard. Advancements in non-contact, multiparameter physiological measurements using a webcam. *Biomedical Engineering*, 58(1):7–11, 2011. 74

Ming-Zher Poh, Daniel J. McDuff, and Rosalind W. Picard. Non-contact, automated cardiac pulse measurements using video imaging and blind source separation. *Optics Express*, 2010. 13

B Prathyusha, T Sreekanth Rao, and D Asha. Extraction of respiratory rate from PPG signals using PCA and EMD. *Engineering and Technology*, 1(2):164–184, 2012. 74

Julien Rabin, Julie Delon, and Yann Gou. Circular earth mover's distance for the comparison of local features. In *International Conference on Pattern Recognition (ICPR)*, 2008. 132

Julien Rabin and Nicolas Papadakis. Convex color image segmentation with optimal transport distances. In *Scale Space and Variational Methods in Computer Vision*, pages 256–269, 2015. 132

Tauhidur Rahman, Alexander T Adams, Ruth Vinisha Ravichandran, Mi Zhang, Shwetak N Patel, Julie A Kientz, and Tanzeem Choudhury. Dopplesleep: A contactless unobtrusive sleep sensing system using short-range doppler radar. In *Pervasive and Ubiquitous Computing*, pages 39–50, 2015. 12, 14

Miguel Reyes, Jordi Vitria, Petia Radeva, and Sergio Escalera. Real-time activity monitoring of inpatients. In *MICCAT*, 2010. 12, 14, 84

Robert Rice and James Plaunt. Adaptive variable-length coding for efficient compression of spacecraft television data. *Communication Technology*, 1971. 128, 129

Wietske Richard, Dennis Kox, Cindy den Herder, Martin Laman, Harm van Tinteren, and Nico de Vries. The role of sleep position in obstructive sleep apnea syndrome. *Oto-Rhino-Laryngology and Head & Neck*, 263(10):946–950, 2006. 95

Jorma Rissanen. Generalized Kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, 20(3):198–203, 1976. 121

Mahsan Rofouei, Mike Sinclair, Ray Bittner, Tom Blank, Nick Saw, Gerald DeJean, and Jeff Heffron. A non-invasive wearable neck-cuff system for real-time sleep monitoring. In *International Conference on Body Sensor Networks*, pages 156–161, 2011. 11, 14

Antoine Rolet, Marco Cuturi, and Gabriel Peyré. Fast dictionary learning with a smoothed Wasserstein loss. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016. 131, 132

Sam Roweis. EM algorithms for PCA and SPCA. In *Advances in Neural Information Processing Systems (NIPS)*, 1998. 69

Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision (IJCV)*, 40 (2):99–121, 2000. 131

Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. A metric for distributions with applications to image databases. In *International Conference on Computer Vision (ICCV)*, 1998. 131

Silvia Rus, Tobias Grosse-Puppendahl, and Arjan Kuijper. Recognition of bed postures using mutual capacitance sensing. In *European Conference on Ambient Intelligence*, pages 51–66, 2014. 10, 14

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 141

Soo Ryeon Ryu, Isao Noda, and Young Mee Jung. Moving window principal component analysis for detecting positional fluctuation of spectral changes. *Korean Chemical Society*, 2011. 69

Avi Sadeh and Christine Acebo. The role of actigraphy in sleep medicine. *Sleep medicine reviews*, 6(2):113–124, 2002. 84

Lauren Samy, Ming-Chun Huang, Jason J Liu, Wenyao Xu, and Majid Sarrafzadeh. Unobtrusive sleep stage identification using a pressure-sensitive bed sheet. *Sensors*, 14(7):2092–2101, 2014. 10, 14

Serap A Savari. Variable-to-fixed length codes and plurally parsable dictionaries. In *Data Compression Conference (DCC)*, 1999. 122

Lorenzo Scalise, Ilaria Ercoli, Paolo Marchionni, and Enrico Primo Tomasini. Measurement of respiration rate in preterm infants by laser doppler vibrometry. In *Medical Measurements and Applications Proceedings*, 2011. 12, 14

Boris Schauerte, Benjamin Kühn, Kristian Kroschel, and Rainer Stiefelhagen. Multimodal saliency-based attention for object-based scene analysis. In *International Conference on Intelligent Robots and Systems (IROS)*, 2011. 86

Secumatic. Wespot secnurse. http://www.healthcare-in-europe.com/en/article/1048-back-to-bed-safely.html, 2006. [Online; accessed 09-March-2017]. 13

W.P. Segars, S. Mori, G.T.Y. Chen, and B.M.W. Tsui. Modeling respiratory motion variations in the 4D NCAT phantom. In *Nuclear Science*, 2007. 66

Claude E. Shannon and Warren Weaver. The mathematical theory of communication. *The University of Illinois Press*, 1949. 128

Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013. 62, 63, 96

Richard Sinkhorn. Diagonal equivalence to matrices with prescribed row and column sums. *The American Mathematical Monthly*, 74(4):402–405, 1967. 132, 134

Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG)*, 34(4):66, 2015. 132

Justin Solomon, Raif Rustamov, Leonidas Guibas, and Adrian Butscher. Earth mover's distances on discrete surfaces. *ACM Transactions on Graphics (TOG)*, 33 (4):67, 2014a. 132

Justin Solomon, Raif Rustamov, Leonidas Guibas, and Adrian Butscher. Wasserstein propagation for semi-supervised learning. In *International Conference on Machine Learning (ICML)*, 2014b. 132

Luciane de Souza, Ana Amélia Benedito-Silva, ML Nogueira Pires, Dalva Poyares, Sergio Tufik, Helena Maria Calil, et al. Further validation of actigraphy for sleep studies. *Journal of Sleep and Sleep Disorders Research (SLEEP)*, 26(1):81–85, 2003. 93

Dan Strother. Fpga stereo vision project. danstrother.com/2011/01/24/fpga-stereo-vision-project, 2011. [Online; accessed 15-February-2014]. 109

I Sutskever, J Martens, G E Dahl, and G E Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning (ICML)*, 2013. 132

F Tahavori, E Adams, M Dabbs, L Aldridge, N Liversidge, E Donovan, T Jordan, PM Evans, and K Wells. Combining marker-less patient setup and respiratory

motion monitoring using low cost 3d camera technology. In *SPIE Medical Imaging*, 2015. 14

Chihiro Takano and Yuji Ohta. Heart rate measurement based on a time-lapse image. *Medical Engineering and Physics*, 29(8):853–857, 2007. 13, 69

Yasuhiro Takemura, Jun Y. Sato, and Masato Nakajima. A respiratory movement monitoring system using fiber-grating vision sensor for diagnosing sleep apnea syndrome. *Optical Review*, 2005. 10, 14, 57

K Song Tan, Reza Saatchi, Heather Elphick, and Derek Burke. Real-time vision based respiration monitoring system. In *Communication Systems Networks and Digital Signal Processing*, 2010. 66, 69

Mashiro Tanaka. Application of depth sensor for breathing rate counting. In *Asian Confrence on Control (ASCC)*, 2015. 14

Zeev Tarantov and Steinar Gunderson. Google Snappy: A fast compressor/decompressor. google.github.io/snappy, 2011. [Accessed 9-March-2016]. 121, 122, 128

Carlos Torres, Victor Fragoso, Scott Hammond, Jeffrey Fried, and B.S. Manjunath. Eye-cu: Sleep pose classification for healthcare using multimodal multiview data. In *Winter Conference on Applications of Computer Vision (WACV)*, 2016. 11, 14, 57, 95, 96, 100

Brian Parker Tunstall. Synthesis of noiseless compression codes. *Ph.D. dissertation, Georgia Institute of Technology*, 1967. 122, 128, 129

Jan de Vaan. CharLS, a JPEG-LS library. github.com/team-charls/charls, 2007. [Accessed 20-October-2016]. 122, 128, 129

S. S. Vallender. Calculation of the wasserstein distance between probability distributions on the line. *Theory of Probability & Its Applications*, 18(4):784–786, 1974. 132

Eduard E Vasilevskis, Jin H Han, Christopher G Hughes, and E Wesley Ely. Epidemiology and risk factors for delirium across hospital settings. *Best practice & research Clinical anaesthesiology*, 26(3):277–287, 2012. 3

Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa. Human action recognition by representing 3d skeletons as points in a lie group. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 88

Heng Wang, Alexander Klaser, Cordelia Schmid, and Cheng-lin Liu. Action recognition by dense trajectories. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 88, 90

Marcelo J Weinberger, Gadiel Seroussi, and Guillermo Sapiro. The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS. *IEEE Transactions on Image Processing*, 9(8):1309–1324, August 2000. 122

Daniel Weinland, Remi Ronfard, and Edmond Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding (CVIU)*, 115(2):224–241, 2011. 88

Ross N Williams. An extremely fast Ziv-Lempel data compression algorithm. In *Data Compression Conference (DCC)*, 1991. 121

Hirosuke Yamamoto and Hidetoshi Yokoo. Average-sense optimality and competitive optimality for almost instantaneous VF codes. *IEEE Transactions on Information Theory*, 47(6):2174–2184, 2001. 122

Satoshi Yoshida and Takuya Kida. An efficient algorithm for almost instantaneous VF code using multiplexed parse tree. In *Data Compression Conference (DCC)*, pages 219–228, 2010. 122

Meng-Chieh Yu, Huan Wu, Jia-Ling, Liou, Ming-Sui Lee, and Yi-Ping Hung. Multiparameter sleep monitoring using a depth camera. In *Biomedical Engineering Systems and Technologies*, pages 311–325, 2013. 13, 14, 57, 95, 96

Chiyuan Zhang. Mocha - a deep learning framework for Julia. `github.com/pluskid/Mocha.jl`, 2015. [Accessed 09-March-2017]. 132

Zhen Zhu, Jin Fei, and I. Pavlidis. Tracking human breath in infrared imaging. In *Bioinformatics and Bioengineering*, 2005. 13, 14

Domenico Zito, Domenico Pepe, Martina Mincica, Fabio Zito, Alessandro Tognetti, Antonio Lanatà, and Danilo De Rossi. SoC CMOS UWB pulse radar sensor for contactless respiratory rate monitoring. *IEEE Transactions on Biomedical Circuits and Systems*, 5(6):503–510, 2011. 12

Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977. 121

OMRON HSL-101. `www.healthcare.omron.co.jp/product/etc/hsl/hsl-101.html`, 2012. [Accessed 12-March-2017]. 12

Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001. 21

Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(7): 1409–1422, 2012. 21

Ahnaf Rashik Hassan and Mohammed Imamul Hassan Bhuiyan. An automated method for sleep staging from eeg signals using normal inverse gaussian parameters and adaptive boosting. *Neurocomputing*, 219:76–87, 2017. 92

David G Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision (ICCV)*, 1999. 57

Kai Berger, Marc Kastner, Yannic Schroeder, and Stefan Guthe. Using sparse optical flow for two-phase gas flow capturing with multiple kinect. In *Computer Vision and Machine Learning with RGB-D Sensors*, pages 157–169. Springer, 2014. 104, 108

David Fiedler and Heinrich Müller. Impact of thermal and environmental conditions on the kinect sensor. In *Advances in Depth Image Analysis and Applications*, pages 21–31. Springer, 2013. 113

Péter Várady, Tamas Micsik, Sandor Benedek, and Zoltán Benyó. A novel method for the detection of apnea and hypopnea events in respiration signals. *IEEE Transactions on Biomedical Engineering*, 49(9):936–942, 2002. 80

Péter Várady, Szabolcs Bongár, and Zoltán Benyó. Detection of airway obstructions and sleep apnea by analyzing the phase relation of respiration movement signals. *IEEE Transactions on Instrumentation and Measurement*, 52(1):2–6, 2003. 80

Alfredo Burgos, Alfredo Goñi, Arantza Illarramendi, and Jesús Bermúdez. Real-time detection of apneas on a PDA. *IEEE Transactions on Information Technology in Biomedicine*, 14(4):995–1002, 2010. 80

JY Tian and JQ Liu. Apnea detection based on time delay neural network. In *Engineering in Medicine and Biology Society*, 2005. 80

Jordi Sola-Soler, Raimon Jane, Jose Antonio Fiz, and Jose Morera. Automatic classification of subjects with and without sleep apnea through snoring analysis. In *Engineering in Medicine and Biology Society*, 2007. 80

J N McNames and A M Fraser. Obstructive sleep apnea classification based on spectrogram patterns in the electrocardiogram. In *Computers in Cardiology*, 2000. 80

John Parker Burg. A new analysis technique for time series data. In *Advanced Study Institute on Signal Processing*, 1968. 70

John Vanderkooy and Stanley P Lipshitz. Resolution below the least significant bit in digital systems with dither. *Journal of the Audio Engineering Society*, 32(3): 106–113, 1984. 71

Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972. 58