Karlsruhe Institute of Technology

A Guidance and Obstacle Evasion Software Framework for Visually Impaired People

Diploma thesis of

Daniel Koester

At the faculty of Computer Science
Institute for Anthropomatics

Reviewer:	Prof. Dr.-Ing. Rainer Stiefelhagen
Second reviewer:	Prof. Dr.-Ing. Rüdiger Dillmann
Advisor:	Dipl.-Inform. Boris Schauerte

Duration: 1. August 2012 – 31. January 2013

Computer Vision for Human-Computer Interaction Research Group
Institute for Anthropomatics
Karlsruhe Institute of Technology
Title: A Guidance and Obstacle Evasion Software Framework for Visually Impaired People
Author: Daniel Koester

Daniel Koester
Bernhardstraße 21
76131 Karlsruhe
dk@hyve.org

Statement of Authorship

I hereby declare that this thesis is my own original work which I created without illegitimate help by others, that I have not used any other sources or resources than the ones indicated and that due acknowledgement is given where reference is made to the work of others.

Karlsruhe, 31. January 2013

.....
(Daniel Koester)

Abstract

Information about the environment is desired in several applications, for example autonomous robots and support systems for visually impaired persons. Like with most scenarios where a human being uses a support system, reliability is of utmost importance. This creates a high demand for performance and robustness in real-world settings. Many systems created towards this purpose cannot cope with constraints such as platforms with a large amount of uncontrolled ego-motion and the need for real-time processing of information and are thus not feasible for this specific situation.

The topic of this thesis is a novel framework to create vision based support systems for visually impaired persons. It consists of a modular, easily extendable and highly agile software system. Furthermore, a ground detection system is created to aid in mobile navigation scenarios. The system calculates the accessible section by relying on the assumption that the orientation of a given plane segment can be calculated using a stereo camera reconstruction process.

Many frameworks have been created to simplify the developing process of large and complex systems and to foster collaboration among researchers. Usually, such frameworks would be created towards a certain purpose, for example a robotic application. In such a scenario, many elements are needed to manage the components of the robotic platform, such as motor controls. This creates dependencies on the availability of specific building blocks and induce great overhead if such components are not needed. Thus, the created framework imposes no restrictions on its use case by moving such functionality into modular components.

In computer vision many features and algorithms to detect ground plane exist. Some of these are quite costly to calculate, for example segmentation based algorithms. Others use a random sample consensus (RANSAC) based approach that shows problems in situations where the existing ground plane only accounts for a small part of the examined input data. To alleviate these problems a simple, yet robust, feature is proposed which consists of a gradient detection in the stereo reconstruction data. The gradient of a region in the disparity map correlates directly with the orientation of a surface in the real world. Since the gradient calculation is not complex, a fast and reliable computation of the accessible section becomes possible.

To evaluate the proposed ground detection system, a dataset was created. This dataset consists of 20 videos recorded with a hand held camera rig and contains a high degree of camera ego-motion to simulate a system worn by a pedestrian. The accessible section detection based on the gradient calculation shows promising results.

Kurzbeschreibung

Informationen über die nähere Umgebung sind in den verschiedensten Anwendungsfällen sehr begehrt, zum Beispiel für autonome Roboter und unterstützende Systeme für Blinde und sehbehinderte Menschen. Wie in den meisten Szenarien, in denen sich der Mensch auf ein unterstützendes System verlässt, ist Zuverlässigkeit ein entscheidender Faktor. Diese Tatsache stellt hohe Anforderungen an Performanz und Robustheit unter realen Bedingungen. Viele Systeme die zu diesem Zweck erstellt wurden weisen Probleme auf, welche durch die Verwendung von mobilen Plattformen mit einer starken Eigenbewegung sowie der Erfordernis an echtzeitfähiger Verarbeitung entstehen. Sie sind daher in solch spezifischen Situationen nicht verwendbar.

Der Schwerpunkt dieser Arbeit liegt auf einem neuartigen Software Framework für die bildbasierte Unterstützung von Blinden oder sehbehinderten Menschen. Es besteht aus einem modularen, einfach zu erweiternden und höchst agilem Software System. Weiterhin wird eine Methode zur Erkennung der begehbaren Fläche zur Hilfe bei der mobilen Orientierung einer Person erstellt. Dieses System ermittelt die begehbare Fläche basierend auf der Annahme, dass die Ausrichtung eines Flächenabschnittes aus den gewonnen Daten einer Stereo Rekonstruktion berechnet werden kann.

Viele Software Frameworks wurden bereits erstellt um den Entwicklungsprozess von großen und komplexen System zu vereinfachen und um die Zusammenarbeit zwischen Forschern voranzutreiben. Normalerweise werden solche Software Frameworks zu einem bestimmten Zweck erstellt, zum Beispiel der Steuerung eines Roboters. In solch einer Umgebung werden viele Elemente gebraucht und die verschiedensten Komponenten einer solchen Robotik-Plattform anzusprechen, z.B. Motor Steuerungen. Dies kann eine Abhängigkeit zu spezifischen Bestandteilen erzeugen und damit einen Mehraufwand hervorrufen, sollten solche Komponenten nicht gebraucht werden. Daher wurde das Framework so erstellt, dass es keinerlei Einschränkungen an sein Benutzungsszenario stellt, indem jegliche derartige Funktionalität ausgelagert wird.

In der Bild-Verarbeitung existieren bereits viele Algorithmen zur Bestimmung der Bodenfläche. Manche dieser Algorithmen sind sehr aufwendig zu berechnen, zum Beispiel Segmentierungen. Andere benutzen einen *random sample consensus (RANSAC)* basierten Ansatz, welcher Probleme aufzeigt, wenn die zu erkennende Bodenfläche nur einen kleinen Teil der zu untersuchenden Daten ausmacht. Um solche Probleme zu vermeiden, wird ein einfaches, jedoch robustes, Merkmal vorgestellt, welches auf der Berechnung eines Gradientens in einer Stereo Rekonstruktion basiert. Dabei lässt sich feststellen, dass der Gradient einer untersuchten Region der Stereo Rekonstruktion mit der Ausrichtung einer Fläche in der realen Welt korreliert. Da die Gradienten Berechnung nicht komplex ist, wird eine schnelle und zuverlässige Berechnung der begehbaren Sektion ermöglicht.

Um das erstellte System zu evaluieren, wurde ein Datensatz erstellt. Dieser Datensatz besteht aus 20 Videos, welche mit einer handgeführten Kameraanlage aufgenommen wurden. Die Videos enthalten einen hohen Grad an Eigenbewegung der Kameras um ein tragbares System zu simulieren. Die Bodenerkennung basierend auf der Gradienten Berechnung zeigt hierbei verheißungsvolle Ergebnisse.

Contents

List of Figures

List of Abbreviations

ARToolKit augmented reality toolkit 10
AR augmented reality 10
AUC area under the curve 38
BVS blind and visually impaired support system
CARMEN carnegie mellon navigation toolkit 9
CMake cross-platform make 17
CPU central processing unit 10
DTAM dense tracking and mapping 51
DWARF distributed wearable augmented reality framework 9
ELAS efficient large-scale stereo matching 30
GPS global positioning system 7
GPU graphics processing unit 26
GUI graphical user interface 50
IMU inertial measurement unit 8
IPC inter-process communication 10
IRLS iteratively reweighted least-squares 11
LIDAR light detection and ranging 1
MATLAB matrix laboratory 34
OpenCV open source computer vision library 9
PR precision-recall 38
PTAM parallel tracking and mapping 51
RANSAC random sample consensus v
RGB-D red-green-blue and depth 8
RGB red-green-blue 8
ROC receiver operating characteristic 38
ROS robot operating system 9
RTTI run-time type information 20
SLAM simultaneous localization and mapping 51
STOC stereo on chip 13
TCP transmission control protocol 10
TOF time of flight 8
UAV unmanned aerial vehicle 1

Acknowledgements

I want to thank the people without whom my diploma thesis would not have been made possible.

First, Rainer Stiefelhagen for the opportunity to write my thesis with his Computer Vision for Human-Computer Interaction Lab (CVHCI), for letting me participate in his research area and allowing me to concentrate my efforts on topics very dear to my heart. I am very grateful to Boris Schauerte for his interest in my diploma thesis topic, his support and guidance as well as his supervision and experience in all related matters. Furthermore, I am proud that Rainer Stiefelhagen is giving me the chance to become a member of his CVHCI lab. I suppose I also have to partially thank Boris for this opportunity, you are a great mentor and I am sure, will become even more so in the future.

My thanks go to all the CVHCI members for their interesting discussions as well as valuable input whenever problems arose. The supportive working atmosphere and the relocation to a new, much quieter facility, makes the CVHCI lab an amazing place to work in.

Many thanks and apologies go to my fellow lab students Matthias Lang and Timo Schneider for testing as well as using early prototypes of the created framework and creating bugs I could never even imagine. I would also like to thank my very good friends Paul Märgner and Jan-Ole Sasse for proofreading, your corrections are much valued.

I want to thank my family for their support and encouragement throughout the years of my academic studies. Especially my parents and my sister, who are optometrists, followed my diploma thesis with great interest. Their ongoing assistance has helped me greatly to pursue this endeavor. I hope to make them proud and to show them my appreciation for all they have done for me.

Finally, I want to thank my girlfriend Nicole. Not only for her efforts put into countless hours labeling the dataset with me, but especially for her love, support and care whenever it was most needed. You make my day.

1. Introduction

"Walking with a friend in the dark is better than walking alone in the light."
Helen Keller

Navigation in an unknown or known terrain has been an intensively researched topic for a long time. It exists in several scientific domains, either directly related to computer vision or using computer vision as a means to gather such information, often in combination with other methods and sensors. A robot could search for a path in an unknown terrain. An automobile could not only scan for the existence of road, respectively lanes in front of the car, but additionally try to warn the driver against pedestrians or other traffic participants that are getting in the way or too close to the vehicle. An unmanned aerial vehicle (UAV) could try to avoid collisions while being controlled remotely or flying autonomously. A visually impaired person could simply search for a way to walk without colliding with moving or static obstacles.

There already exists an entire array of techniques and specialized hardware to gather information in an unknown surrounding, such as for example radar, sonar or light detection and ranging (LIDAR). Most of these require elaborate, complex, obtrusive, bulky or expensive hardware. Meanwhile, digital cameras and raw computing power have seen a constant decrease in cost and size. This manifests itself in mobile smartphones which have become commonplace in our modern world. Computer vision can replace their more expensive counterparts and provide an inexpensive and mobile alternative for navigational aid systems.

Figure 1.1.: A visually impaired person testing drivers' behavior when presented with a blind person crossing the street in Switzerland. By law, drivers are required to stop and let them pass. (Source: [Sü10])

For visually impaired people this advance in technology can have a great impact upon everyday situations. For example, figure 1.1 shows a visually impaired person facing the difficulty of crossing a street in Switzerland. A digital assistant could not only help in such a navigational context, but also support its user in various other activities. It could, for example, help regain lost or misplaced items or read text off (street) signs to help with orientation inside buildings or in pedestrian areas. Furthermore, books could be made accessible without having to create a special braille version. Pricing information could be read while shopping. Moreover, broader information about the user's surrounding in specific situations, such as the number of people in a room or whether a person is currently looking at the user, might also be desired.

Such support systems can help visually impaired persons to reduce their dependence upon human assistance. This is important for self-determination and a recovery of autonomy in everyday situations. While some of these things might also be interesting for everyone, the helpfulness of such mobile systems, especially for visually impaired people, can hardly be emphasized enough.

1.1. Motivation

Numerous systems to recognize either ground plane, certain classes of obstacles, walkway or road surface have already been proposed. Most of these do not allow for an intense

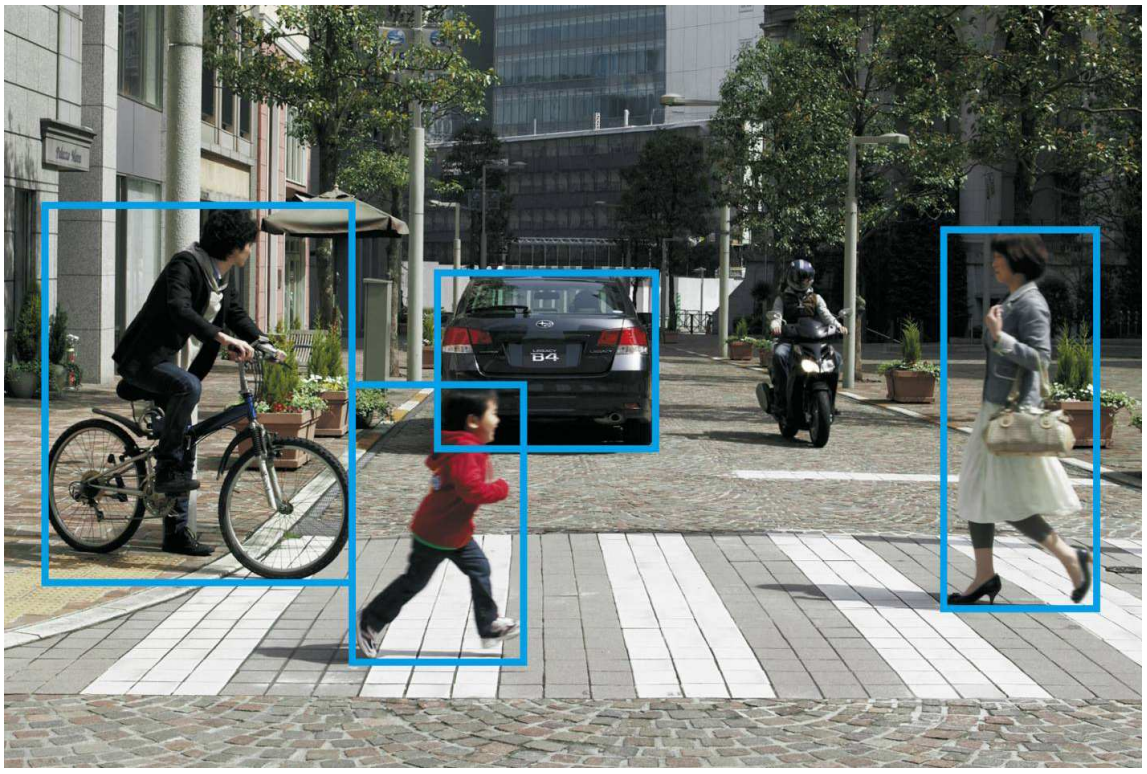


Figure 1.2.: Subaru's EyeSight system can recognize pedestrians, cyclists and other cars to help drivers avoid collisions. It also triggers several active, passive and preventive systems when hazards are detected. (Source: [Ste11])

camera movement and have so far been used in rather static settings such as upon slowly moving robots. This is due to the fact that they impose constraints upon the observed application-dependent environment, for example viewing a typical street scene from a moving car, as can be seen in figure 1.2.

These constraints are a problem when applied to a more versatile platform, as such systems should be worn on the body by a visually impaired person, which induces ego-motion. To become insusceptible to such ego-motion, the entire system needs to be built without relying on specific external conditions, such as the exact distance of the camera to the ground, the angle of the camera towards the horizon or the direction of the camera movement as well as its possible rotations. Thus, most of the existing systems cannot be used in such a setting.

On the software side, most systems are built for a specific subset of the possible settings. These situations often enforce specific requirements, for example integration into existing frameworks and commercial robotic or automotive platforms. Such platforms always contain domain specific components and restrictions that are hard to remove in case they are not required. The reuse of individual components is made harder by dependencies between those components as well as reliance on specific framework attributes. In order to increase reusability, a framework should thus be as abstract as possible while still being built towards a particular target.

1.2. Contributions

Even though there already exist quite a few software frameworks, the focus of this thesis compartmentalizes itself between a software framework and a base method for a ground detection system for visually impaired people.

A framework's role should be to foster collaboration between researchers by providing a common platform as well as tools to simplify the development process. Furthermore, it can help to reduce development overhead by splitting the effort between researchers. This is done by creating a modular software framework, which can easily be extended with software modules. Each of these modules can represent a small part, even a single functionality. It is also possible to create a module that contains an entire collection of related functionality. Due to the separation of ideas and principles into encapsulated modules it becomes easy and straightforward to reuse them. This has been inspired by modern object oriented design paradigms as well as already existing systems. Since research mostly focuses on a very small and specific topic, a main design aspect is ease of use and that a new system for a specific research interest can be build from scratch as fast and easy as possible. In addition, much infrastructure will be built around the core framework to provide support tools for common tasks.

A method for a ground detection system for visually impaired persons is then build using above framework. Its prospective purpose is to provide a research base for further investigation into a mobile navigation system for visually impaired persons. It will contain functionality for ground detection and basic obstacle avoidance. An already existing library [GRU11] to retrieve stereo disparities, also known as depth-images, is used. These depth images are the basis of the ground detection system, so its correctness is heavily dependent on the stereo algorithm quality. The ground detection part of the system uses a strong correlation between an analyzed region's gradient and the surface normal of the world area it represents. Next, a simple search for the accessible section is performed using the calculated surface normals. This section is then returned by the ground detection system.

1.3. Outline

Chapter 2 lists and discusses related work. Moreover, chapter 2 provides further motivation for the created framework and the ground detection system for visually impaired people. Therefore, it is separated into several sections and subsections, each of which concentrates on a specific component. It starts with a short summary of some of the already existing systems that act as navigational aids for visually impaired people. Then it moves onwards to a discussion of similar frameworks, based in robotics and computer vision. It finishes with a short summary of some ground plane as well as obstacle detection approaches.

In chapter 3, a high level summary of the created framework is given. Some design issues as well as the compromises made are discussed in detail. Public and private interfaces are described, as well as selected base modules. It finishes with a short description of additional support tools, which have been created to aid the user with various tasks.

The ground detection method is described in chapter 4. A simple, yet effective, feature is introduced and its motivation explained. Furthermore, the created dataset is described in

detail. Afterwards the evaluation is discussed. The chapter concludes with a description of the used criteria and the evaluation results.

Finally, chapter 5 gives a summary of this thesis and discusses potential areas for future work to further improve usability of the framework as well as robustness of the ground detection system.

2. Related Work

This chapter discusses related work of the framework and ground detection. First, navigational aids for visually impaired people are presented in section 2.1. Then other frameworks that inspired this work are presented in section 2.2 and differentiated with respect to the created system. Finally, in section 2.3, several techniques for ground plane or obstacle detection are discussed.

2.1. Navigational Aids for Visually Impaired People

As argued by Olson and Robinson [OR12], "Humans often move and rotate faster and with more complex motions than robots, therefore requiring increased processing speed and robustness and the use of specialized algorithms."

There have been attempts to replace the walking stick with digital means. The GuideCane [SUB03], as shown in figure 2.1, replaces the walking stick with a digitally enhanced counterpart. It is similar to a normal cane, but has a two wheeled base and an array of distance sensors mounted to it. Obstacles in front of it, perceived close to the ground, are evaded. This is achieved by breaking the corresponding wheel, which generates an evasive maneuver by the person pushing it. The user is steered away from the obstacle in a circular motion until the obstacle is passed. In this regard, it acts similar to a guide dog. Another method uses sonar sensors and haptic feedback through small vibration units sewn into the wearer's garment and provides an unobtrusive and almost invisible way to signal feedback [CTV05].

To achieve real-time capabilities, some systems rely on the existence of specific markers or real world characteristics. This can be hard to do with a purely vision based approach due to hardware restrictions imposed by the mobile platform. For example Coughlan and Manduchi [CM09] rely on colored markers installed throughout a building that are detected by a mobile phone application to help location- and way-finding inside buildings. The idea is to improve location aware systems where the global positioning system (GPS) is not available. Furthermore, this system can directly provide location and routing information to the user. The main disadvantage of this system is that the markers must be provided and

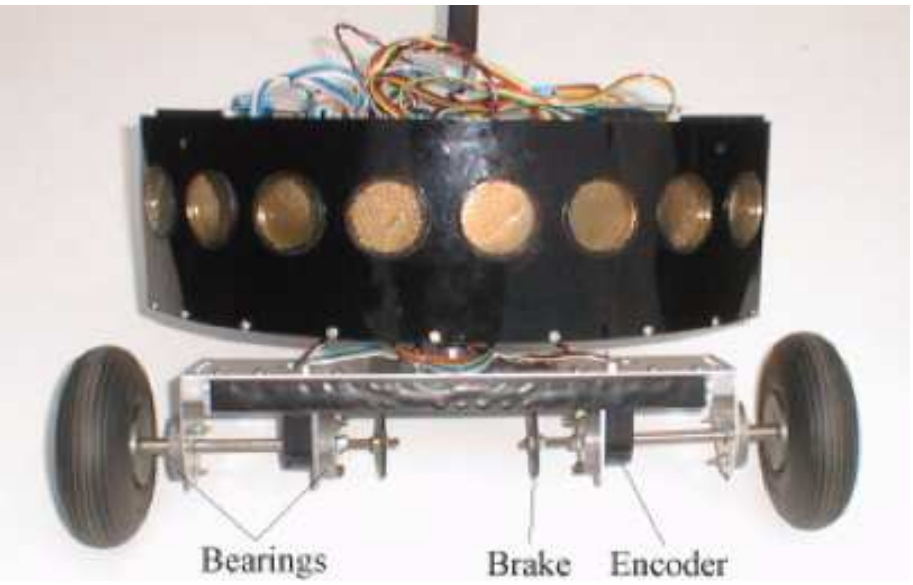


Figure 2.1.: The GuideCane housing and wheelbase. (Source: [SUB03])

the layout of a site must be known. Also, since its focus is clearly navigation, it does not consider moving obstacles, such as people. It should therefore be seen as another complementary aid to already existing techniques (such as the cane). The need for such markers is alleviated by Chen et al. [CFZ+12]. Here, an inertial measurement unit (IMU) is used to sample the user's kinematic data and thus its ego-motion. By combining information about walking direction, step length and frequency, a position estimation is created on an a priori known map.

A different approach is to focus on a specific subset of obstacles. Martinez and Ruiz [MR+08] warn of aerial obstacles such as branches or low hanging street signs. This complements the walking stick, since those aerial obstacles present a problem that is typically not sensed with a cane. They use a wearable stereo device and a laptop as show in figure 2.2. A system to detect staircases by Hoon et al. [HLM12] relies on a trained classifier. Here RANSAC is used to estimate the ground plane and remove false detections. Lee et al. [LSC12] use saliency maps and stereo vision to segment obstacles that have a high saliency. Objects with a similar color or structure to their background are problematic in this case. The existence of lane markers is depended upon in a system created by Le et al. [LPB12]. Color segmentation and intensity information are used for detection. In a probabilistic framework, multiple geometric cues are used for verification.

Since relying completely on computer vision might not yet be possible due to hardware constraints, some work has been done using specialized equipment, especially particular camera technologies. One of these is a time of flight (TOF) camera, used by Lee et al. [LSC12]. This specialized camera system already returns a disparity image. The returned image is segmented by detecting, as well as removing, edges and grouping the remaining depth layers into obstacles. A similar system [LM11] on the other hand uses a red-green-blue and depth (RGB-D) camera. This special camera not only returns a red-green-blue (RGB) image, but also a registered point cloud and thus compensates for

Figure 2.2.: A wearable stereo device for aerial obstacle detection. It is comprised of a *Bumblebee* stereo camera and a laptop. (Source: [MR+08])

the costly depth map calculation.

2.2. Frameworks

As stated by Sullivan, Griswold and Hallen [SGCH01], "... modularity in design creates value in the form of *real options* to improve a system by experimenting with new implementations and substituting any superior ones that are found." Such modularity cannot only help to improve a system by replacing its parts, it also divides development efforts more clearly among several developers, since it encourages a separation of responsibilities. To support modularity, a constantly growing number of frameworks have been developed, such as robot operating system (ROS) [QGC+09], Player 2.0 [CMG05], carnegie mellon navigation toolkit (CARMEN) [MRT03] or the distributed wearable augmented reality framework (DWARF) [BBK+01].

2.2.1. Robotic Operating System

One of the best known and most widely used frameworks is the robot operating system (ROS) [QGC+09]. It reuses many parts of other open-source projects such as Player [GVS+01, GVH03, CMG05], open source computer vision library (OpenCV) [Zel09] and OpenRAVE [DK08] and is itself free and open-source. It is specifically tailored towards a robotic scenario and thus includes specific tools suited for this particular task. Its design considers many problems and trade-offs regarding that setting, such as abstraction and reliability of services, but also robotic specific functionality like motor control.

The creators use a peer-to-peer protocol as a basis for communication among multiple entities. These entities can even spread over various computers on a heterogeneous network. Without implying that ROS is slow in any way, while its approach brings many

advantages, such as possible redundancy, fault tolerance and a distributed load, one can also argue that it induces a fair amount of overhead. It might be more feasible to assemble modular parts into a single execution unit and use multithreading to distribute work on a single machine, while keeping tight control over all contained modules. This could be especially useful when targeting a mobile platform, since mobile platforms tend to have less computing power and resources available. Also when moving towards a mobile platform, multi machine capabilities are most likely to not be needed, so a compact integration to decrease latency and overhead should be preferable as there is only a single central processing unit (CPU) available and no distribution over the network required. Therefore a monolithic design should be favored, as it reduces the overhead from communication among entities. To be extensible while still being monolithic, a functional and object oriented approach can be used, where each module is controlled by a master authority.

While at first a framework that has similar goals to ROS will likely seem redundant, the differences in design choices made during implementation of these ideas might largely only be perceived upon intense usage of said systems. The plethora of available tools, options and settings of ROS can be intimidating and represents an unwanted barrier to entry, so a smaller framework more suited towards a particular goal can greatly improve learning speed. This is especially true when for example an assignment is given to lab students. In such a scenario, using a framework is desired in order to increase productivity by reusing already available components. Furthermore, collaboration among individual groups is promoted through the use of a common base. One of the main goals of the created framework is thus ease of use, especially for students that do not have prior experience with computer vision systems and programming. Using a fully featured framework like ROS can be more of a hindrance under such specific circumstances.

2.2.2. Player 2.0

Player 2.0 [CMG05] is a widely adopted framework for robotic scenarios. It is an advancement of Player [GVS+01, GVH03], which uses transmission control protocol (TCP) sockets to create a server/client architecture. Besides sockets, Player 2.0 is also using inter-process communication (IPC) as a means to connect various software modules and libraries with one another.

Being a robotic framework, Player contains, just like ROS 2.2.1, many parts to control robot actuators and get information from its sensors. While different kinds of sensors are often used in a computer vision oriented framework, several capabilities are not needed in that case, like motor control. Although this server client module brings much flexibility, it also creates communication overhead that is best avoided when being on a limited platform.

2.2.3. Distributed Augmented Reality Framework

The distributed wearable augmented reality framework (DWARF) [BBK+01] is another system. Figure 2.3 shows an indoor scenario with a superimposed map of a building. It is a framework built to simplify the developing process of augmented reality (AR) applications. Similar to the augmented reality toolkit (ARToolKit) [Kat02] library, DWARF contains a component to deal with low level hardware issues, such as trackers. Furthermore, the DWARF framework concentrates on providing a complete world model, a task flow engine

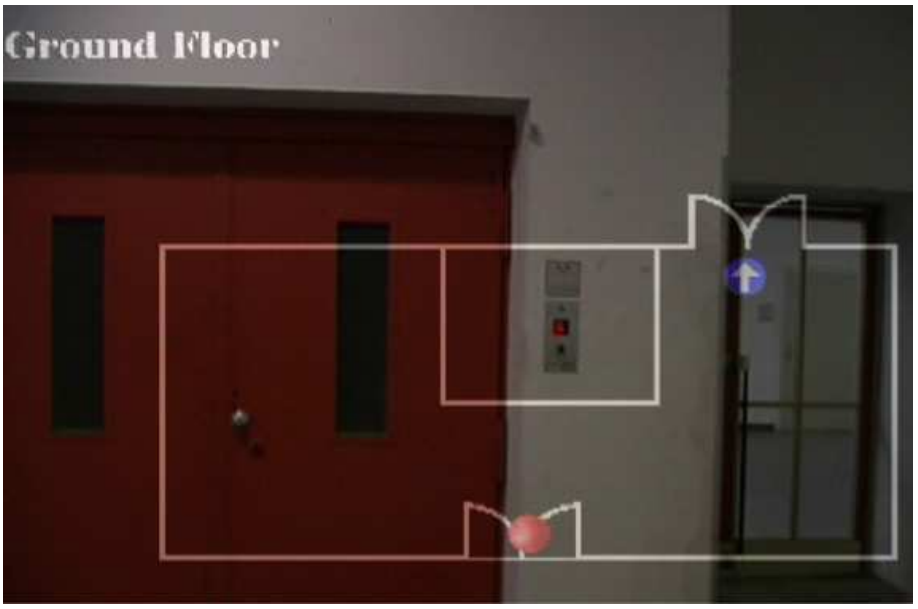


Figure 2.3.: An example application built with the Distributed Wearable Augmented Reality Framework. In this application, an indoor map is overlaid in to the users field of view. (Source: [BBK+01])

and a context aware service access. While DWARF might contain the needed parts to build an AR application, it is rather oversized for usage in a visually impaired person context. It seems preferable to decouple low level hardware and software issues from the framework and provide such capabilities as modules instead.

2.3. Ground Plane and Obstacle Detection

2.3.1. Plane Fitting

Se and Brady [SB02] use a linear relationship between image pixel coordinates and ground plane disparity. With a combination of a Sobel edge detector and RANSAC [FB81], they need to repeat the sampling 300 times to achieve a good ground plane estimation. One of their applications is pose estimation. With the recent availability of IMUs this can be done more efficient and accurate. They detect obstacles by marking features with disparities outside of the expected ground plane disparity range. Thus they often cannot discriminate between ground plane and smaller obstacles, or the lower parts of large obstacles. Furthermore, their ground plane disparity range estimation needs a constant reinitialization for camera pitch and roll. There is however the benefit of a reliable curb detection "if the curb step is sufficiently large" [SB02].

An iteratively reweighted least-squares (IRLS) [HW77] approach is used by Chumerin and Hulle [CVH08] to estimate a disparity plane. This approach has no additional information about possible ground plane configurations and can therefore yield undesirable results when the ground plane is not the dominant planar structure in the scene. In order to prevent this, a fixed set of nine image points (3x3 lattice) as well as two stabilization points are used for the calculation. The stabilization points are computed by using knowledge

(i) (ii)

Figure 2.4.: Close Range Human Detection for Head Mounted Cameras. The left image (i) shows the setup which consists of a *Bumblebee* camera mounted to a helmet. In the right image (ii) detections of humans in a pedestrian area can be seen. (Source: [ML12])

about the camera position inside a moving vehicle. Adding these points to a wearable and therefore highly versatile platform seems hardly possible. The absence of appearance models allow this system to deal with a wider range of scenarios, since it does not rely on specific assumptions about the scene or the existence of for example lane markings. Due to the usage of only a small subset of the available disparity map, the quality of the ground plane detection depends greatly on the quality of the disparity map in these exact image regions. Thus it can lead to incorrectly identified ground planes. An approach that uses all or even as many as possible points of a disparity map should be insusceptible to such behavior.

2.3.2. Segmentation

Segmentation techniques for ground plane and obstacle recognition are used by Lombardi [LZM05]. Instead of using two-dimensional segmentation techniques, three-dimensional segmentation with a disparity map is used. First, road surface candidate pixels are selected by using bottom-up techniques. These candidates then vote for road models from a predefined set of appearance models. Using the winning model, all road surface candidates within this model are selected. This represents the top-down part of the algorithm. Finally, an optional fine boundary segmentation can be included to further refine the detected road. Obstacle detection can be performed by additional investigation of road disparity regions that do not fit into the winning model. The appearance models depend heavily on the camera position with respect to the ground as well as the existence of a narrow or wide strait (in this case road surface).

2.3.3. Obstacles

While most of the works from section 2.3 already use a correlation between the presence of ground plane and obstacles, there also exists an array of work focusing on obstacle detection. Many of those deal with automobile settings [LAT02, BPCL06]. They use varying techniques, such as the Hough Transformation [DH72], to detect prominent or salient objects as well as optical flow [HS81].

A quite different approach is used in an obstacle avoidance system for a rotorcraft unmanned aerial vehicle (UAV) [Hra08]. Through the use of special equipment, such as stereo on chip (STOC) technology and an embedded computer, real-time capabilities are achieved. A sophisticated framework for probabilistic map occupancy is then used to generate a point cloud representing the surrounding obstacles. This technique is limited by "the amount of memory (such a map) can require" [Hra08], since it becomes very big for rather small voxel resolutions. Furthermore, a probabilistic roadmap planning as well as a stereo-based replanning is calculated using techniques such as A* or, more efficiently, D* Lite.

A great many works deal with pedestrian detection in urban settings, some even on a wearable platform [ML12]. Here, depth information is gathered through the use of a *Bumblebee* camera, which is attached to a helmet as shown in figure 2.4 (i). Then depth templates of upper bodies are learned and matched to the disparity image. This works really well, even for crowded scenes as can be seen in figure 2.4 (ii). In a similar system stereo camera rigs mounted on wheeled vehicles are used [ELSVG09, ESLVG10]. This results in a rather steady camera movement with only very few pitch and roll changes. Here a probabilistic model is used through a Bayesian network, which models the dependency of person detection, detection size and location, and by using all detections creates a common ground plane estimation. This model is then updated in each frame, so it can deal with static as well as moving obstacles.

3. The Blind and Visually Impaired Support System

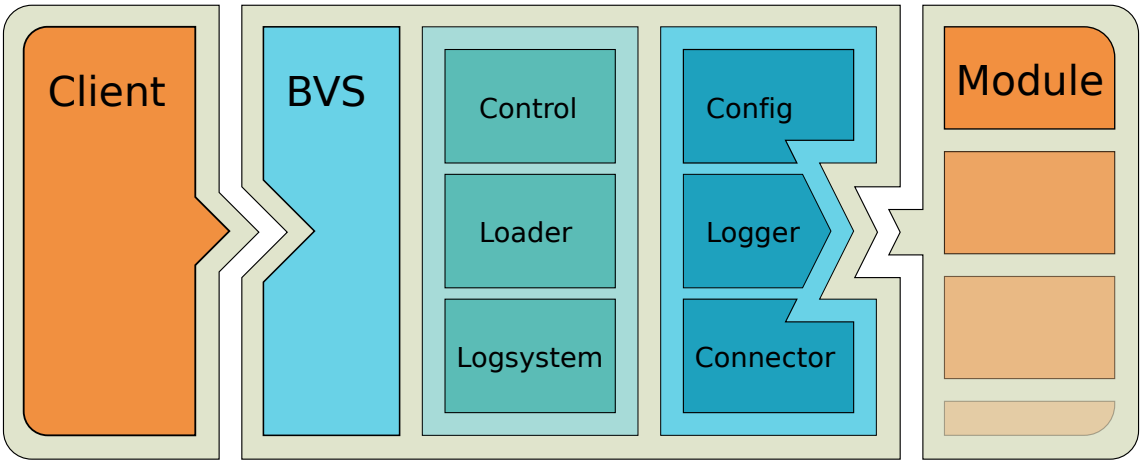


Figure 3.1.: An overview of the blind and visually impaired support system (BVS) framework. Additional system parts are shown in appendix A.

This chapter gives an overview of the created framework presented in this thesis. Its main parts, public and private interfaces, are discussed and some design decisions are explained. Then, the modules created to aid in computer vision tasks are presented in detail.

3.1. Main Design Goals

The blind and visually impaired support system (BVS) has been created to simplify collaborative research efforts towards visually impaired users. It was designed to be fast to learn and as easy to use as possible with a setup time of less than a few hours. In order to not impose special demands onto the user, the framework itself is completely free of any special purpose parts created towards a particular research domain. Such functionality is

outsourced into modules, so for every use case the decision to include these can be made upon the specific needs and requirements of a project. Thus the framework itself is kept as abstract as possible.

One of the reasons to create this framework is to encourage code reuse instead of code duplication. Often, specific code is written to deal with a certain library, driver or data conversion. This code is often hard to separate from other aspects of an application since it will most likely be deeply interwoven with other application parts. Therefore, reuse is made much harder than it would be in a strictly separated environment. Also, bug fixing efforts must be duplicated, because a bug found in one application does not translate well into another application using parts of the same code base. To counter this development, it has long been proposed and successfully deployed to group functionality into classes, objects or functions. The notion of modularity represents nothing different, but goes a step further. With separate objects or classes, there are no specifications about common data structures or caller formats. On the contrary, within a modular approach, policies and procedures to share and use data are fundamental attributes. Every module part of a larger system has to obey these rules. Thus modularity presents a stronger argument towards reusability, connectivity and interchangeability.

To reduce overhead created by the modular framework, a functional approach has been taken. Each module consists of an object with a core functionality. This functionality represents its main contribution, while the rest of the module deals with startup and shutdown of the module, for example initializing and shutting down hardware systems. Oftentimes, additional information or functionality is necessary and must be provided outside the core functionality. By separating support and core functionality, the concept of a pure function is approached. Modules can be connected, just like functionality would be chained, in a program to request a certain outcome. Using the framework, this can be done dynamically. The framework will call their core functionality in the order they have been specified, while all the side effects of data passing and data conversion have already been taken care of. The inflicted overhead of the framework can be made negligible by using a flat calling hierarchy. Eventually, the framework approximates the running costs of a specifically created application which simply calls several functions, but with the added advantage of being dynamically changeable, even at runtime.

The framework itself is completely encapsulated inside a library object, also known as ”shared objects” or ”dynamic link libraries” [BWC01]. While this contradicts the common notion that a library is not a framework, it allows for the creation of a two sided interface, one for clients that desire to control the framework, the other one for framework modules. Both interfaces provide the specific parts needed and are described in the following sections. Since the framework allows for easy creation of additional clients, it is a straightforward process to include this framework into others.

Documentation is an important step towards creating an easy to use framework. Thus, the popular *Doxygen* [vH08] documentation system is used throughout the framework and its modules. This allows for a comfortable generation of the documentation as well as the ability to supply pre-generated documentation bundled with the framework library.

The, at the time of this writing, new *C++11* standard is used extensively throughout the framework. It greatly helps with programming several aspects of the framework, especially

multi threading. Although the multi threading support in *C++11* is far from complete, it supports system portability as well as code simplicity. As a build system, cross-platform make (CMake) [MH08] is used to simplify the build process of the framework and its modules. It allows for an easy to use and understandable build structure. Furthermore, each module can be bundled with its own compiling directives, which separates the process from the framework as well as other modules. Thus, it alleviates the treatment of dependencies, even inter-module dependencies. To manage the framework's source code as well as the source code of created modules, *git* [TH05] is used. It allows for a distributed version control and is used widely in the industry as well as the open source software scene, amongst others the Linux kernel development. Moreover, *git* has become another important tool to promote the sharing of resources with other researchers or research students.

3.2. Public Interfaces

3.2.1. Client Interface

The client interface provides the basic functionality for the entire BVS framework. It allows for a dynamic change of all existing framework traits at startup as well as any time during usage. Such traits are for example a logging system, which will be described in detail as part of the module interface in section 3.2.2. Furthermore, multi-threading and module pooling can be enabled or disabled. The major reason to allow explicitly the removal of multi-threading is debugging. In a debugging scenario, it is often preferable to have all modules running inside a single thread, since not all debugger handle multi-threaded scenarios rather well. Also, it allows the debugging process to concentrate on issues that are not related to the nature of simultaneous execution in multi-threaded scenarios. As such, it cannot help with issues that stem from multi-threaded synchronization issues.

A framework client can, after the creation of a framework instance, load modules individually or use the supplied configuration mechanism (section 3.2.4) to load a predefined number of modules as well as framework settings. This mechanism allows for a fast creation of systems that can be loaded simply by specifying a configuration file to the client. Furthermore, it is easy to create for example demo applications using this technique. An added advantage in such a scenario is that while the number as well as the order of components are fixed, individual components benefit from bug fixing and feature improvements.

Of course, the client interface contains methods for the client to start, run, step, pause and quit the frameworks operation. A step, also referred to as a round, is considered to be a single execution of all loaded modules. All modules are run synchronized by the framework, executed in the order they are specified or running in parallel in a highly multi-threaded scenario. Synchronization is achieved by using a barrier, a synchronization method that regulates concurrency among a fixed number of threads by enforcing rendezvous points. So during each round, it is guaranteed that each module of a system will run exactly once.

An asynchronous control mechanism has been considered, but it was found that it is preferable to leave unused processing power to busy modules, instead of having all modules constantly poll, therefore actively competing for resources, and thus occupying all available CPU cores. The major drawback of having all modules run in a synchronized fashion is overall round time as well as system delay. The overall round time is the time that it

takes for all modules to finish one round. The system delay, however, is the accumulated time it takes for information to pass through the entire system, that is the entire chain of modules from the first to the last one. Thus one slow module, most likely processor or input/output dependent, can dramatically increase the round time and thereby the overall system delay. In order to alleviate the effects of such a behavior, statistics such as the collection of individual module and pool runtimes during each round have been integrated into the framework to detect such ill manners. A module pooling mechanism has been integrated to join the execution of other modules into single threads, thus decreasing the overall round time, see section 3.3.2. Since this mechanism enables multiple modules to be executed using current round information from their predecessors, the system delay can be reduced by several round times, depending on how many modules can be bundled into such a pool.

Given that all modules belong to the same system process, unlike with ROS, passing data between modules has been implemented by using shared memory instead of message passing or IPC. This guarantees a low overhead, as connections must only be established once and is especially advantageous for large amounts of data, such as images. More information about this mechanism can be found in the section about module connectors, section 3.2.3.

The framework has been designed to make heavy use of the *Pimpl Idiom* [Sut09] wherever possible, which is also known as a "compilation firewall." Its purpose is to hide the specifics of an implementation behind a public interface. This allows programmers to change the inner workings of a system, without exposing these changes to the outside world. Any system depending on the framework does not need to be recompiled upon framework implementation changes, since those are not reflected in the public interface. Therefore, the usage of the *Pimpl Idiom* allows for later internal changes of the framework without affecting already used instances.

As stated above in section 3.1, the purpose of having this client interface is the ability to encapsulate the framework into a library object. This allows for greater portability as well as easier integration within other systems. However, this leads to a minor problem, which is that a framework that needs to be controlled by a client cannot run on itself. That is why a daemon is provided with the framework. It is presented in the next section.

Daemon

A daemon has been created that serves two purposes. First, it can control the framework in headless environments, meaning it runs on the command line. Second, it serves as an example of how to use the client interface. Future work is planned for a graphical client, which should also serve as a debugging tool.

The daemon starts an interactive shell through which the user interacts with the framework. In addition, this shell can be used to output the systems log. These are not framework features, but belong to the daemon implementation. Giving the user interactive control to the framework is perhaps the most important aspect of the development process. Only through this access modules can easily be interchanged at runtime. A framework client or daemon could also be written to simply run a specific system, which could be created through the use of the configuration mechanism, or implemented inside the daemon's source code.

Furthermore, the daemon allows to overwrite any kind of configuration option on the command line. This permits the usage of the daemon inside a fully scripted scenario, where any property of the system can be changed upon each execution of the framework. Especially for system and algorithm optimization this represents an invaluable tool.

3.2.2. Module Interface

The second interface is aimed towards the integration of framework modules, it must be conformed to in order to allow the framework proper integration. To promote a consistent module interface, a helper tool has been created that creates an empty template for the newly created module. This frame can then be extended to include the desired functionality.

Such an approach allows for easy, partly automated and, therefore, fast creation of library wrappers. These wrappers are completely self-contained, they only define their desired input channels and provided output channels as well. Once the library functionality has been added to the module, this library can then be reused in many different projects. Instead of having custom written code each time an external library is used, this process abstracts the library and wraps it into an easily reusable container.

While modules are controlled completely by the framework, they are still allowed to make full use of all system resources, for example starting additional threads. Such behavior should always be considered carefully, since especially in high speed scenarios the additional overhead caused through the additional synchronization layer is not negligible.

There is however one important ability for modules: they can request a system shutdown if required. This can happen when for example an input module has no more data to process or when faulty states are detected inside a module. The framework will then not immediately shut itself down, but rather keep the system running for a while to give all other modules that belong to an active system a chance to finish processing their input and properly shutdown for example hardware components. Additionally, modules can inform the framework of various module states. One such state could be that a module is repeatedly not receiving any input. While the framework itself cannot handle such situations yet, a client could, in an interactive setting, act upon it.

Sometimes it is needed to store various configurations of an algorithm, which is itself represented as part of the framework through a module. The configuration subsystem, described in section 3.2.4, allows to store these settings inside a more dynamic environment. They can be bundled into module configurations, which can then easily be accessed through the frameworks module configuration syntax, shown in section 3.2.4. This allows for a fast switch between several already created module configurations.

3.2.3. Module Connectors

Module connectors define the way for a module to acquire input from other modules as well as send output to others. The connector system has been built to allow for a high speed message exchange with negligible overhead. Such connections are initialized by the framework upon startup. Afterwards, a sending entity can put its data into a shared memory region. This region is automatically announced to the receiving entity. To prevent

concurrent access, a locking mechanism is included. This ensures the integrity of data in multi threaded scenarios by excluding race conditions and other synchronization issues.

The framework has been built to be completely agnostic to the type of passed data inside such connectors. To it, all passed data are just memory regions. The main reason for this design decision is to keep the framework free from application specific code. This however, has the disadvantage that each module must check the validity of its input before processing it further to avoid crashes resulting from incomplete or corrupted data, which is widely considered a *good behavior* anyway. On the software side, through the use of metaprogramming, a connector is type safe, feels and handles just like a native object, so it is easy to use.

In order to provide at least a bit of support with type checking, the framework can rely on run-time type information (RTTI) to detect incorrectly setup connectors during the startup phase. This can be explicitly disabled if desired, since not all compilers might be able to include such information in the resulting binary. The produced overhead of this mechanism is negligible, as it only affects the startup time.

3.2.4. Configuration Subsystem

The configuration subsystem, named simply *Config*, serves an important purpose while developing a system. It allows for an easy change of system settings and configurations. With the configuration mechanism, the user can create or change all system aspects, without having to recompile. Furthermore, having a configuration that can be saved and passed around to other researchers enables easy sharing of system implementations, instantiation, algorithm parametrization and their results.

As with any configuration mechanism, it demands a specific syntax to be followed. This syntax has been kept simple, while still allowing for advanced features, such as array specifications inside a single line, as shown in figure 3.2. As can be seen in the example, through the use of metaprogramming, the configuration system converts desired option values into the demanded formats. Thanks to this mechanism, the user does not have to take care of any form of configuration parameter conversion. This mechanism can even be extended to user created types.

In order to separate module configurations from the base framework settings, a sourcing mechanism has also been included. This allows to distribute a module's configuration bundled with the module code, while still enabling an easy integration into an existing system by simply sourcing its configuration options.

Module Configuration Syntax

To support modules that assemble a broader array of functionality or support different modus operandi, a special module syntax is used by the framework as shown in figure 3.3. This syntax helps in the definition of modules using various of its possible capabilities by allowing to select a module with a certain configuration. Furthermore, by decoupling the name or identification of a module from its origin, the underlying implementation of a module instance can easily be switched by another implementation without causing further problems down the module graph, especially when connecting the module to other

```
# everything after '#' is considered a comment
option = ignored                        # must belong to a section

# sections and options are case insensitive
[section]                               # same as [SeCtIoN]
option1 = value1                        # comment
Option1 = value1                        # warning, redefinition

# spaces are stripped if not inside quotes
option2 = "value2 with spaces"          # single quotes also possible
option3=value
option4  =  value
 option5     =value
# all of the above are valid entries

booleanOption = true
# for boolean type option-value pairs:
# '1', 'true', 'True', 'TRUE',
# 'on', 'On', 'ON',
# 'yes', 'Yes' or 'YES'
# are interpreted as TRUE, everything else as FALSE

stringList = elementOne,"element Two",'element Three'
booleanList = true, false, True
# accessed by using getValue with std::vector<TYPE>

# the plus-equal operator can be used to expand existing options
# this turns them into a list, allowing for faster reordering
list = one
list += two
list += three
#list += four
list += five
# the result of this will be: list = one,two,three,five

# other config files can be sourced (included)
source OtherConfigFile.txt
```

Figure 3.2.: Configuration subsystem syntax.

```
# modules [+]= [+|poolName]id[(library[.config])][.connections]
#
# [+] -> Append to the module list.
# [+|poolName] -> Load module inside its own pool ('+') or
#                 add/create to a module pool of name 'poolName'
#                 which also runs inside its own thread and executes
#                 added modules in the given order.
#                 NOTE: '+' is effectively a shorthand for '[id]id'.
#                 NOTE: 'poolName' MUST be inside '[...]', '+' not.
# [(library...)] -> Use as module library to actually load
#                 module from, useful for multiple modules from a
#                 single library.
# [.config] -> Use this configuration for the module, useful so
#                 the module name does not change but its
#                 configuration can be changed easily.
# [.connections] -> Options for connectors, looks as follows:
#                 input(test.output)[.input2(test.output2)]...
#
# If configuration and/or library are not given, the system will
# use the given id instead (useful as a shorthand).
#
# Examples:
# < SHORTHAND >                          < VERBOSE >
# id                                      id(id.id)
# id(lib)                                 id(lib.id)
# +id2(lib2)                              [id2]id2(lib2.id2)
# +id3(lib2.conf).in(id2.out)    [id3]id3(lib2.conf).in(id2.out)
# [pool]id4.in(id3.out)                   [pool]id4(id4.id4).in(id3.out)
```

Figure 3.3.: Module configuration syntax.

modules. Without this mechanism, all appearances of the original module identification would have to be changed as well if an implementation was switched. An example of how to use the module configuration syntax to create and connect various modules is shown in figure 4.1.

3.2.5. Logging Subsystem

Having the ability to output any data at any given point during a programs execution time can be a major factor in diagnosing problems early on. As such, a logging system should be powerful and easy to use but still lightweight.

In order to simplify the logging mechanism for an user, a logger instance is created, which is then used to interact with the logging backend. This object contains all the users settings for its logger instance, such as the instance's name and verbosity level. Furthermore, it can be selected, whether the instance should log to a console only or also to a file. The

0 Errors that are likely to cause system failure.

1 Warning messages to the user that problems occur.

2 Info displays progress information.

3 Debug messages, this level is used to output internal information.

4 Level 4 and above are user defined levels.

Figure 3.4.: An overview of the recommend logging levels.

separation of the log system backend and its interface through logger instances allows a for a complete modification of the logging system without relying on changes to client code.

Various log levels are supported by the framework, show in figure 3.4. This separation of logger contents helps with parsing relevant information.

This design approach has been chosen to give the user flexibility when using the logging subsystem. However, the logging backend reserves itself the ability to override all logger instances verbosity levels as well as targets. This way a framework client can selectively parse logging information without being interrupted by dispensable information provided by other module instances.

To handle highly multi threaded applications correctly, an additional locking mechanism is used to prevent the output messages from interleaving each other. Also, logging can be completely disabled at compile time, thus reducing any overhead created by it in the first place.

3.2.6. System Information

The framework's information system can be used by modules to acquire various data of the framework. These informations include for example access to the frameworks configuration mechanism, so that modules with different configurations can be loaded. Furthermore, round and statistic information, such as individual module and pool runtimes, can be acquired. It cannot be used to pass information between modules or to the framework itself, as there are other methods included to do that.

3.3. Private Interfaces

While so far the focus has been upon the public interfaces for framework clients and modules, some of the most important private interfaces are discussed in the following.

3.3.1. Loader

Perhaps the most important aspect when creating an extensible framework is the ability to load extensions. Hence the frameworks loading mechanism has been designed to handle the operating system specific tasks of integrating precompiled code into the system at

runtime. The loader abstracts the rather arcane operating system functions to handle dynamic libraries into an easy to use interface. It handles the loading of dynamic library objects, the creation of module instances from these and it registers the modules within the framework. Of course, it also does the opposite operations when closing a module that is no longer needed. While the integration of modules seems trivial, it strongly depends upon the used language and its implementation. At the time of writing, *C++* did not have any form of module support. That is why the frameworks relies on old *C* interfaces to achieve the same functionality since these are platform specific. Concentrating this functionality into a single place decreases the overhead needed when porting to a different platform is desired.

Working with a modular system can bring a lot of benefits, such as the ability to reload a module because it is either misbehaving, has been corrupted or there is a newer version of it available. The latter happens especially often during the development cycle of a system. When reloading a module through traditional means, it loses all internal state information, unless these were saved inside an external structure beforehand and, therefore, supported by the module. In order to support a true module reloading experience, a hot swapping capability has been added to the loading mechanism.

Module Hot Swapping

To substantially decrease the time of a development cycle, a hot swapping mechanism has been included in the framework. Instead of relying on a module to lose all its internal state or save said state in an external structure, the occupied memory of a module instance is reused when recreating the updated module instance. While this technique is neither desirable nor standard conform, it has proven itself to be reliable. Instead of deleting a module and thus erasing all of its memory, the module instance is kept in memory while the old library version is unloaded. Then, a newer version of the library is loaded into the system and combined with the old instance's memory. However, this can lead to all sorts of problems and crashes and should therefore only be used inside a development scenario. Even then, it has its limitations due to the way objects are laid out in memory. Except for a hot swapping mechanism that requires special support, there is nothing that can be done to prevent it. Such an operating procedure is even considered "undefined behavior" by the *C++* standard [Str93] and should thus be used with caution.

Although the inherent insecurity of this mechanism cannot be eliminated, it has proven itself really valuable in various development situations. Especially when relying on hardware that has a long warm up and cool down phase - for example some camera systems - this reduces the iteration cycle by a fair amount of time.

3.3.2. Control

The next important part of a framework is the ability to gain full control over all modules. In the BVS framework this is implemented in the control system.

The control system supports a messaging mechanism. It can be started as a separate control process, which runs inside its own thread. That way, a client could offer an interactive shell that connects to the control system and signals the desired actions to the control system. Furthermore, the control system can be used directly by a client through

the public interface, but this has the disadvantage that every round needs to be triggered manually.

A round is considered a full cycle, that is an execution of all involved modules from start to finish. The round time is limited by the slowest module since all modules run in a synchronized mode. The reason for this is mainly to not lose any data due to overzealous modules and to free processing capabilities.

A barrier is used to synchronize the individual modules. Such a barrier is a well known thread synchronization primitive used in many operating systems. *C++* does so far not include a barrier mechanism, while it has gained a lot of support for multi threading in its latest incarnation *C++11*. Furthermore, most barrier implementations only consider a static barrier, where the number of participants never changes. Since the frameworks intended use case requires it to be as dynamic as possible, such a static barrier cannot be used in this context. Thus a custom barrier had to be written that allows for a dynamic adjustment of the participating parties, in this case module pool threads. The pool abstraction is explained in the following section.

Pools

Pools are an abstraction mechanism of the framework to simplify dealing with individual modules or module groups. A module pool consists of at least one or more modules. If a pool does not contain any more modules, it takes care of its own proper destruction. Modules can be added or removed from a pool dynamically, i.e., during runtime, without affecting other modules, except when disintegrating parts of their input sources.

The major reason to introduce module pools is the ability to group small and fast modules together. As explained in section 3.2.1, this can significantly reduce the overall system delay when a single module has a large execution time. Thus a higher throughput can be achieved, which equals a higher frame rate. While at first, there were different control structures depending on whether a module was to be run in parallel or not, module pools greatly simplified the underlying architecture.

3.4. Base Modules

The base framework itself contains no specific low level drivers. To support development of systems aimed towards visually impaired people, some needed functionality has been encapsulated inside basic modules. These modules are provided in combination with the framework and can thus, while they are technically not required to run it, be considered a part of it. They also serve the purpose of being exemplary to the creation of further modules.

3.4.1. Capture Module

The capture module has been created to simplify the process of retrieving input from one camera or if necessary camera arrays. This is a fundamental step in computer vision applications. OpenCV [Zel09] is relied upon to deal with hardware issues, such as driving different camera gear and retrieving images over various connection types. The number of desired cameras, also referred to as nodes, can be set upon startup in a configuration file.

During usage of the framework it became apparent there would be a need to store captured data and retrieve it later on. Thus, the capture module has been extended. It can store individual frames as images or videos as well as retrieve recorded data from these. This allows for a complete replay of recorded data. Such a mechanism can be used for example to evaluate algorithms or a dataset.

3.4.2. Calibration Module

When working with a stereo camera system, calibration is an important step and this module has been conceived as a result. It is designed in such ways that it supports intrinsic calibration of multiple cameras as well as extrinsic calibration of stereo camera systems, while it could also be extended to calibrate an entire array of cameras. It serves as an abstraction layer to the OpenCV [Zel09] calibration methods. It connects to the capture module to do a live calibration but it can also calibrate a camera from already saved image series.

To support live calibration, an auto shot mode is integrated into the module. In this mode the calibration module waits until the calibration pattern can be reliably detected in all cameras. Then snapshots are taken which are in return used later for further processing. In order to increase the calibration quality, which is an important step in retrieving good results from stereo algorithms, a calibration guide has been designed. This optional guide tries to ensure that pattern detections are evenly spread all over the image in such a way that the calibration process does not overfit towards a particular region of the image, while neglecting others.

After a successful calibration, the results can be extracted from a calibration file for further use and the input images can be extracted from the calibration module in a rectified condition. Also, after a successful calibration, the results will automatically be reused during the following executions.

3.4.3. Stereo Vision Module

To test the speed and reliability of a few stereo algorithms, a separate module has been created. This module uses the graphics processing unit (GPU) and OpenCV [Zel09] to speedup the stereo reconstruction process. It can dynamically switch between a block matching algorithm, a belief propagation and a constant space belief propagation. The module was used in early stages to test the correctness of the capture system, especially the calibration module.

3.5. Additional Tools

To simplify framework usage, some additional tools have been included. When an user first downloads the framework, a setup function can be run. This will install the desired modules that are already known to the framework, but this process can also be expanded to include repositories of other origins.

Furthermore, an update function is provided. Since the framework supports modules from varying sources, it can become quite cumbersome to retrieve updates from a lot of different repositories and keep all in sync. That is why the update function will look for installed

repositories and update these accordingly. This can even be done without losing local information. Sometimes it might be necessary to do the opposite of a framework update, a version rollback. For this case, there also exists a helper function.

Also, to ensure module consistency, a module creation script is included. This has already been described to some detail in section 3.2.2. All of the above capabilities are integrated into the main utility script located in the base of the repository.

4. The Ground Detection System

After creating the framework a ground detection mechanism has been developed. Section 4.1 discusses the design decisions, some of its components and the underlying algorithm. An introduction into the created dataset comes afterwards in section 4.2. In section 4.3 follows a description of the experimental setup as well as the used measurements. Finally, the results are presented in section 4.4.

4.1. Design

The ground detection system is built using the created framework that is described in chapter 3. Moreover the supplied modules detailed in section 3.4 are used. These base modules are used to reduce the effort of building a functional prototype as well as to test their functionality. Furthermore they provide much of the needed capabilities, such as input and output of video or image data and calibration of the used stereo camera system.

Since the entire framework is built upon the assumption of a modular system, the actual ground detection algorithm is encapsulated inside a module. This way it can be easily used with the created framework and reuse in different applications or usage scenarios is greatly simplified. Encapsulation of the ground detection system furthermore enables the use of sophisticated framework support tools. For example the hot swapping mechanism described in section 3.3.1 helped to reduce the development time of the algorithm by a fair amount. Due to the modular nature of the framework, changing the input to the ground detection module is quite simple.

In the following sections, a few parts of the ground detection system will be further explained. This starts with a general system overview. Afterwards, a short description of the used stereo reconstruction algorithm follows. It is followed by an explanation of the algorithm and its main feature, the gradient detection.

4.1.1. System Description

The ground detection system consists of various components, such as the base modules described in section 3.4 and the actual ground detection. Furthermore an additional stereo re-

```
### INPUT
modules = cap(CaptureCV.video)

### PROCESSING
modules += calib(CalibrationCV).in1(cap.out1).in2(cap.out2)
modules += elas(StereoELAS).inL(calib.out1).inR(calib.out2)
modules += ground(GroundDetector.GroundDetector).in(elas.outL)

### OUTPUT
modules += vid(CaptureCV.video_writer).in1(ground.out1)
```

Figure 4.1.: A framework configuration example. Here, a video is being processed and the calculated ground plane information is saved into another video file. This example shows the convenience of a modular framework. With a few configuration lines, a complex system can be realized. It should be noted that in this example multi threading is not used, but doing so requires only a simple operation, adding a '+' in front of each module instantiation.

construction module has been created to encapsulate the efficient large-scale stereo matching (ELAS) stereo library that yields better results than the OpenCV implementations.

The capture module, discussed in section 3.4.1, is used to read parts of the dataset into the system. This can be done in a video format, as individual images or from a live camera system. Then the calibration module from section 3.4.2 uses calibration information to rectify captured images. The calibration information is obtained beforehand by doing a calibration whenever the stereo camera system is used. Even marginal alterations within stereo camera system result in a change of the stereo base and thus render older calibration information useless. The rectification process is necessary to achieve good stereo reconstruction results. Without these, the proposed algorithm does not work properly. After the images are rectified, they are passed on to the stereo reconstruction module.

Since the created base stereo reconstruction module of the framework 3.4.3 does not deliver well enough results to create a proper ground detection, an additional stereo module has been created. This module uses the ELAS [GRU11] system that is explained further in section 4.1.2.

The configuration layout of the system is shown in figure 4.1. More information about the configuration system and the module syntax can be found in section 3.2.4.

4.1.2. Efficient Large-scale Stereo Matching

As stated in section 4.1.1, the stereo reconstruction module delivered with the framework does not produce depth information of a sufficient quality. Thus another module has been created that uses the efficient large-scale stereo matching (ELAS) reconstruction method [GRU11]. Some results of the ELAS algorithm are shown in figure 4.2.

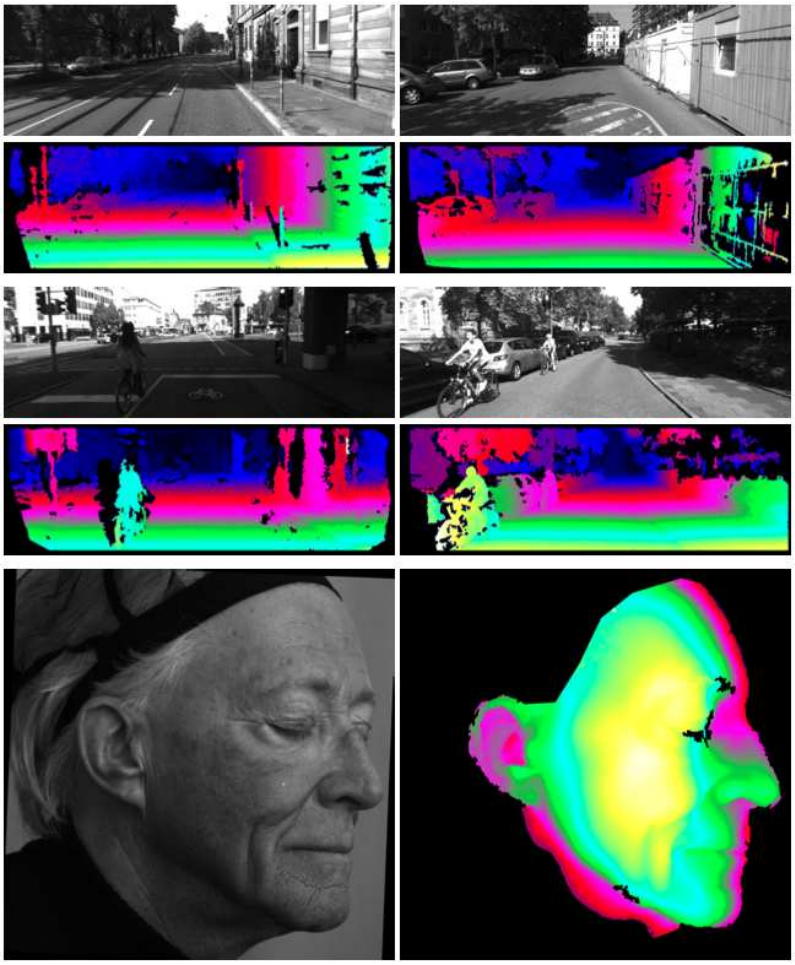


Figure 4.2.: Results of the ELAS stereo reconstruction on an urban video sequence and a face image. (Source: [GRU11])

The ELAS system uses a dense reconstruction algorithm that does not require global optimization. First, support points are calculated on rectified stereo images using a Sobel operator. Then a Delaunay [Del34] triangulation is calculated to create a two dimensional mesh. This mesh improves the disparity search by narrowing the available search space for each point. By using a local approach supported by a globally computed prior depending on the Delaunay mesh, even image regions with sparse texture information can be reconstructed. Final post processing is performed in order to fill small holes and a consistency check can be done for left and right disparity maps. Although the library's processing time is slow for normal sized images, its stereo reconstructions are excellent, even under quite difficult conditions.

4.1.3. Gradient Calculation

While there already exist quite a few techniques to detect ground plane as discussed in chapter 2, each of these have a varying degree of dependencies and requirements towards the observed scene. In order to create a ground plane detection that handles a large degree

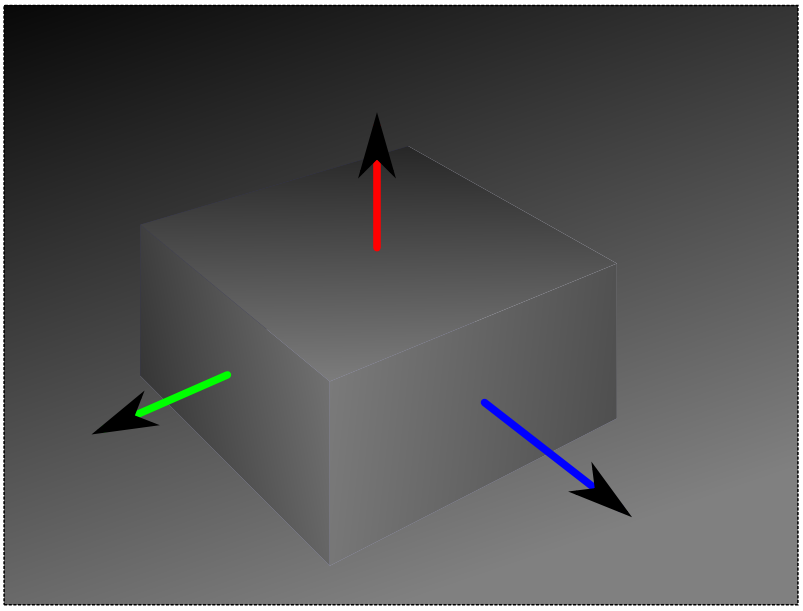


Figure 4.3.: This disparity map mockup shows the relationship of a region's gradient and its corresponding surface normal. The obstacle's gradients are exaggerated to increase visibility, real life gradients in stereo reconstruction maps are much more subtle, but can still be detected.

of ego-motion from the used camera system, a rather simple approach that does not rely on the external setting was chosen.

A direct connection can be made between the local orientation of a segment in the real world and its representation in the disparity image. Real world points with a greater distance have a smaller disparity than points that are closer to the camera system. Figure 4.3 displays this relationship.

The gradient of a subsection of the disparity map can be calculated as shown in equation 4.1.

$$\nabla f = \frac{\delta f}{\delta x}\hat{x} + \frac{\delta f}{\delta y}\hat{y} \tag{4.1}$$

A discrete version of the ∇ operator used in image processing is to use a filter kernel. Such a filter kernel works on a small image region and is computed for every pixel. The kernels have to be separated for the ∇X and ∇Y direction and are shown in equations 4.2 and 4.3, where $f(X)$ and $f(Y)$ represent the region around X and Y of the appropriate size.

$$\nabla X = \begin{bmatrix} -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \end{bmatrix} f(X) \tag{4.2}$$

$$\nabla Y = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} f(Y) \tag{4.3}$$

Different kernel sizes yield different results. In this system, a kernel size of $32x32$ pixels has been chosen. This computation would be costly for every image pixel and thus is only performed block wise. While in this work, only a single kernel size is computed, further scaling of the kernel should be done in order to account for different input data as well as computation of the gradient for every image pixel. Furthermore, a quad-tree structure can be used to concentrate the computational efforts on specific regions of interest.

From ∇X and ∇Y the orientation of a surface region represented by the processed disparity map region can be calculated as shown in equation 4.4.

$$\theta = \arctan \frac{\nabla Y}{\nabla X} \tag{4.4}$$

While this calculation seems to consider only two dimensions, it also works in the three dimensional case, since we are only interested in real world surfaces that are upright. In these cases, the calculated value of θ will be close to $\pi/2$, or $90°$.

Although the calculation of the surface orientation seems simplistic, it works quite well. Furthermore, such a simple approach consisting of two filter kernels as well as an orientation calculation can be computed efficiently and at high frame-rates.

4.1.4. Accessible Section

The accessible section is determined by adhering to geometric constraints imposed by the real world setting. After determining the complete ground section, the accessible section of it is the part that can be directly reached from the current camera position without conflicting with any obstacles. Since the part of the camera image that is closest to the view point is the bottom image border, the accessible section must be connected to it.

As a criterion the calculated angle is used and a deviation of $\pi/8$ is allowed. Thus a simple upwards search from the bottom towards the top border yields the accessible section by checking all blocks that fit this criterion. While it was considered to allow the detection process to "peek around corners," it has not been implemented. The detection of the accessible section for now returns the *directly* accessible section, that is the part of the section that can be reached without having to navigate around or behind obstacles.

4.2. The Flower-Box Dataset

In order to test the ground detection system, a video dataset was created. This has been necessary, since other datasets concerned with ground detection usually focus on road scenes or people detection inside a pedestrian area. The need to create a new dataset originates from the earlier made observation that existing systems do not handle intense

camera movement well. Thus, a dataset has been created where an intense ego-motion can be observed.

The dataset consists of 20 videos of varying length. These videos cover an array of different scenes and objects, such as sidewalks or walkways with parked cars or bicycles, a few pedestrians or cyclists and street poles as well as a few edge cases such as a ladder like sculpture and a narrow ridge. A complete overview of the dataset is given in table 4.4. Some key frames of each video can be seen in figures 4.5 and 4.6.

4.2.1. Acquisition

The dataset has been recorded using a hand-held stereo camera rig and a laptop. This camera system is comprised of two *Point Grey Grasshopper 2* cameras mounted onto a metal carrier at a fixed distance. These cameras can record images at up to 2448x2048 with 15 frames per second or smaller resolutions at higher frame rates. Due to technical limitations the dataset could not be recorded using the fully available frame rate and resolution. The used laptop, a *MacBook Pro 7.1*, possesses only a single FireWire port to connect to the cameras. Luckily, these can be daisy-chained, which also brings the added benefit of camera synchronization, especially important in stereo reconstruction scenarios. Due to these hardware limitations, the dataset has been recorded at a resolution of 1028x768 pixels and 15 frames per second in 8-bit grey mode.

The videos were recorded within a duration of two hours on a sunny day inside an urban area. They were not captured in one session and the listed videos of table 4.4 are not sorted by time, but by content. Altogether, the videos add up to almost nine minutes of video data. Also, much care has been taken to record the videos under realistic settings. Direct sunlight, weak lens flares and strong shadows can be seen in the videos. Furthermore there is much camera movement on all axes to simulate a mobile platform carried by a pedestrian.

A calibration has been obtained before and after the recording of the dataset to mitigate the possibility of an unintended change in the relative camera position due to outside forces. No modification was detected.

In order to improve the stereo reconstruction, several precautionary measures have been taken. The cameras' optical axes are aligned to be parallel. Both cameras record in a synchronized manner through the use of a built-in synchronization mechanism over the FireWire bus. Also, the cameras gains and exposures are synced to create stereo images with similar illumination and contrast.

4.2.2. Labeling Process

Together, all videos contain 7789 frames. Every fifth frame has been labeled, that amounts to three labeled frames per second. Furthermore, the first 30 frames (or two seconds) of each video were not labeled to account for camera initialization of gain and exposure, which adds up to a reduction of 100 labeled frames.

To simplify the labeling process, the videos were first converted into their individual frames. After excluding undesired frames, the remaining were then labeled by hand using a specifically created labeling tool using the matrix laboratory (MATLAB) environment. This

Name	Description	#Labels	#Frames
Alley	Pedestrian walkway with flower-boxes on both sides as well as some parked bicycles.	61	333
Alley Leveled	Like alley, but the camera is approximately leveled to the horizon.	85	452
Bicycle	Bicycle driving up front for a few seconds, a small box obstacle as well as a lamp post and a slope in the end.	58	315
Car	Navigation around a tree, afterwards along a narrow path between a low hedge and a car.	33	194
Corridor	An indoor scene of a long corridor with doors to adjacent offices.	98	516
Fence	A sidewalk with a low fencing on the left side and parked cars to the right.	46	257
Flower-box	Navigation between low flower-boxes with tall coppice and some parked cars.	78	417
Hedge	A sidewalk with a tall hedge along one side, some poles at start and end.	93	490
Ladder	A ladder like sculpture with wide horizontal beams, used as an edge case.	32	186
Narrow	A narrow sidewalk between parked cars and bicycles as well as lamp posts.	91	482
Pan	Horizontal pan of a parking area with flower-boxes and parked car.	44	246
Passage	Passageway containing some flower-boxes, a few parked bicycles and a door.	93	490
Railing	A railing blocking the path and a post along the sidewalk.	48	266
Ramp	A large ramp towards the street with three poles on the side of the road.	41	233
Ridge	A dead end on a narrow walkway between two steep slopes with a wall at the end.	33	193
Sidewalk	A typical sidewalk scene, some parked bicycles and cars, a person walking towards the camera.	171	884
Sidewalk 2	The continuation of the *Sidewalk* video, similar situation, bicycles parked on both sides.	110	578
Sidewalk Leveled	Similar situation as in *Sidewalk*, with the camera leveled at horizon.	121	630
Sign	A tall sign at the side of the pavement, used as an edge case.	22	137
Street	A walk on the street between parked cars and a passing cyclist.	93	490

Figure 4.4.: A list of all Flower-Box dataset videos with a short description of their content. It consists of 20 videos of varying length, totaling 7789 frames of which 1451 have been labeled.

Figure 4.5.: Key frames of (i) Alley, (ii) Alley Leveled, (iii) Bicycle, (iv) Car, (v) Corridor, (vi) Fence, (vii) Flower-box, (viii) Hedge, (ix) Ladder and (x) Narrow.

Figure 4.6.: Key frames of (i) Pan, (ii) Passage, (iii) Railing, (iv) Ramp, (v) Ridge, (vi) Sidewalk, (vii) Sidewalk 2, (viii) Sidewalk Leveled, (ix) Sign and (x) Street.

		Ground Truth	
		pos.	neg.
Prediction	pos.	TP	FP
	neg.	FN	TN

Figure 4.7.: The confusion matrix relating prediction and ground truth. The used abbreviations are: true positive (TP), false positive (FP), false negative (FN) and true negative (TN).

tool marks the desired region of interest with an user specified polygon. The polygon is then in return used to create a binary mask of the labeled frame.

The label is used to mark the accessible ground section, i.e., the image region that can be walked upon by a pedestrian. Further constraints have to be added to guarantee that the labeled section can actually be reached. A valid accessible section has to be connected to the bottom frame boundary. This ensures that no obstacle can separate the ground section from the camera operator and the labeled section is accessible from its current position. Moreover, only the reachable parts of the ground section are labeled, regions that are disconnected by obstacles such as trees or lamp posts are not directly accessible from the current view point and should thus not be labeled. A result of the labeling process can be seen in the evaluation section in figure 4.8(i).

4.3. Evaluation

The performance of the created ground detection system is evaluated by calculating the receiver operating characteristic (ROC) curve and the precision-recall (PR) curve as well as aggregated measures in form of the area under the curve (AUC) and F_β scores. In order to compute these, the predictions are separated into 4 classes, shown in the confusion matrix and an example image in figures 4.7 and 4.8.

The ROC measurement illustrates the performance of a binary classifier relative to a system parameter. To plot the ROC curve, the True Positive Rate (TPR)

$$\mathrm{TPR} = \frac{\#\text{true positives}}{\#\text{positives}} = \frac{\#\text{true positives}}{\#\text{true positives} + \#\text{false positives}}, \tag{4.5}$$

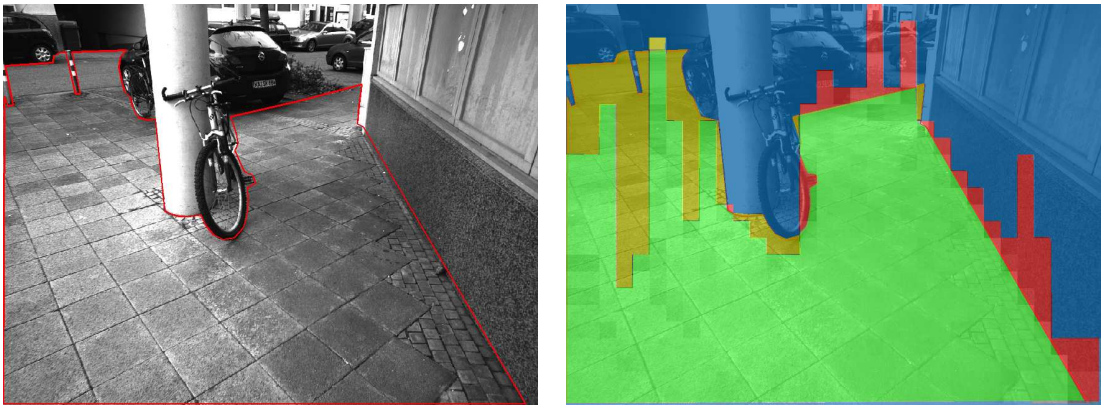
as well as the False Positive Rate (FPR)

$$\mathrm{FPR} = \frac{\#\text{false positives}}{\#\text{negatives}} = \frac{\#\text{false positives}}{\#\text{false positives} + \#\text{true negatives}}, \tag{4.6}$$

are measured for representative thresholds. The threshold for the accessible section detection system will be the deviation of a surface normal and its corresponding calculated angle that is pointed straight upwards.

The recall is the proportion of the predicted positive values which are actual positives. Here, these are all the image pixels classified by the system as accessible section that are contained inside the labeled ground truth regions. Thus the recall can be calculated as

$$\mathrm{recall} = \frac{\#\text{true positives}}{\#\text{true positives} + \#\text{false negatives}}. \tag{4.7}$$



(i) (ii)

Figure 4.8.: An example of the four prediction classes. The original image (i) with the accessible section outlined in red. The bottom image (ii) shows the prediction with the individual classes which are the correctly identified section (true positive) in *green*, the missed parts (false negatives) in *yellow*, false identified parts (false positives) in *red* and the remainder of the scene (true negative) in *blue*.

The precision is the proportion of correct matches relative to the false and correct matches,

$$\text{precision} = \frac{\#\text{true positives}}{\#\text{true positives} + \#\text{false positives}}. \tag{4.8}$$

The F_β score combines precision and recall into a single value, their individual weights determined by β,

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}. \tag{4.9}$$

When evenly weighted, the F-score becomes the balanced F-measure, also known as the F_1 score,

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{4.10}$$

Additionally, in the following $F_{0.5}$ will be used, as it places a higher importance on precision than recall. A high precision seems more relevant than a high recall in this application, since it correlates with a high percentage of correct results with few false positives. This is especially important when dealing with a system that directly affects a human being, as it seems preferable to detect all obstacles rather than all of the accessible section.

Finally, the overall accuracy for each video is calculated as

$$\text{accuracy} = \frac{\#\text{true positives} + \#\text{true negatives}}{\#\text{true positives} + \#\text{true negatives} + \#\text{false positives} + \#\text{false negatives}}. \tag{4.11}$$

4.4. Results

The created accessible section detection achieves varying results depending on circumstances like the amount of observable ground plane or its texture. Three good classification results can be seen in figure 4.9, one indoor and two outdoor scenes. These three settings constitute common scenarios for a navigational support system for visually impaired persons. The outdoor scenes contain a sidewalk, once narrowed by parked cars and bicycles and once mainly free of obstacles. In these and similar scenarios, where there is a great variability in the texture of the observed scene, the classification achieves good results. This is due to the fact that the stereo reconstruction algorithm provides a stable and gap-less disparity map which in turn leads to a good classification.

In figure 4.10 three bad classification results can be seen. The system fails these situations and only a small percentage of the accessible ground section is classified. Either the stereo reconstruction cannot deliver sufficient information or the classifier cannot deal with the created noise in the disparity map. Figure 4.11 shows such a case that is part of the *Pan* video. Large holes and artefacts as well as strong noise in the disparity map cause the classification to fail on many image parts. Originally, 36 videos were recorded for the data set, but almost half had to be discarded due to a poor stereo reconstruction quality. To further improve the classification method and discern between a failure on the segmentation part, additional measures have to be taken. This could for example include the development of a disparity map check of some sort.

Figures 4.12 and 4.13 show the per video averaged ROC and PR curves. Especially for the *Alley*, *Alley Leveled*, *Narrow*, *Sidewalk*, *Sidewalk 2*, *Sidewalk Leveled* and *Street* videos a very high discrimination ratio is achieved, almost always above 90%. These videos have a fairly large stretch of accessible section with obstacles located on both sides in common as well as only very few obstructions separating the ground section. Good results are also achieved on the *Car*, *Corridor*, *Fence*, *Hedge*, *Ladder*, *Passage* and *Sign* videos. Here, a variable amount of directly accessible ground section is available and it is often separated into several parts. Classification rates are still above 80% and support the initial assumption that the gradient classifier could deliver good results even under difficult circumstances. The *Bicycle*, *Flower-box*, *Pan*, *Railing* and *Ramp* videos show the algorithm's limits. Large parts of the ground section cannot be recognized. The worst detection accuracy is observed in the *Ridge* video. Much noise is created in the disparity map by the combination of grass partly covered with snow, which has very low texture information. Furthermore, this video has by far the smallest amount of visible ground section which causes the algorithm to detect more false positives in relation to other videos.

An overview of the achieved classification rates for all videos is shown in figure 4.14 as well as their standard deviation 4.15. The area under the curve (AUC) for ROC and PR are given as well as various F-scores. $F_{0.5}$ and F_1 scores show that the algorithm has a high reliability. Also, the small difference between the mean of the maximum and the fixed threshold F-scores and accuracies show that the selection of a general threshold is possible.

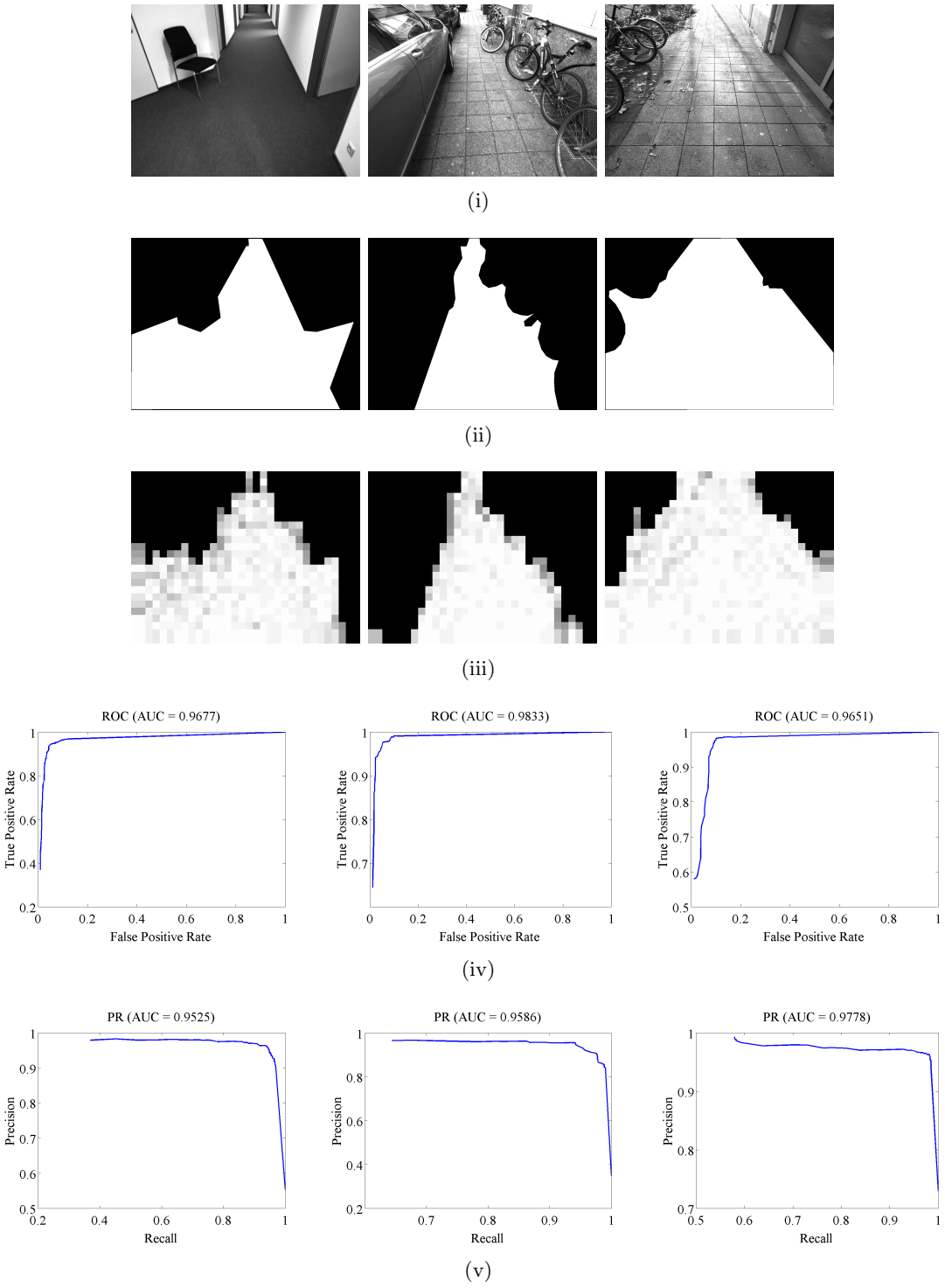


Figure 4.9.: Three good recognition examples. They are taken from the *Corridor*, *Sidewalk* and *Narrow* videos (from left to right) and show: (i) original frame, (ii) generated ground truth mask, (iii) accessible ground plane prediction, (iv) ROC curve and (v) PR curve.

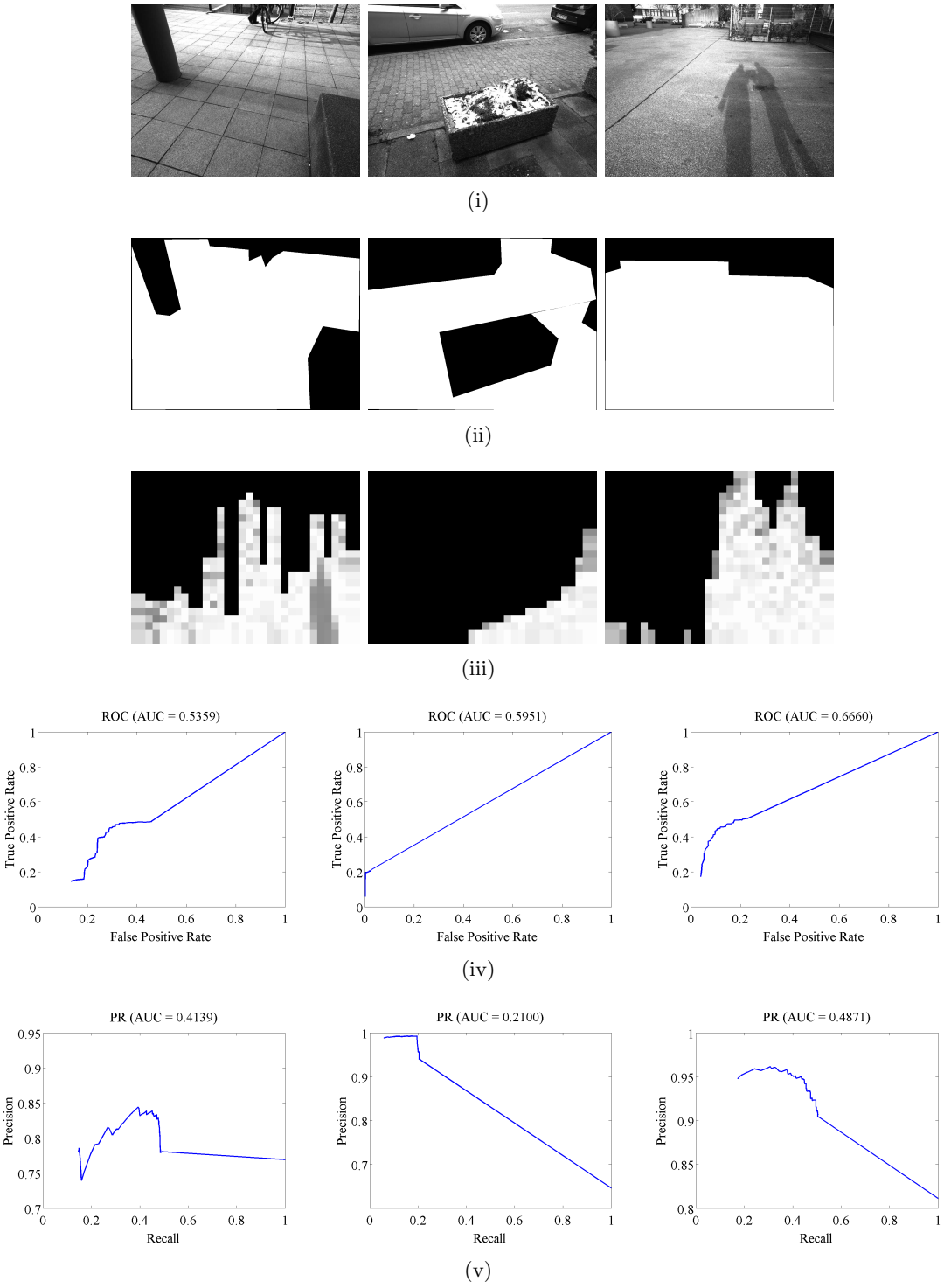
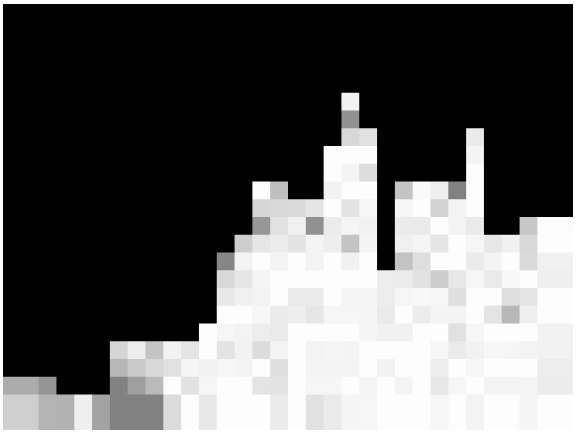


Figure 4.10.: Three bad recognition examples. They are taken from the *Bicycle*, *Flowerbox* and *Pan* videos (from left to right) and show: (i) original frame, (ii) generated ground truth mask, (iii) accessible ground plane prediction, (iv) ROC curve and (v) PR curve.

(i)

(ii)



(iii)

Figure 4.11.: Here a low recognition score is caused by a failing of the stereo reconstruction algorithm. The images show the open area of a parking space (i), the problems of the disparity map with large holes and artifacts (ii) and the failing accessible section detection (iii).

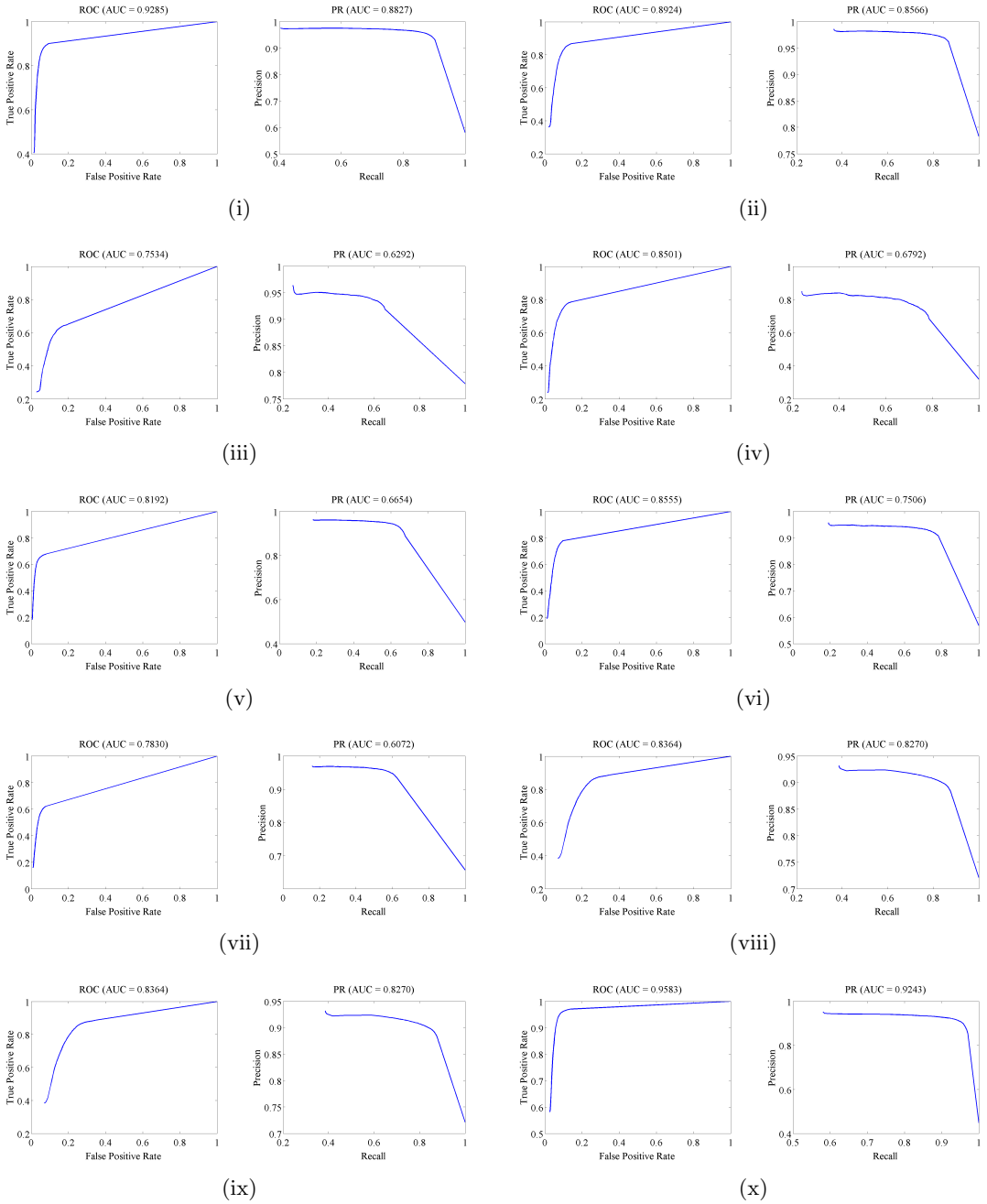


Figure 4.12.: ROC and PR curves of (i) Alley, (ii) Alley Leveled, (iii) Bicycle, (iv) Car, (v) Corridor, (vi) Fence, (vii) Flower-box, (viii) Hedge, (ix) Ladder and (x) Narrow.

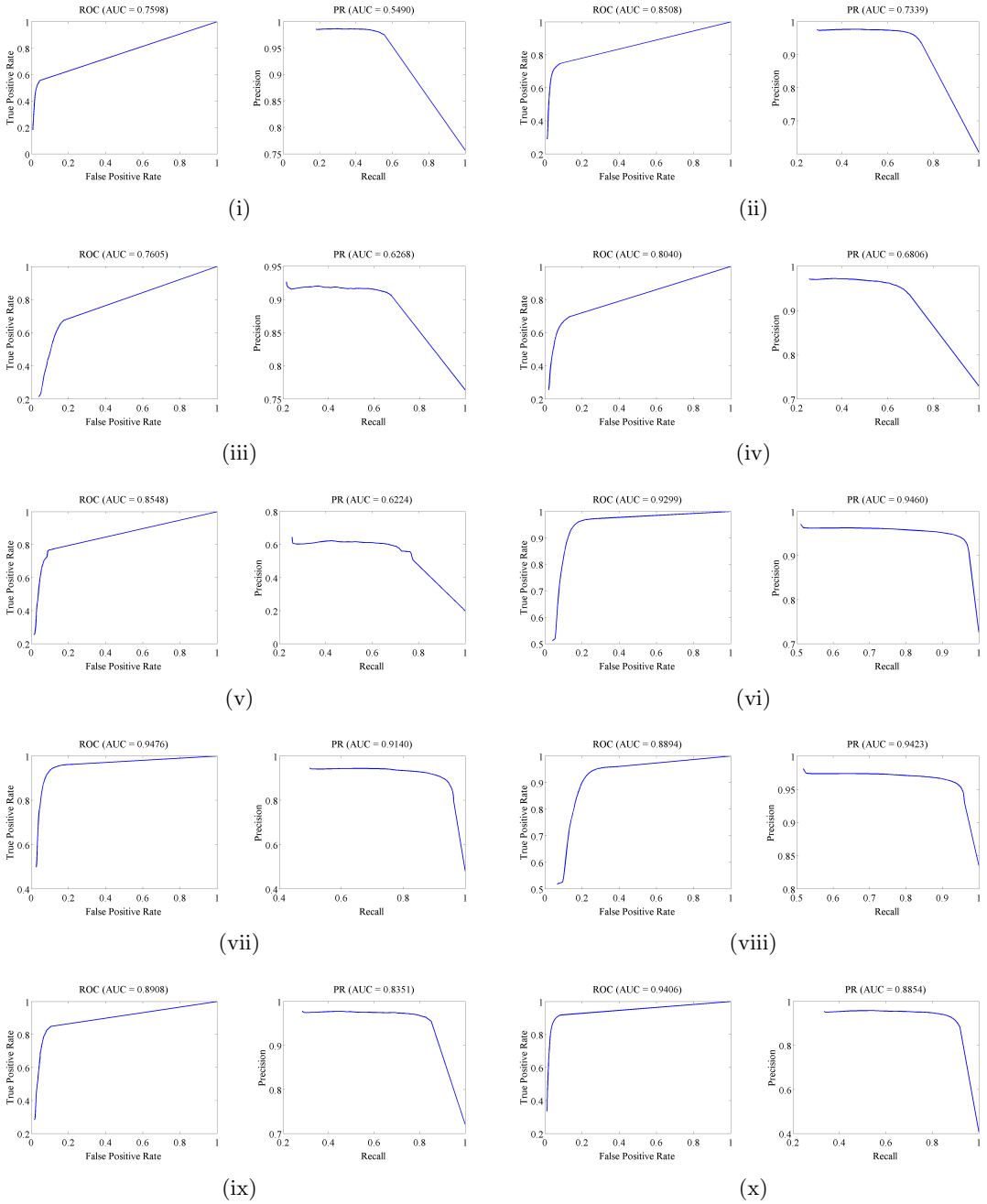


Figure 4.13.: ROC and PR curves of (i) Pan, (ii) Passage, (iii) Railing, (iv) Ramp, (v) Ridge, (vi) Sidewalk, (vii) Sidewalk 2, (viii) Sidewalk Leveled, (ix) Sign and (x) Street.

Name	$\int$ROC	$\int$PR	$\hat{F}_{0.5}$	$F_{0.5}$	$\hat{F}_1$	F_1	$\widehat{Accuracy}$	Accuracy
Alley	0.928	0.882	0.940	0.937	0.919	0.916	0.904	0.901
Alley Leveled	0.892	0.856	0.943	0.941	0.919	0.911	0.867	0.862
Bicycle	0.753	0.629	0.871	0.843	0.882	0.869	0.786	0.676
Car	0.850	0.679	0.774	0.763	0.749	0.739	0.858	0.851
Corridor	0.819	0.665	0.851	0.816	0.799	0.750	0.805	0.796
Fence	0.855	0.750	0.885	0.878	0.843	0.834	0.823	0.815
Flower-box	0.783	0.607	0.848	0.838	0.815	0.789	0.750	0.724
Hedge	0.836	0.827	0.898	0.882	0.901	0.872	0.855	0.814
Ladder	0.836	0.629	0.774	0.757	0.744	0.736	0.878	0.868
Narrow	0.958	0.924	0.927	0.922	0.932	0.928	0.932	0.929
Pan	0.759	0.548	0.848	0.843	0.862	0.861	0.690	0.650
Passage	0.850	0.733	0.898	0.889	0.857	0.821	0.819	0.805
Railing	0.760	0.626	0.868	0.842	0.876	0.852	0.790	0.696
Ramp	0.803	0.680	0.880	0.870	0.879	0.839	0.773	0.731
Ridge	0.854	0.622	0.624	0.230	0.648	0.304	0.750	0.199
Sidewalk	0.929	0.945	0.947	0.943	0.950	0.947	0.919	0.913
Sidewalk 2	0.947	0.914	0.920	0.913	0.920	0.912	0.915	0.904
Sidewalk Leveled	0.889	0.942	0.958	0.954	0.959	0.950	0.923	0.912
Sign	0.890	0.835	0.935	0.933	0.901	0.899	0.859	0.854
Street	0.940	0.885	0.923	0.919	0.910	0.904	0.922	0.917
$\bar{x}$	0.852	0.753	0.873	0.861	0.842	0.828	0.837	0.784

Figure 4.14.: Per dataset video results: the area under the curve (AUC) for the ROC and PR curves, F_β for $\beta = 0.5$ and $\beta = 1$ as well as per video accuracy. The $\hat{}$ operator denotes the average of the maximum $F_{0.5}$ score threshold over each frame. $\bar{x}$ shows the average score of the entire dataset.

Name	$\int$ROC	$\int$PR	$\hat{F}_{0.5}$	$F_{0.5}$	$\hat{F}_1$	F_1	$\widehat{Accuracy}$	Accuracy
Alley	0.0252	0.0518	0.01266	0.0129	0.0271	0.0276	0.0265	0.0281
Alley Leveled	0.0378	0.0596	0.01716	0.0168	0.0300	0.0336	0.0440	0.0447
Bicycle	0.0659	0.0989	0.05984	0.0597	0.0667	0.0906	0.0903	0.0897
Car	0.0525	0.1132	0.09208	0.0932	0.0767	0.0756	0.0523	0.0579
Corridor	0.1013	0.2011	0.09240	0.1176	0.0910	0.1601	0.1018	0.1016
Fence	0.0471	0.0861	0.02208	0.0228	0.0393	0.0467	0.0608	0.0661
Flower-box	0.0686	0.1437	0.06126	0.0697	0.0662	0.0680	0.0752	0.0949
Hedge	0.0829	0.0875	0.05748	0.0521	0.0556	0.0485	0.0665	0.0700
Ladder	0.0503	0.1464	0.13493	0.1377	0.1165	0.1207	0.0408	0.0417
Narrow	0.0168	0.0278	0.01600	0.0168	0.0147	0.0157	0.0190	0.0238
Pan	0.0422	0.0814	0.03054	0.0343	0.0203	0.0201	0.0753	0.0687
Passage	0.0707	0.1405	0.04571	0.0570	0.0523	0.0989	0.0827	0.1024
Railing	0.0621	0.0745	0.09489	0.0753	0.0834	0.1338	0.1143	0.0717
Ramp	0.0705	0.1212	0.04445	0.0421	0.0450	0.0674	0.1012	0.1018
Ridge	0.1766	0.3410	0.33426	0.1767	0.3405	0.2142	0.3590	0.1609
Sidewalk	0.0287	0.0299	0.01605	0.0161	0.0149	0.0175	0.0285	0.0301
Sidewalk 2	0.0278	0.0432	0.02825	0.0293	0.0284	0.0310	0.0298	0.0343
Sidewalk Leveled	0.0602	0.0361	0.01785	0.0190	0.0205	0.0214	0.0349	0.0310
Sign	0.0182	0.0316	0.00861	0.0086	0.0152	0.0175	0.0258	0.0257
Street	0.0283	0.0554	0.02149	0.0219	0.0302	0.0309	0.0257	0.0276
$\bar{x}$	0.0567	0.0985	0.0604	0.0540	0.0617	0.0670	0.0727	0.0636

Figure 4.15.: Per dataset video standard deviation: the area under the curve (AUC) for the ROC and PR curves, F_β for $\beta = 0.5$ and $\beta = 1$ as well as per video accuracy. The $\hat{}$ operator denotes the standard deviation of the maximum $F_{0.5}$ score threshold over each frame. $\bar{x}$ shows the standard deviation of the entire dataset. See also figure 4.14.

5. Conclusion

The existence of numerous frameworks for various research purposes indicates the need of a common base to foster collaboration among researchers. However, due to their development towards a particular purpose, they often contain many parts that are not needed in other research areas and may become hard to use in a different scenario. Thus, in this thesis, a framework aimed towards visually impaired persons has been presented with a focus on simplicity and ease of use by other research students. The frameworks main design goals: modularity, versatility, functionality, encapsulation and provision of core components have greatly helped in the development process of a walkable area detection system for visually impaired persons. By reusing already existing framework components, much development time was saved. Furthermore, the framework could be tested with regard to its ease of use and flexibility. It can be concluded that the framework justifies its invested work. Already, a few useful modules have been developed by other student researchers from the same laboratory.

While there already exist many ground plane detection systems, such systems were usually developed taking into account several artificially enforced constraints. Conditions such as knowledge about the exact camera position were used as well as the existence of specific markers, amongst others. This dependency upon external constraints renders them unfeasible for visually impaired persons, as these cannot be guaranteed. Thus, a simple and effective feature has been introduced, the gradient detection inside a disparity map produced by a stereo reconstruction algorithm. The gradient uses the correlation between the orientation of a surface in the real world and its representation in the acquired disparity map. Its calculation is neither complex nor computationally expensive and can thus be done fast and efficiently. In addition, a simple detector to retrieve the accessible section was created. This detector relies on a few simple geometric constraints to process the detected ground region. The simple approach of the ground detection system yields surprisingly effective classifications, but can only be seen as a small step towards a reliable navigation system for visually impaired persons.

A real world dataset was created to evaluate the developed ground detection system. It was collected inside an urban area using a hand held stereo camera rig and a laptop.

The dataset consists of 20 videos with almost 8000 frames in total. The videos cover a large variety of scenes ranging from simple walkways, sidewalks including parked cars or bicycles, a few pedestrians, cyclists, street poles and edge cases such as a narrow ridge. Of these, in 1451 frames the accessible section has been labeled by hand using a specifically created labeling tool.

The proposed system achieved an overall correct classification rate of 0.784 averaged over the entire dataset. The accuracy for a fixed threshold varies from 0.199 for an edge case up to 0.929 for each video.

5.1. Future Work

The existing framework proves to be useful in the development of applications and prototypes. Through combination of existing modules and the creation of missing pieces existing applications can easily be extended. Nonetheless, there are features that are still missing.

Probably the most desired feature during the ground detection development using the provided framework has been the need for an unified graphical user interface (GUI). A GUI can greatly help with control of the framework, as it would provide an even easier access to its functionality. An example of this would be the creation of an application by simple drag and drop operations using a visual representation of the existing modules and their connections. Such a visualization of the module graph could greatly help in understanding the interaction and cooperation between different used modules. By displaying additional information, such as latency and runtime information, bottlenecks become easily visible. Furthermore, such a visualization could immensely help with the debugging process. Through provision of general and abstract view ports of the data inside the framework connections, problems with features or algorithms become easily accessible.

Since it is not always possible, or simply impractical, to get input from a live system, such as a camera rig, a data recorder and replay functionality is desirable for the development and testing phase, as well as for evaluation. By moving such functionality into the framework, they become abstract facilities that do not depend on specific data types. Thus, their reuse is encouraged for varying applications that needed an array of different data types. Furthermore, by providing a deep integration into the frameworks internals, advanced functionality such as data skip and rewind can be implemented. Such capabilities can, like the GUI, greatly help in the development process.

In the research community an important tool next to *C++* is the matrix laboratory (MATLAB) computing environment. Much research is done inside MATLAB due to its great integration of mathematical concepts as well as functionality to plot data and integration of commonly required algorithms. It has already been used in this thesis to create the labeling tool. But MATLAB can also interface with other languages, such as *C++*. Thus it would be interesting to connect the created framework with MATLAB by creating bindings for it. This step would allow the early prototyping of modules inside MATLAB and simplify later integration into the framework.

In order to create a wearable system aimed at navigational support for visually impaired people, a port to a mobile platform is necessary. Modern smartphones could provide an ideal target for this. They have greatly gained in processing power as well as available

memory over the years and also contain high resolution cameras. The Android platform seems especially suited for this purpose, since it is open source and widely used.

Ways of providing feedback to the user must also be explored. Through haptic interfaces, such as the Vibrotac [SEWP10] device, guidance information could be passed along. Sonification could also provide such a feedback system, but the feasibility of any approach must be examined first. Integration of such systems is an important step towards the creation of usable navigation support systems.

The ground detection system can be further improved by evolving the gradient and the detection of the accessible system using quadtree like structures and varying gradient kernel sizes. By processing the disparity map with such improvements, the focus could be concentrated on regions of interest and thus a finer grained representation of the ground plane could be acquired. Furthermore, using a three dimensional representation could help in specifically difficult situations. Various approaches, such as simultaneous localization and mapping (SLAM) [Dav03], dense tracking and mapping (DTAM) [NLD11] and parallel tracking and mapping (PTAM) [KM09] already exist, although their feasibility on mobile platforms remain to be tested due to high complexity and processing requirements.

Bibliography

[BBK+01] M. Bauer, B. Bruegge, G. Klinker, A. MacWilliams, T. Reicher, S. Riss, C. Sandor, and M. Wagner, "Design of a component-based augmented reality framework," in *Augmented Reality, 2001. Proceedings. IEEE and ACM International Symposium on*, 2001, pp. 45–54.

[BPCL06] C. Braillon, C. Pradalier, J. Crowley, and C. Laugier, "Real-time moving obstacle detection using optical flow models," in *Intelligent Vehicles Symposium, 2006 IEEE*, 0-0 2006, pp. 466–471.

[BWC01] D. Beazley, B. Ward, and I. Cooke, "The inside story on shared libraries and dynamic loading," *Computing in Science Engineering*, vol. 3, no. 5, pp. 90–97, sep/oct 2001.

[CFZ+12] D. Chen, W. Feng, Q. Zhao, M. Hu, and T. Wang, "An infrastructure-free indoor navigation system for blind people," *Intelligent Robotics and Applications*, pp. 552–561, 2012.

[CM09] J. Coughlan and R. Manduchi, "A mobile phone wayfinding system for visually impaired users," *Assistive technology research series*, vol. 25, no. 2009, p. 849, 2009.

[CMG05] T. H. Collett, B. A. MacDonald, and B. P. Gerkey, "Player 2.0: Toward a practical robot programming framework," in *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005)*, 2005.

[CTV05] S. Cardin, D. Thalmann, and F. Vexo, "Wearable obstacle detection system for visually impaired people," in *VR workshop on haptic and tactile perception of deformable objects*, 2005, pp. 50–55.

[CVH08] N. Chumerin and M. M. Van Hulle, "Ground plane estimation based on dense stereo disparity," in *The Fifth International Conference on Neural Networks and artificial intelligence, Minsk, Belarus*, 2008, pp. 209–213.

[Dav03] A. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, oct. 2003, pp. 1403 –1410 vol.2.

[Del34] B. Delaunay, "Sur la sphere vide," *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, vol. 7, no. 793-800, pp. 1–2, 1934.

[DH72] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[DK08] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, 2008.

[ELSVG09] A. Ess, B. Leibe, K. Schindler, and L. Van Gool, "Moving obstacle detection in highly dynamic scenes," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, may 2009, pp. 56–63.

[ESLVG10] A. Ess, K. Schindler, B. Leibe, and L. Van Gool, "Object detection and tracking for autonomous navigation in dynamic environments," *The International Journal of Robotics Research*, vol. 29, no. 14, pp. 1707–1725, 2010.

[FB81] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[GRU11] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," *Computer Vision–ACCV 2010*, pp. 25–38, 2011.

[GVH03] B. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proceedings of the 11th international conference on advanced robotics*, vol. 1. Portugal, 2003, pp. 317–323.

[GVS+01] B. Gerkey, R. Vaughan, K. Stoy, A. Howard, G. Sukhatme, and M. Mataric, "Most valuable player: a robot device server for distributed control," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 3, 2001, pp. 1226–1231.

[HLM12] Y. Hoon, L. T.-s. Leung, and G. Medioni, "Real-time staircase detection from a wearable stereo system," *ICPR - International Conference on Pattern Recognition*, pp. 6–9, 2012.

[Hra08] S. Hrabar, "3d path planning and stereo-based obstacle avoidance for rotorcraft uavs," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, sept. 2008, pp. 807–814.

[HS81] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1, pp. 185–203, 1981.

[HW77] P. W. Holland and R. E. Welsch, "Robust regression using iteratively reweighted least-squares," *Communications in Statistics-Theory and Methods*, vol. 6, no. 9, pp. 813–827, 1977.

[Kat02] H. Kato, "Artoolkit: Library for vision-based augmented reality," *IEICE, PRMU*, pp. 79–86, 2002.

[KM09] G. Klein and D. Murray, "Parallel tracking and mapping on a camera phone," in *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, oct. 2009, pp. 83 –86.

[LAT02] R. Labayrade, D. Aubert, and J.-P. Tarel, "Real time obstacle detection in stereovision on non flat road geometry through "v-disparity" representation," in *Intelligent Vehicle Symposium, 2002. IEEE*, vol. 2, june 2002, pp. 646–651.

[LM11] Y. H. Lee and G. Medioni, "Rgb-d camera based navigation for the visually impaired." Robotic Science and Systems, 2011.

[LPB12] M. C. Le, S. L. Phung, and A. Bouzerdoum, "Pedestrian lane detection for the visually impaired," in *Digital Image Computing Techniques and Applications (DICTA), 2012 International Conference on*, dec. 2012, pp. 1–6.

[LSC12] C.-H. Lee, Y.-C. Su, and L.-G. Chen, "An intelligent depth-based obstacle detection system for visually-impaired aid applications," in *Image Analysis for Multimedia Interactive Services (WIAMIS), 2012 13th International Workshop on*, may 2012, pp. 1–4.

[LZM05] P. Lombardi, M. Zanin, and S. Messelodi, "Unified stereovision for ground, road, and obstacle detection," in *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, june 2005, pp. 783–788.

[MH08] K. Martin and B. Hoffman, *Mastering CMake 4th Edition.* Kitware, Inc., 2008.

[ML12] D. Mitzel and B. Leibe, "Close-Range Human Detection for Head-Mounted Cameras," *The British Machine Vision Association and Society for Pattern Recognition*, pp. 1–11, 2012.

[MR+08] J. M. S. Martinez, F. E. Ruiz *et al.*, "Stereo-based aerial obstacle detection for the visually impaired," in *Workshop on Computer Vision Applications for the Visually Impaired*, 2008.

[MRT03] M. Montemerlo, N. Roy, and S. Thrun, "Perspectives on standardization in mobile robot programming: the carnegie mellon navigation (carmen) toolkit," in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 3, oct. 2003, pp. 2436–2441.

[NLD11] R. Newcombe, S. Lovegrove, and A. Davison, "Dtam: Dense tracking and mapping in real-time," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, nov. 2011, pp. 2320 –2327.

[OR12] C. Olson and A. Robinson, "Camera-aided human navigation: Advances and challenges," in *Emerging Signal Processing Applications (ESPA), 2012 IEEE International Conference on*, jan. 2012, pp. 75–78.

[QGC+09] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009.

[SB02] S. Se and M. Brady, "Ground plane estimation, error analysis and applications," *Robotics and Autonomous Systems*, vol. 39, no. 2, pp. 59–71, 2002.

[SEWP10] S. Schaetzle, T. Ende, T. Wuesthoff, and C. Preusche, "Vibrotac: An ergonomic and versatile usable vibrotactile feedback device," in *RO-MAN, 2010 IEEE*, sept. 2010, pp. 670 –675.

[SGCH01] K. J. Sullivan, W. G. Griswold, Y. Cai, and B. Hallen, "The structure and value of modularity in software design," *ACM SIGSOFT Software Engineering Notes*, vol. 26, no. 5, p. 99, Sep. 2001.

[Ste11] M. Stevens, "Subaru eyesight coming to australia for liberty and outback in late 2011," 2011. [Online]. Available: http://www.themotorreport.com.au/51481/subaru-eyesight-coming-to-australia-for-liberty-and-outback-in-late-2011

[Str93] B. Stroustrup, "New Casts Revisited," 1993. [Online]. Available: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/1993/N0349a.pdf

[SUB03] S. Shoval, I. Ulrich, and J. Borenstein, "Navbelt and the guide-cane [obstacle-avoidance systems for the blind and visually impaired]," *Robotics Automation Magazine, IEEE*, vol. 10, no. 1, pp. 9–20, mar 2003.

[Sut09] H. Sutter, "The Fast Pimpl Idiom," 2009. [Online]. Available: http://www.gotw.ca/gotw/028.htm

[Sü10] Südostschweiz.ch, "Am Tag des Weissen Stocks testen Blinde die Anhaltedisziplin," 2010. [Online]. Available: http://www.suedostschweiz.ch/vermischtes/am-tag-des-weissen-stocks-testen-blinde-die-anhaltedisziplin

[TH05] L. Torvalds and J. Hamano, "GIT-fast version control system," 2005. [Online]. Available: www.git-scm.com

[vH08] D. van Heesch, "Doxygen: Source code documentation generator tool," 2008. [Online]. Available: www.doxygen.org

[Zel09] A. Zelinsky, "Learning opencv—computer vision with the opencv library (bradski, g.r. et al.; 2008)[on the shelf]," *Robotics Automation Magazine, IEEE*, vol. 16, no. 3, p. 100, september 2009.

Appendix

A. BVS System Overview

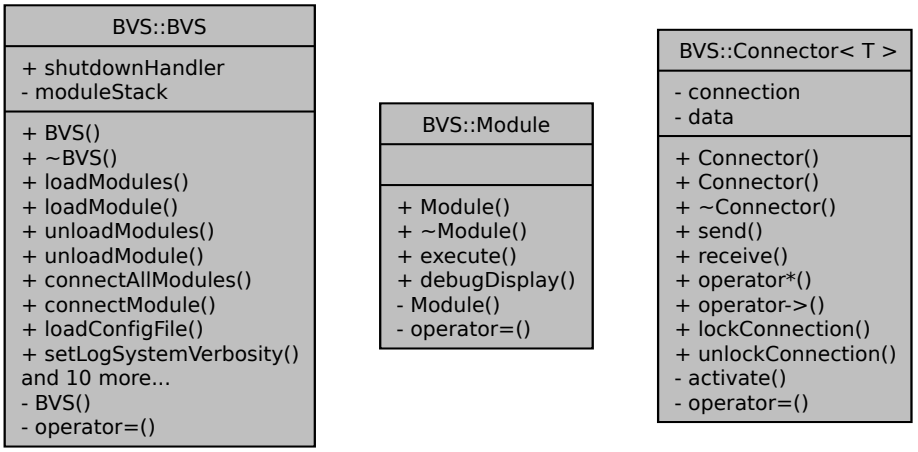


Figure A.1.: These are BVS' main interfaces. On the left, the client interface can be seen. It is used to control the framework and interact with its parts. In the middle, the module interface used by the framework to interact with each module is shown. On the right, the module connector template used by module connections is displayed.

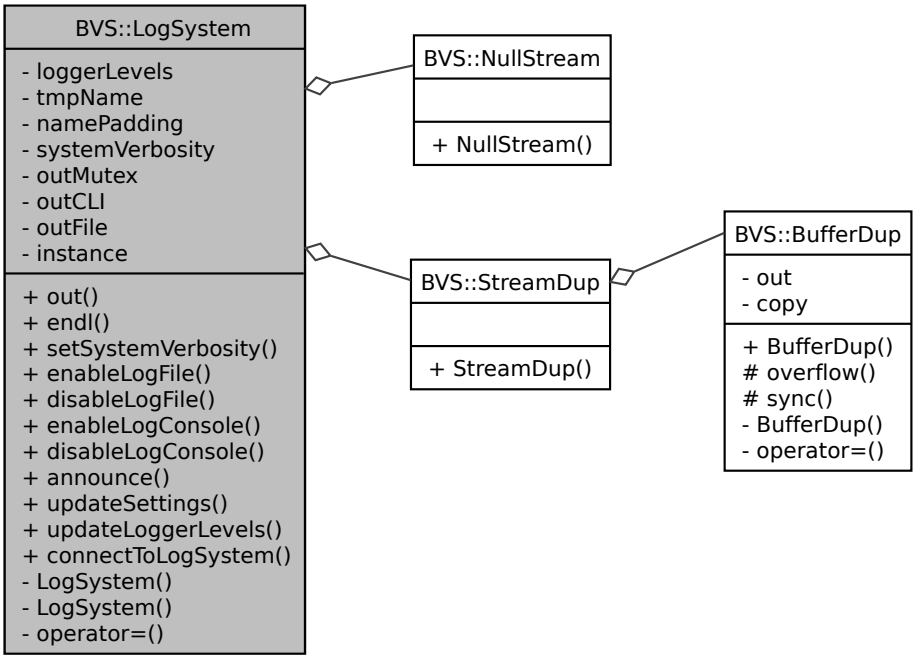


Figure A.2.: This is the logging system backend together with its helper classes.

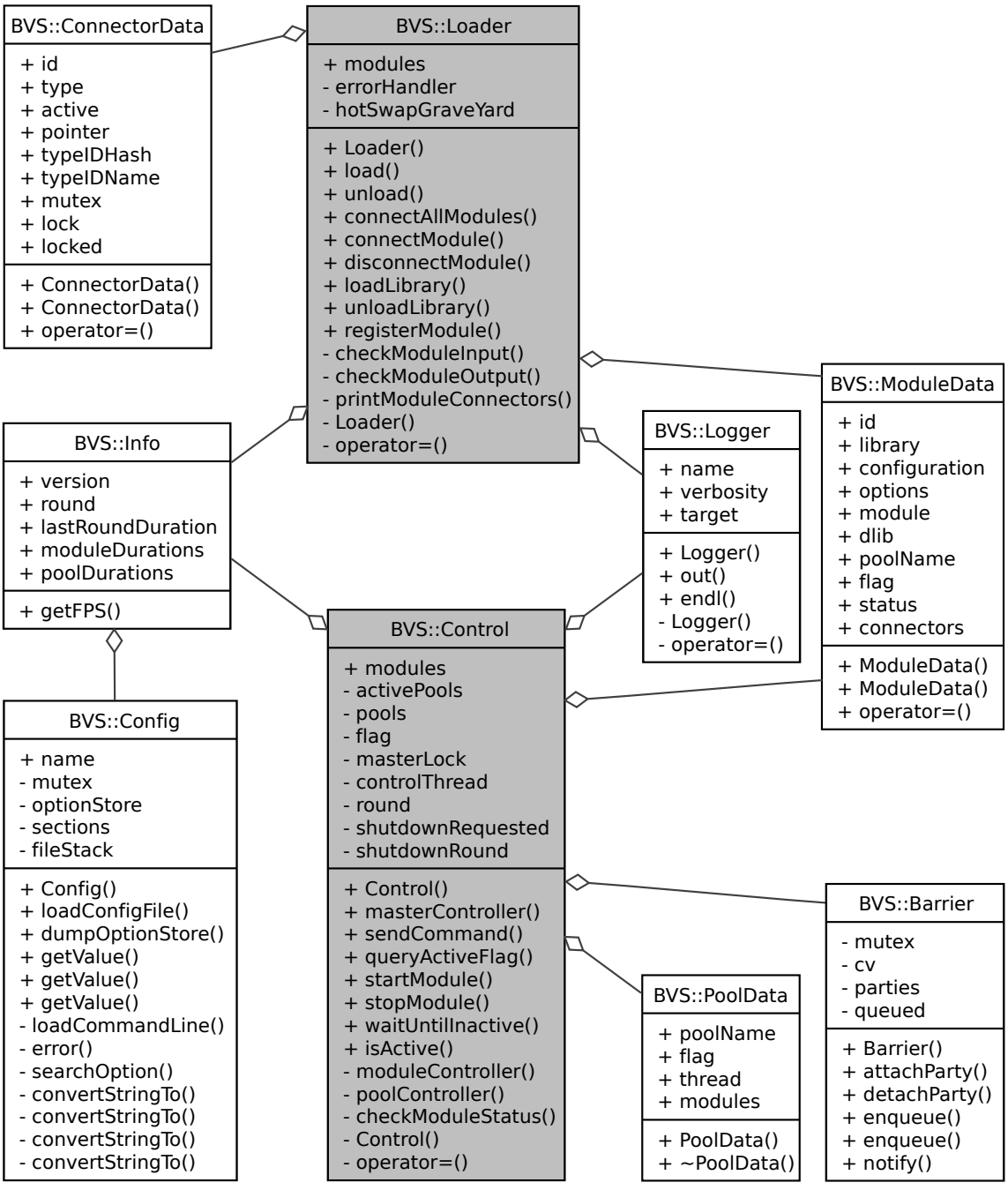


Figure A.3.: Here the relationship between the loading mechanism and the control system is shown. Both share access to commonly used data and control structures as well as internal framework interfaces.