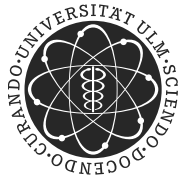


ulm university universität
uulm

Faculty of Engineering and Computer Science
Institute of Neural Information Processing

Biologically Inspired Model for Visually Driven Navigation

Diploma Thesis

submitted by
Nicolai Sebastian Waniek

Reviewers
Prof. Dr. Heiko Neumann, Ulm University
Dr. Florian Raudies, Boston University

19th March 2012

Nicolai Sebastian Waniek
Mat.No. 625861
nicolai.waniek@uni-ulm.de

Version: 19th March 2012
Typesetting: pdfTeX, Version 3.1415926-2.3-1.40.12

LICENSE DETAILS

ACKNOWLEDGMENTS

First and foremost, I want to express my deepest gratitude to my reviewers and advisers Prof. Dr. Heiko Neumann and Dr. Florian Raudies. Their input and guidance throughout the thesis were invaluable. When I started to wander too far into some fields of research, they helped me to stay on track and focused. Yet, they gave me all the freedom I desired. Their encouragement deepened my excitement about natural computation and computational neuroscience.

I would also like to thank all members of Prof. Dr. Neumann's team, especially Stephan, Stefan, Georg and Tobias. They always took the time to listen to me when I got stuck. Their counsel helped me to view problems differently and thus solve them.

My appreciation extends in particular to Maike, Christoph, Ferdinand, Johannes and my family. Without their support, this thesis would not have been possible.

Finally, I thank everyone who had an impact on me and this work.

ABSTRACT

Acquiring precise self-motion estimates from monocular visual input is an import problem in many SLAM (Simultaneous Localization And Mapping) systems. In this thesis, I present a novel biologically inspired model to increase the accuracy of those estimates. The model fuses two different processing paths, each of which yields an individual estimate of the self-motion. One path uses a template model of cortical area MST, the other path the epipolar geometry. Fusion of the two estimates is carried out by a simple head direction cell network. Augmented by a prediction signal, the fused estimate is used as feedback to the individual paths. In simulations, I can show that the feedback is beneficial to the self-motion estimation process. Consequently, the model yields a higher accuracy when compared to results without feedback. The model can serve to inspire real-time capable algorithms in robotics, or to guide research in neuroscience.

CONTENTS

1 INTRODUCTION TO THE BIOLOGICALLY INSPIRED MODEL FOR VISUALLY DRIVEN NAVIGATION

Many autonomous robots or animals use visual information for dead-reckoning. They process visual cues such as certain landmarks to establish intelligence on their movement and spatial location. If no map is available previously, the area needs to be dynamically mapped. For instance, an animal which is forced to increase its habitat due to food shortage needs to learn the new places in which it preys. Otherwise, homing would not be possible. Another example is a service robot that is deployed in many different buildings. It is often too costly or even not possible at all to create maps in advance. As a consequence, both need to solve the problem of simultaneous localization and mapping (SLAM[1]).

Knowledge about the self-motion is required in most technical solutions to SLAM. In fact, a higher precision of this information can help to improve the quality of the map. Biological systems are able to accurately integrate angular and linear motion to form a "cognitive map" of the environment [MBJ+06]. The question arises if biological mechanisms can be used to improve the self-motion estimate that is required in technical solutions to SLAM.

SPATIAL REPRESENTATION IN THE RODENT BRAIN

There is a variety of cells in the rodent brain[2] that code for pose, spatial location and self-motion. They are believed to help the animal navigate and integrate different parts of its movement. For instance, place cells or grid cells have a direct relation to the environment in which a rat moves [MKM08]. Imagine a rat moving through an area while a recording of the neuronal activity in the rat's hippocampus is conducted. The results show that one specific place cell discharges maximally when the rat is positioned in one certain area, which is approximately circular to some radius. As soon as the rat leaves this area, the

1 Many authors refer to the term as Self-Localization and Mapping. In most cases, the two terms can be used interchangeably.

2 Recent research focuses on the rodent brain. However, some of the described neurons have already been found in the primate brain. See, for example, [Rol99].

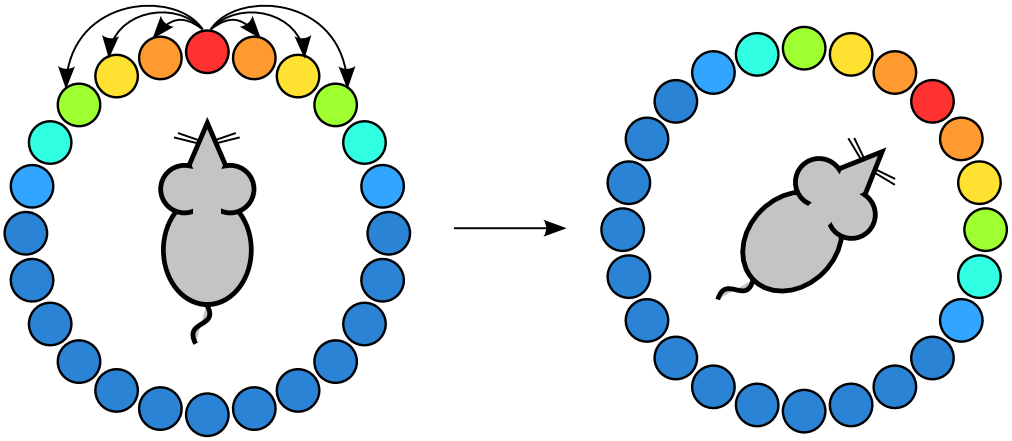


Figure 1.1: Head Direction Network in the rodent brain. All cells are connected with nearby neighbors. For visibility reasons, few connections of only one cell are shown. Cells with warmer colors are more active than cells with colder colors. Head direction (HD) cells have a preferential tuning towards a certain direction. Thus, they will only be active when the rat's pose describes a certain orientation. As soon as the rat turns in another direction, the HD cell with the corresponding tuning discharges maximally. The Figure contains only a simple version of a head direction network, they are more complex in nature. (cf. [MBJ+06])

neural activity of this place cell drops while the activity of another place cell increases. The area in which a place cell discharges is called place field. Place fields of different place cells overlap, and exist in differing sizes. In contrast to place cells, grid cells have multiple fields in which they are active. The grid fields form a repeating hexagonal pattern, which is the name origin for these cells. Beside those two examples, there are many more cells coding for different portions of a pose and spatial representation.

Head direction (HD) cells act like a compass. They yield an information about the rat's heading direction, but they do not depend on magnetic information. HD cells are primarily fed by allothetic[3] visual input, but receive activity from idiothetic motion cues as well [JS98]. As a consequence of the input, the state of the HD cells (or HD network) changes. The HD network thus is an integrator of the rotational information of self-motion. Only those cells are active whose preferential tuning corresponds to the integrated state. An example of the change of the internal state is shown in Figure 1.1. There are many different mathematical models available for HD cells, e.g. [RET96], and simulated HD networks were already used for robotic orientation [DLBA04]. However, in which way this information is correlated with other spatial information and precisely in which places the integrated HD state information is used in other parts of the brain is still up to discussion.

3 Allothetic information originates externally from the agent. Idiothetic information in contrast originates from within the agent. For instance, the vestibular system yields angular information of the agent's motion.

Apparently, the rodent brain is capable of solving the SLAM problem. Each time a new environment is entered it must be dynamically mapped when, for instance, an animal preys on food.

A SHORT SURVEY OF DIFFERENT SLAM METHODS

Many different technical solutions for SLAM were proposed. In fact, a review of the whole research area is not feasible. Instead, a list of selected methods is presented in Table 1.1 and continued in Table 1.2. Note that the Tables span over multiple pages. The list does not claim to be complete but highlights approaches to SLAM that use differing internal mechanisms.

Relating the presented methods to a biological context is difficult. Not every method nor its internals have a direct correspondence in the biological realm. Nevertheless, Figure 1.2 gives an overview how the models relate to each other and in which way they can be viewed in neurophysiological terms. Not every model establishes a full system for visual navigation but details only a small number of components though. For example, *MoreVision* is an overview of different visual processing methods. In contrast, *RatSLAM* is an implementation of a biologically inspired model which encompasses many of the items of Figure 1.2. The Figure demonstrates that SLAM research covers a broad range of different topics. However, there are still topics left open which could directly lead to an improvement of different models.

According to Figure 1.2, there is no direct link from visual odometry back to the sensory system. The question arises if such a feedback would improve the accuracy of self-motion estimates.

In this thesis I suggest a biologically inspired model which improves self-motion estimates with the help of feedback. The main source of input to the model is visual data. In addition to estimating the self-motion, it investigates the possibility of fusing different estimates within a very simple model of head direction cells. Thus, it tries to answer the following primary questions: Is it possible to fuse sensory information in a very simple head direction cell network? Will the fusion improve the quality of estimates? Is a feedback from this head direction network to upstream areas beneficial to the estimation process? To answer these questions, a complex model of self-motion estimation needs to be implemented. All individual parts have to be analyzed separately and in combination.

The proposed model has a versatile impact. For instance, it may be used on robotic platforms that are used for life-long service. Opposed to other models, it does not constantly increase its memory consumption over time. In addition, the model hints to possible connections in the rodent and primate brain. The findings may guide neurophysiological research to investigate the dependency of different cortical areas.

Chapter 2 will detail the proposed model. After giving an overview of the complete structure, each submodule will be detailed. In addition, error metrics that are required to analyze the model's behavior will be presented. The results are exposed in Chapter 3: at first, examinations of different individual modules are given, then of the whole model. Subsequently, Chapters 4 and 4.2 will give

Table 1.1: Comparison of different technical SLAM approaches. Note that the Table spans over multiple pages. Abbreviations and used terms: *IMU* – Inertial Measurement Unit; *special* world – an indoor environment explicitly designed for the system.

Name	Sensors	World	Method
GammaSLAM [MHB+08]	two stereo cams, IMU	outdoor	Rao-Blackwellized particle filter; stores height variances to a grid map
FrameSLAM [KA08], [AKB08], [KAS07], [EMC09]	stereo, applicable to mono	indoor, outdoor	estimates frame-to-frame motion; calculates nonlinear optimization problem (least squares)
RatSLAM [MW10]	mono	indoor, outdoor	uses place cells and head direction cells (named PoseCells) and distinguishable views to code for a place
View Graphs [FSG+97]	mono	special	creates undirected graph from distinguishable views
Geometric Module [SCS+09]	mono	virtual	similar to RatSLAM, but uses grid cells and place cells
Brain-Based [KSN+05]	mono, infrared	special	neural simulation of different parts of the human brain
Vision-Based SLAM [LBJL07]	mono, stereo, IMU		stereo: Harris Features, 6DoF; mono: delayed matching (feature position calculated by multiple frames)
Bayesian Surprise [RD09]	8-camera rig	indoor	Rao-Blackwellized particle filter that uses Bayesian Surprise for landmark detection

Table 1.1: Comparison of different technical SLAM approaches. Note that the Table spans over multiple pages. Abbreviations and used terms: *IMU* – Inertial Measurement Unit; *special* world – an indoor environment explicitly designed for the system.

Odometry	Visual Processing	Comment
visual odometry by feature tracking	corner detector; estimates ego-motion by 3D feature positions determined by stereo cams	in contrast to regular grid maps: does not assume flat surfaces or that a cell is free or occupied
visual: stereo matching	stereo matching; CenSurE features	stores subset of frames (*skeleton*) as constraints for least squares
wheel encoder or simple visual	row-wise cross correlation of input images	stores experiences in a graph similar to *View Graphs*, but with distance information
	cross correlation of horizontal pixel line	does not store distances, nodes contain only information about views
"self-motion input"; assumed: Exact value	Gabor filter bank, cross correlation of their output	place cells are recruited during learning
wheel encoder	neural modeling: V1/2/4; Gabor filters	
odometry by visual motion	stereo: Harris Detector, cross correlation, interest point group matching; uses mono for feature initialization; computes visual motion by feature tracking	uses the Extended Kalman Filter for mapping
	features: Harris, Maximally Stable External Regions (MSER); Multivariate Polya Model	builds topological map with particle filter

Table 1.2: Continuation of Comparison Table 1.1. Note that the Table spans over multiple pages. Abbreviations and used terms: *IMU* – Inertial Measurement Unit; *special* world – an indoor environment explicitly designed for the system.

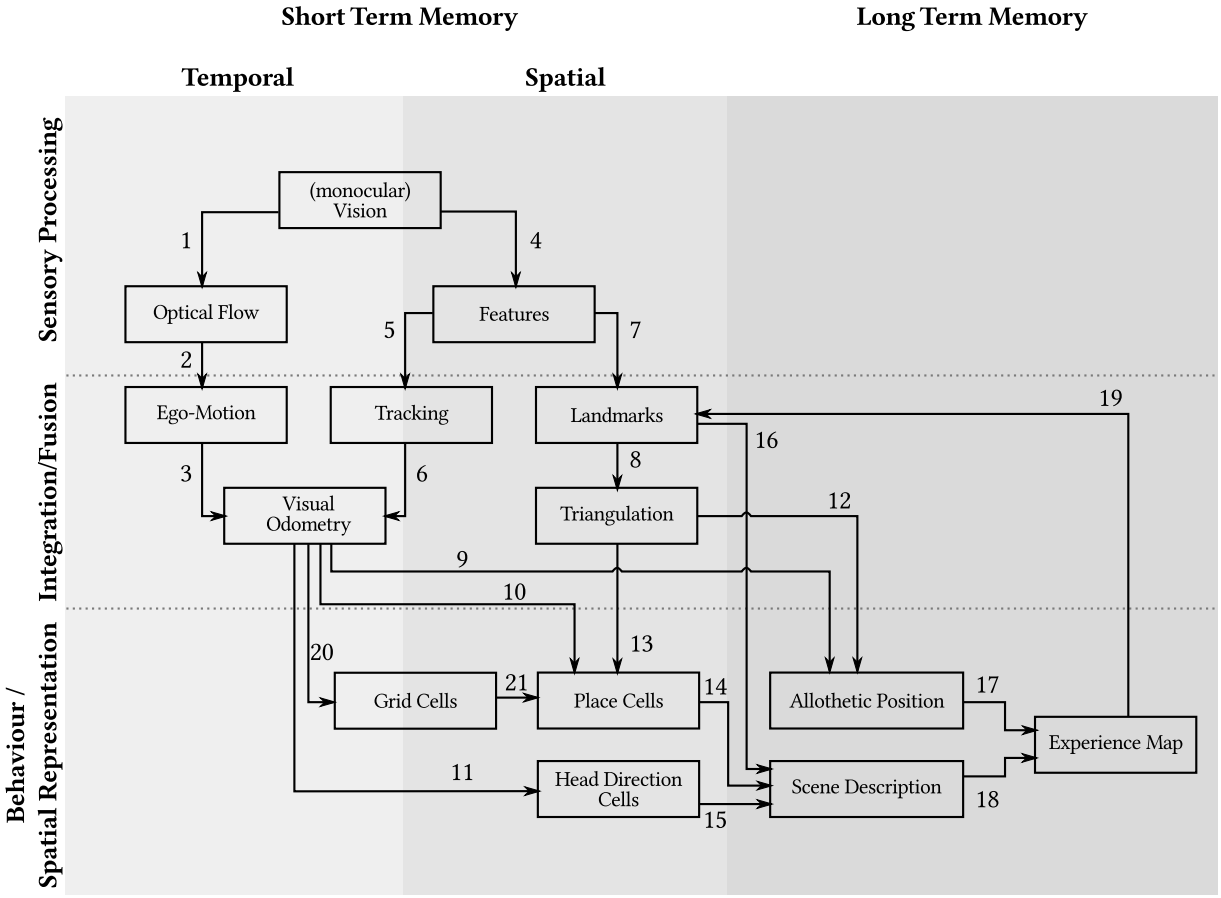
Name	Sensors	World	Method
VecSLAM [SK09]	Laser Range Finder	indoor	line segments, dubbed vectors, are tracked and used for SLAM
Parallel [KM07]	mono	indoor	inspired by bundle adjustment and structure from motion; builds feature map with help of RANSAC
More Vision [LLB08]	mono, stereo	indoor, outdoor	
CV Methods [MRS06]	stereo	outdoor	Iterative Closes Point (ICP)
Artificial [BKS03]	mono	indoor	uses artificial landmarks to estimate position
Small Body [JCM00]	mono, stereo	space	combination of feature tracking and triangulation by landmarks
Markerless [LH09]	mono	indoor	paper doesn't exactly tell how to get from mapped features to pose estimate

Table 1.2: Continuation of Comparison Table 1.1. Note that the Table spans over multiple pages. Abbreviations and used terms: *IMU* – Inertial Measurement Unit; *special* world – an indoor environment explicitly designed for the system.

Odometry	Visual Processing	Comment
"from robot (10% error)"		vectors are aligned in a global coordinate system; uses recursive least squares for vector merging
	Fast-10 corner detector, Shi-Tomasi features	is more concerned about Augmented Reality than robotic SLAM
	Harris, SIFT, Line Features, Plane Features, First Order Geometric Primitives	paper is mostly about vision techniques for SLAM
ICP	feature tracking (Shi-Tomasi) based on cross correlation	paper is about visual odometry for SLAM
	determines region of interest with artificial landmark and computes the landmark extents to guess distance	
visual (feature tracking)	Benedetti and Perona; Shi-Tomasi features; compares detected landmarks against database	needs human preprocessing of database
	Handy AR, SIFT, Shi-Tomasi; Lucas-Kanade on tracked SIFT features	builds 3D reference frame from initialization

insight into the results, why they emerge and afterward summarize the findings. In addition, their relation to biological models will be discussed. Then, the findings will be related to the initial research questions of this thesis. Finally, Chapter 5 will point out new research that is possible with the new model.

Figure 1.2: *(next page)* SLAM algorithms and their relation to a biological context. Note that not all SLAM approaches listed in Tables 1.1 and 1.2 were included because not every approach can be mapped to a biological overview. In addition, the methods listed in the legend do not necessarily name their computational steps similarly. Hence, the submodules and computational steps were regarded as matching if their main purposes reflect resembling tasks. The Figure contains steps which are not used in the referenced technical SLAM literature. For example, steps 1-3 are not employed. However, such steps are described in literature about biological models that are not necessarily related to technical SLAM. For instance, steps 1-3 are reflected in [Per92].



Step	Implemented in
4	GammaSLAM, FrameSLAM, Vision-Based SLAM, Parallel CV Methods, Artificial, Small Body, Markerless
5	GammaSLAM, FrameSLAM, Vision-Based SLAM, Parallel CV Methods, Small Body, Markerless
6	GammaSLAM, FrameSLAM, Vision-Based SLAM, CV-Methods, Small Body
7	Vision-Based SLAM, Parallel, Artificial, Small Body, Markerless
8	Parallel, Artificial, Small Body, Markerless
10, 11	RatSLAM
12	Parallel, Artificial
14	RatSLAM, Geometric Module
15	RatSLAM
16	View Graphs, Bayesian Surprise, Geometric Module, Parallel, Markerless
17	Parallel
18	GammaSLAM, FrameSLAM, RatSLAM, View Graphs, Bayesian Surprise, Geometric Module, Parallel, Markerless
19	Vision-Based SLAM, Parallel, RatSLAM, View Graphs, Bayesian Surprise, Geometric Module
20, 21	Geometric Module

2 BUILDING BLOCKS OF THE FEEDBACK DRIVEN MODEL FOR EGO-MOTION ESTIMATION

Chapter 1 shows that the problem of simultaneous localization and mapping is intensely researched. Yet there are different areas open to investigation, especially when the models are viewed from a biological context. For example, storing an experience map[4] needs to have a strong influence from behavioral systems and learning theories, because experiences might be perceived as either good or bad. Bad experiences could have the effect of a repellor, leading to the avoidance of certain areas in an animal's habitat. To be able to avoid hazardous areas, a precise knowledge about their positions is necessary.

Problems in existing models like RatSLAM and recent findings in the rodent brain suggest that exact positional information is of high relevance. I was able to show that – besides other issues with RatSLAM[5] – the experience map would benefit from more accurate estimates of the agent's position. One of the internal mechanism of RatSLAM to maintain the experience map would highly profit from more exact estimates of the agent's position. The procedure, which is based on a path relaxation algorithm, causes the experience map to degenerate over time when there are not enough experiences that can be linked together. Such experience links are created when the agent travels from one experience to another. Having more precise estimates of the position would reduce the impact of the relaxation algorithm or make it possible to completely avoid it.

As a consequence, I propose a new model of ego-motion (or self-motion) estimation which will improve the quality of the estimates and, as a an effect thereof, position estimates. The model suggests a feedback from a head direc-

4 An experience is a (directed or undirected) graph which acts like a map. A map entry may consist of different details. In RatSLAM, each experience map entry contains information about the most active PoseCell, a scene description and links to other experiences in the graph. A PoseCell is the combination of place and head direction cells.

5 The thesis was supposed to be an extension to RatSLAM. However, the implementation which is provided on `http://ratslam.itee.uq.edu.au` doesn't follow the referenced RatSLAM literature. In fact, some of the most essential steps described in [Mil08] are missing. Hence, I implemented RatSLAM, precisely following the literature. The system did not work as expected and an investigation of the model revealed several problems. Consequently, it was necessary to detach the thesis from RatSLAM and build the model from scratch.

tion cell network to upstream areas. Although this is biologically inspired, such a feedback was not yet shown to exist in neurophysiology.

After giving a general overview of the model in the first section, the following ones delve into details of specific modules. If not otherwise specified, the software used for the different building blocks was written by myself. In addition to the modules, it was necessary to write different tools which are described in the appendix, Chapter C.

2.1 MODEL OVERVIEW

The model is based on the idea that a feedback from a head direction cell network could be profitable for upstream cortical areas. For instance, cortical area MST (Medial Superiour Temporal) is believed to play a major role in the estimation of ego-motion (or self-motion) from optical flow. Optical flow contains both a rotational and a translational component which give hints to the self-motion. With the help of a feedback or prediction, certain parts of the optical flow could be removed previously to extracting an ego-motion estimate in MST. Consequently, it should be possible to analyze the remaining flow with a higher precision while the number of computations required stays constant.

Template models, like the one described in Section 2.4, require a lot of computations to calculate estimates for the translational and rotational part of optical flow. A feedback in the form of a rotational prediction could help to increase the precision of such models while leaving the number of required computations constant – or even reduce it. For example, estimating the rotational component of optical flow with the help of the template model of Section 2.4 usually requires to sample optical flow that corresponds to a range of possible rotations. This is at best done linearly in order to prevent the miss of a certain rotation. It would, however, require more computations when a higher precision is wanted. Using a feedback which predicts the next movement, the rotational component of the optical flow could be removed almost entirely. The remaining rotational part would reflect the error of the feedback signal, but it could be analyzed with a very high precision while the number of computations is not increased.

Hence, I propose a model for self-motion estimation based on the idea of such a feedback. An overview of the model is given in Figure 2.1.

2.1.1 MODEL DESCRIPTION

At the beginning of the model, visual input is forwarded to two separate processing streams: 1) *template model* path and 2) *epipolar* path. They are named after their main components of ego-motion estimation.

The *template model* path estimates optical flow from the input data using a template model of cortical area MST. Flow estimates are proximately passed to the template model. The template model establishes an estimate of the agent's motion from optical flow. Although the path produces a translational and a rotational estimate, analysis of the model that utilizes a subsequent head direction network uses only the rotational estimate. Due to restrictions of the

template model, the translation is a direct consequence of the estimated rotation. The estimate of the agent's translation is produced with the help of the head direction network later in the model.

The other path, which is called *epipolar* path, detects and tracks features between frames. The method to detect and track features is CenSurE. Feature positions can be used subsequently to reconstruct the translation and rotation between two frames using epipolar geometry. The processing path issues an estimate of both the rotation and the translation.

The integration, or fusion, of both rotational estimates of the preceding paths is the next step. The two different streams produce two confidence measures, which have to be taken into account while merging the estimates. This is followed by forwarding the fused value to the head direction network. Figure 2.1 shows the fusion module as a dashed box because the fusion may be an extension to the head direction network.

The head direction value, which can be extracted from the head direction cells, is used to form an estimate of the agent's translation. It is combined with the translational estimate coming from the *epipolar* path. Focus was not set onto this part, thus simply a weighted average is calculated. The weights are the confidence values which were already used for the fusion of the rotational estimates.

For the next iteration of the estimation process, a prediction signal is applied to the head direction network. The difference between the head direction state before and after the prediction signal is used as a feedback into the preceding computational steps.

The prediction signal could come from different sources. One possibility is an inertial measurement unit that responds quickly to changes of position. Hence, it can be applied to the head direction network before ego-motion estimation in upstream areas is carried out. Different idiothetic cues were discussed, e.g. in [JS98]. Another possibility is to iterate the network twice: first without feedback and a second time with feedback. The latter way requires upstream areas to be remodulated on-the-fly so that they can cope with the different demands. Differences of the *template model* path with and without feedback are detailed in Section 2.4.3. In the implemented model, normally distributed noise was added to the ground truth data to form the prediction signal. The exact specification is given in Section 3.4.

To help the orientation, the following sections will have miniature versions of Figure 2.1 drawn to the page margins. The discussed submodule(s) will be highlighted in terms of color. Note that the feedforward lines of the confidence values were removed in order to make the miniatures less crowded.

2.1.2 MODEL RESTRICTIONS

In order to quantify the results without having too many parameters to set and analyze, there were some constraints imposed on the model. It is important to note these constraints as they have an influence on the settings and computations of the submodules. The constraints are:

- The agent travels on a curvilinear path.

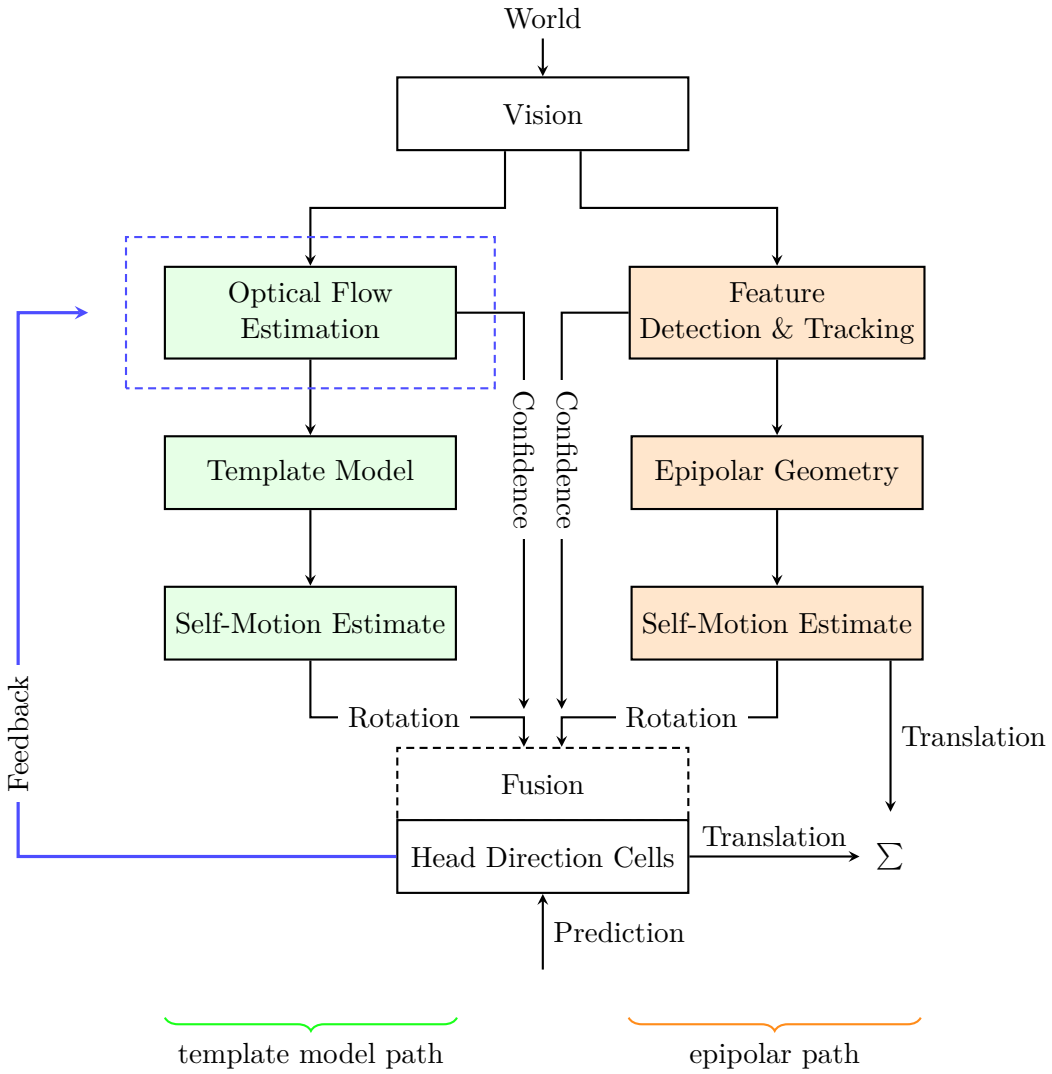


Figure 2.1: Model Overview. The model can be logically split into two processing paths due to their primary modules for ego-motion estimation: 1) *template model* path and 2) *epipolar* path.

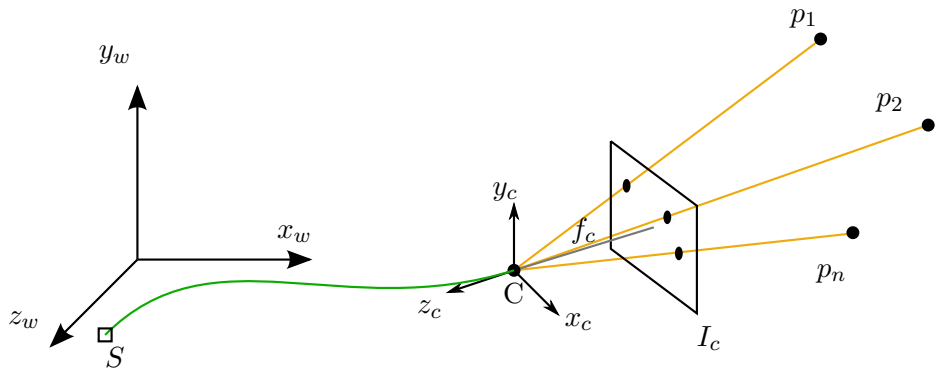


Figure 2.2: World and camera setup. The agent is restricted to move parallel to the xz plane. The pinhole camera is defined by a camera coordinate system with origin C and focal length f_c. The camera is looking along its negative z axis. Objects (points $p_1, \ldots, p_n \in \mathbb{R}^3$ in the image) are projected perspectively onto the image plane I_c. The path on which the camera C is moving is curvilinear. In the image, this path is colored green and starts at S.

- The curvilinear path is traversed tangential to the path.
- Movement is restricted to the xz plane, where x points to the right and z points backwards. The y axis points upwards. Thus, a right-handed coordinate system is used.
- The only allowed rotations are around the y axis. Such rotations are often referred to as yaw rotations in the literature.
- The agent moves through a rigid scene, where no other objects are moving.
- The agent moves with a constant speed of $1\frac{m}{s}$.

2.2 MODEL ENVIRONMENT AND INPUT DATA

In order to investigate the different parts of the model, both analytical and virtual input data was used. Analytical data is, for example, optical flow that was generated for a certain camera movement but without a real structure that generates the flow. Virtual optical flow, on the other hand, is computed for a virtual environment that was modeled using the software `blender`.

The agent travels in an environment where the ground plane is defined by the x and z axis, the up-vector of the world is the y axis. This setup and the configuration of the camera is shown in Figure 2.2. The calculations in the following sections of this chapter assume a right-handed coordinate system.

2.2.1 ANALYTICAL INPUT DATA

FLOW

Analytical optical flow was calculated according to [LP80]. Point coordinates were selected so that they form a regular grid on the retinal plane. Depth values were selected randomly in the range $[0.5, 30]$ m. Figure 2.3 shows examples of analytical optical flow for pure translation, pure rotation and the combined result.

FEATURES

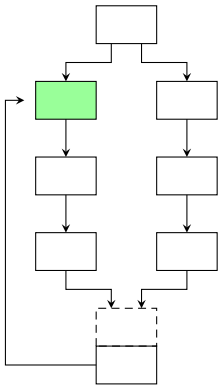
In order to create analytical features, 1000 points were randomly scattered in the range $x \in [-25, 25]$, $y \in [-10, 10]$ and $z \in [-25, 25]$. The resulting point cloud is shown Figure 2.4a. The agent travels inside the point cloud, its maximal distance from the center is 7.5 m. Consequently, there are about 50 to 150 points projected onto the image plane per view. This number resembles the amount of features found in a building of the Ulm University when using CenSurE. CenSurE is described in detail in Section 2.5. Point correspondences can be calculated from the analytical feature positions for two different views. A correspondence example is shown in Figure 2.4b.

2.2.2 VIRTUAL INPUT DATA

In addition to analytical data, I modeled a virtual scene of an office building in `blender`. The textures and 3D models were placed in order to yield a certain number of features when using CenSurE. The goal was to achieve approximately the same number of features for both real world footage from inside the building of Ulm University, and the virtual scene.

The agent's trajectory inside the virtual environment is a circular path with radius $r = 7.5m$. The scene was rendered with `blender`'s default rendering engine and a frame rate of 10 Hz. Each frame was subsequently stored in OpenEXR files which contain the depth information in addition to the image. Hence it it possible to compute the ground truth optical flow for the complete scene with `exrflow`. More details about `exrflow` can be found in the Appendix, Section C.2.

In addition to ground truth optical flow, flow estimated with the help of a biologically inspired algorithm is used. Examples for both, ground truth and estimated optical flow, is shown in Figure 2.6, the estimation algorithm is described in the next section.



2.3 OPTICAL FLOW ESTIMATION

Optical flow is estimated using a biologically inspired recurrent estimator, initially described in [BN07] and henceforth called MotionAlgo. The implementation that was used in this thesis was written in advance to the thesis while I was employed as student research assistant at the Institute of Neural Information Processing at the Ulm University.

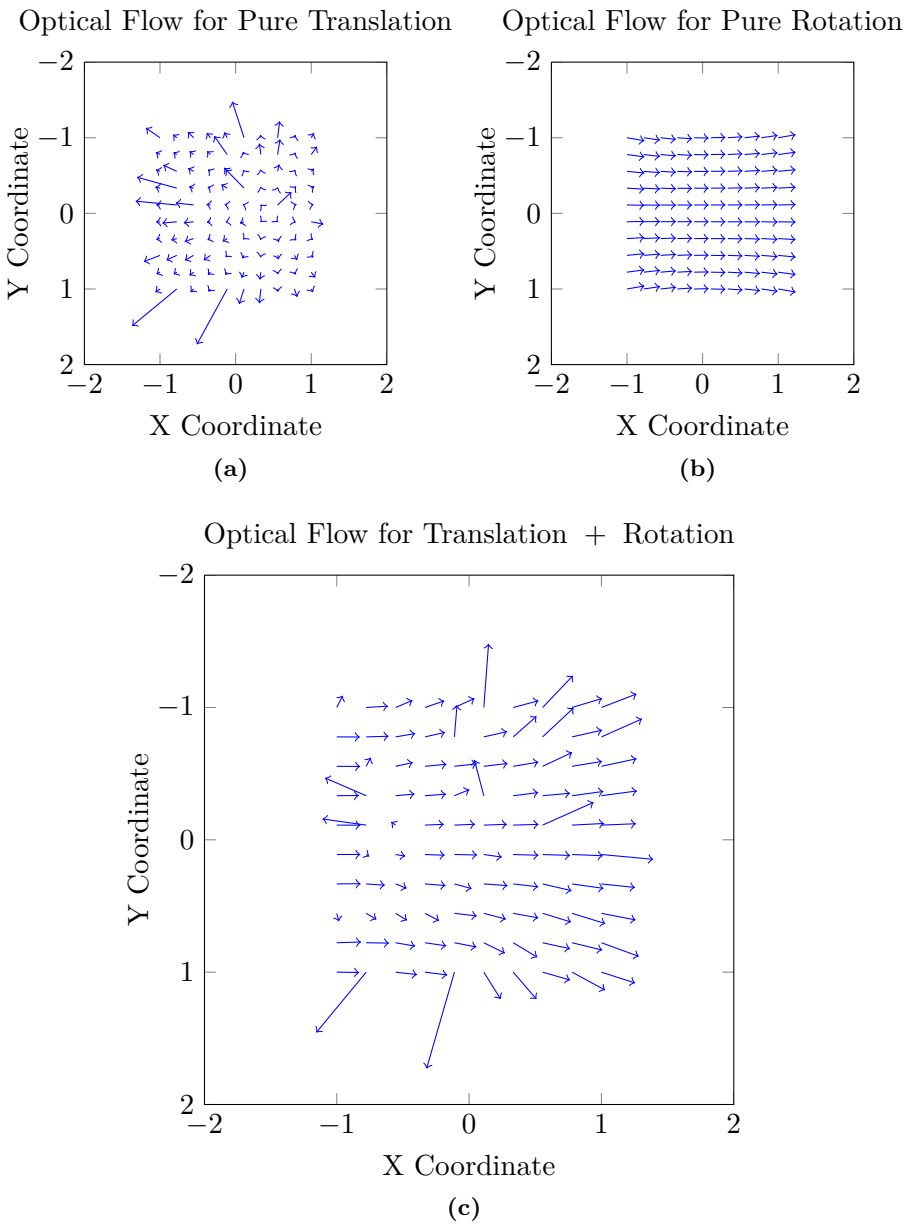
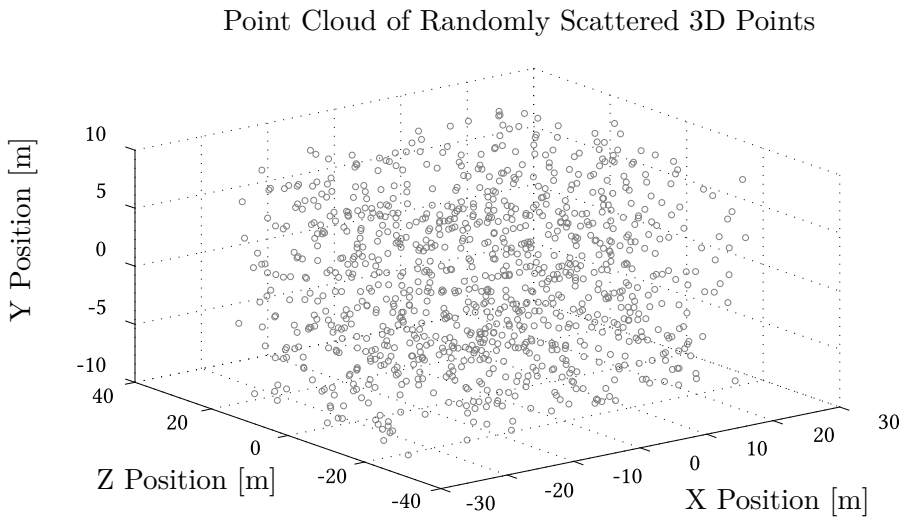
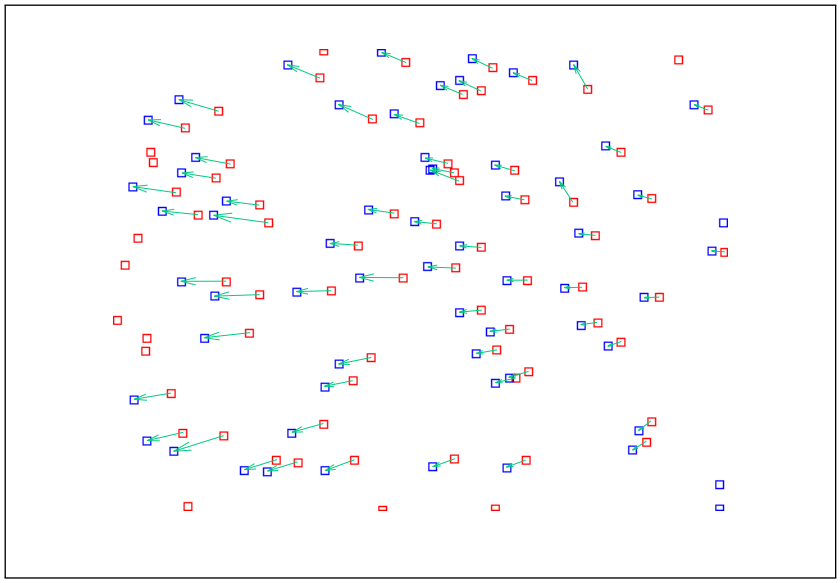


Figure 2.3: Analytical optical flow for a grid of size 10×10. The retinal plane has the upper left coordinate $(-1, -1)$ and the lower right coordinate $(1, 1)$. The plots are extended for visibility reasons. **(a)** Optical flow for a pure translation of 1 m forward. **(b)** Flow for a pure rotation of 5 degrees counter-clockwise around the y axis. **(c)** Flow of the combined movement from (a) and (b).



(a)



(b)

Figure 2.4: Analytical features. **(a)** Point cloud of 1000 features. The agent is moving inside this cloud. Its maximal distance from the center is 7.5 m. **(b)** Feature correspondences for two frames. Features of frame t are marked in red, features of frame $t + 1$ in blue. Their correspondence lines are shown in green. The agent's motion was a translation of 1 m forward, 0.2 m to the right and a rotation 2 degrees clockwise around the y axis. Note that not all features were trackable.

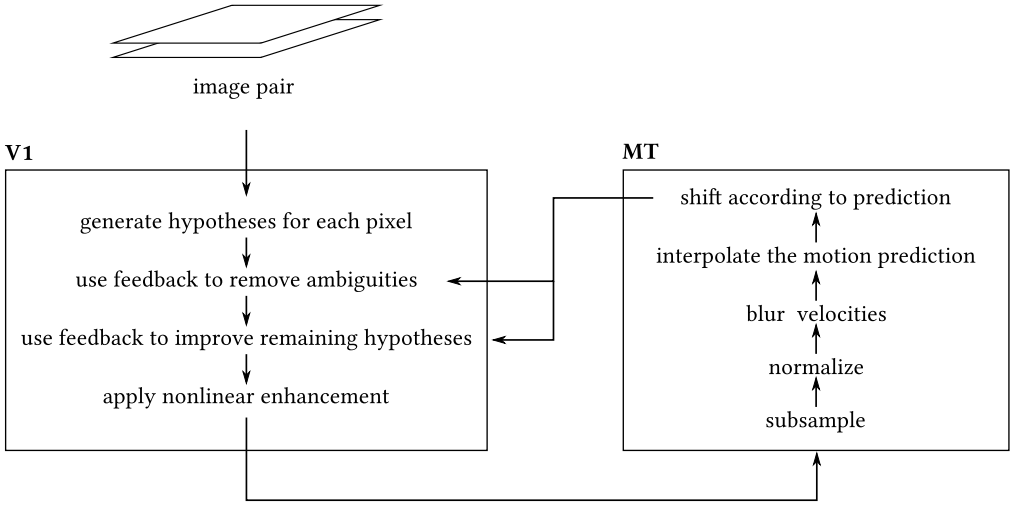


Figure 2.5: Overview of the MotionAlgo. Model area V1 estimates optical flow for each pixel of the input image. The resulting hypotheses are forwarded to area MT which integrates nearby areas. Hypotheses results of MT are subsequently used in V1 to remove outliers and support matching hypotheses. (cf. [BN07])

I will only give a rough outline of the algorithm, a profound description can be found in [BN07]. The MotionAlgo computes optical flow for two consecutive input images by simulating cortical areas V1 and MT. Results from V1 are projected to MT, which in turn feeds its estimates back to V1. In addition to the following paragraphs, Figure 2.5 gives a very short overview of the MotionAlgo.

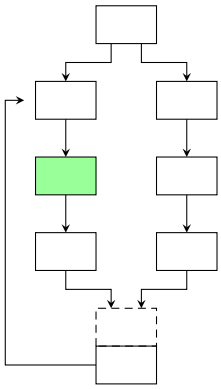
The first step in model area V1 is computing the Census Transform [ZZL94] of each pixel of the input image. This operation creates a neighborhood descriptor for each of the pixels. Subsequently, the descriptors of both frames are matched, which will result in a list of possible matches per pixel. Note that this list might be empty. Afterward, a selection process will ensure that there are only a fixed maximum number of so-called motion hypotheses left per pixel. A motion hypothesis is a vector containing the pixel location, a velocity vector and a weight. The weight gives information about the "believe, confidence or neural activity" [BN07] of the hypothesis. If there are too many hypotheses generated for one single pixel, the motion ambiguity is believed to be too high and all generated hypotheses are rejected for that pixel. The overall result for model area V1 is a sparse coding of possible motions in the input images.

Following the calculation of hypotheses on the full image scale in model area V1, the hypotheses are forwarded to MT. The hypotheses which lie in small receptive fields are integrated. This subsampling process yields data that is smaller than the input size. Hypotheses will be nonlinearly enhanced and a blurring operator is subsequently applied. This means that the velocity components of generated hypotheses influence each other. After the blurred hypotheses are normalized, they are shifted according to their velocity components: For example, take a hypothesis at location (x,y) with velocity information (v_x, v_y). The hypotheses will be moved to the position $(x+v_x, y+v_y)$ during the shifting process.

Finally, the shifted hypotheses provide feedback to model area V1, where they will assist in the correspondence matching process. In addition to contributing to this match selection, weights of hypotheses which were predicted according to the computation in model area MT are nonlinearly increased.

Readouts of model area MT are used as estimated optical flow. Figure 2.6 shows example images of ground truth optical flow and estimated flow. Note that the size of the readout of model area MT is smaller by a certain factor but is rescaled to match the size of the ground truth optical flow. All required setup parameters for the MotionAlgo are listed in Table A.1.

2.4 TEMPLATE MODEL FOR EGO-MOTION ESTIMATION



Two decades ago, Perrone and Stone proposed a model to simulate cortical area MST ([Per92], [PS94], [PS98]). MST is believed to compute self-motion from optical flow which was previously estimated in upstream cortical areas. They suggested to describe the processes in cortical MST with the help of so-called template neurons. This kind of neuron responds only to optical flow that corresponds to its very tuning. Take, for instance, a neuron which encodes for a translation with certain speed and direction to the right and a simultaneous rotation to the left. It will only respond maximally if the optical flow contains components for the exact translation and rotation of its tuning.

The source code for the *template model* was provided by Florian Raudies. The implementation does not exactly follow the methods presented by Perrone and Stone but is made to work for curvilinear motions.

A template neuron needs to consider the whole input flow. The reason is the aperture problem which is illustrated in Figure 2.7. If the neuron took only a small portion of the flow into account, it might not necessarily yield the correct response. For instance, a neuron which codes for a movement towards the top-right might discharge maximally for the flow presented in Figure 2.7a although the motion which underlies the Figure is a rotation around the z axis. Hence, a template neuron needs to extract the precise information from the entire optical flow for its response.

Each template neuron integrates the responses of motion detectors. Each detector responds according to the fitting of its speed and direction tuning to the input flow. The direction tuning of a detector is defined as

$$O_d(\Delta_\varphi) = \frac{\exp\left(-0.5(\frac{\Delta_\varphi}{\sigma_r})^2\right) - \delta}{1-\delta} \quad . \tag{2.1}$$

Δ_φ is the signed angular difference between the detectors preferred flow and the input flow: σ_r influences the fuzziness with which the detector accepts diverging input flow as matching. The preferred flow for a motion detector is defined by the template neuron's preferred motion and calculated according to [LP80]. The detector's speed tuning is defined as

$$O_s(\Delta_s) = \exp\left(-0.5(\frac{\Delta_s}{\sigma_t})^2\right) \quad , \tag{2.2}$$

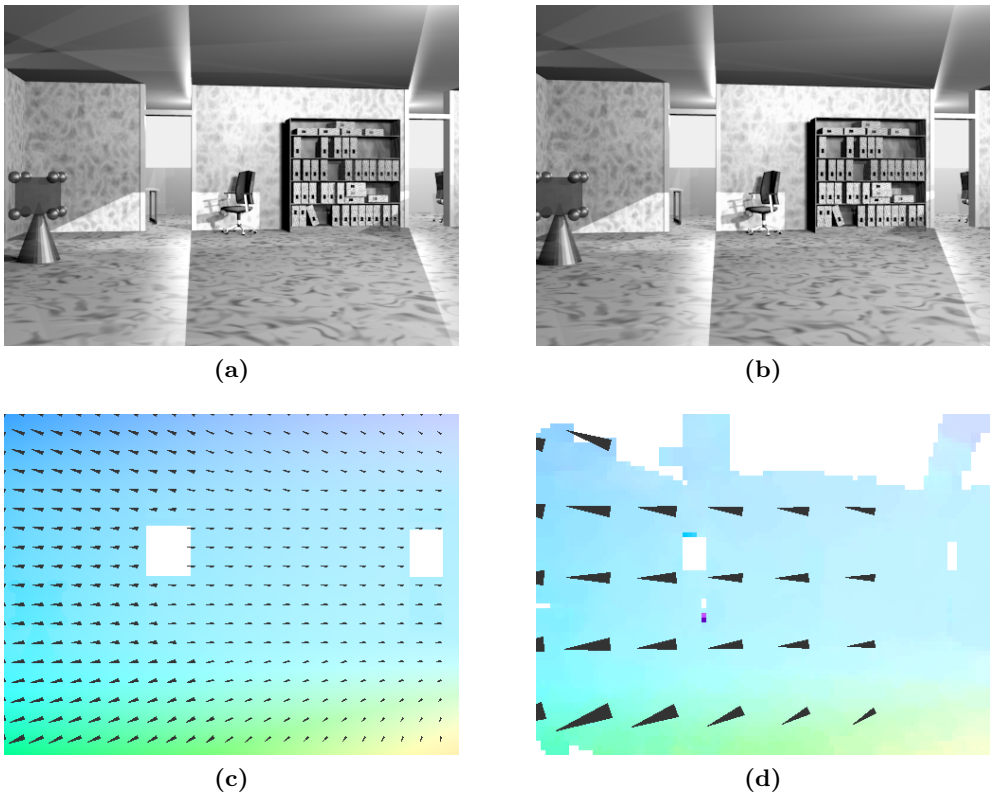


Figure 2.6: **(a)** Frame 10 and **(b)** Frame 11 of the office building sequence. **(c)** Optical flow generated with `exrflow`. **(d)** Optical flow estimated with the MotionAlgo. Note that the output size of the MotionAlgo is smaller than the original input size. However, the output is rescaled to match (c). As a consequence, the flow vectors in (d) are larger than those in (c). Colors are coded according to the Middleburry color code [BSL+07]. Note that the MotionAlgo could not estimate optical flow for the office ceiling because the ceiling-texture produces too many ambiguities. The white spots in (c) are due to infinite depth of the input data at these positions. There was no structure but windows giving view to simulated sky.

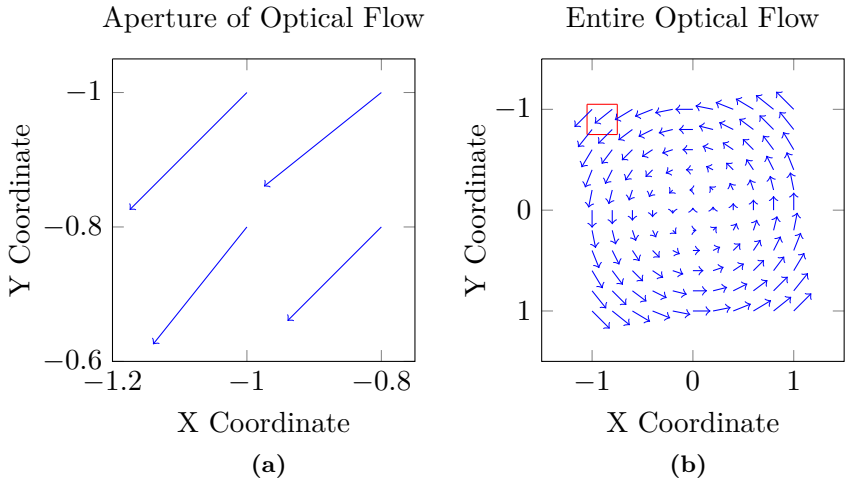


Figure 2.7: Aperture problem. **(a)** shows a small part of flow. The motion coded in the four vectors might be a translation towards the top right. However, **(b)** shows that the flow of originates from a rotation around the z axis. (a) is marked in the red rectangle.

where σ_t serves the same purpose as σ_r in Equation 2.1. Δ_s is the difference between the input flow speed and the preferred speed of the motion detector and defined as

$$\Delta_s = log_2\left(\frac{r}{R}\right) \quad . \tag{2.3}$$

In this equation, R is the detector's preferred speed tuning and r the actual speed of the input flow.

The motion detectors for one template neuron are tuned exactly to the neurons preferred motion. Thus it is necessary to have different template neurons, each having a set of distinct motion detectors, to estimate the agent's translation and rotation. For instance, a template neuron which codes for a rotation 12 degrees to the right employs motion detectors that reflect the optical flow produced for exactly this motion. In order to have a neuron which codes maximally for 13 degrees to the right, a different set of motion detectors is required for this neuron.

O_d and O_s depend on the depth of the scene. The depth of a scene is inversely proportional to the speed of the optical flow. Because the depth of the world is not known in MST, different depths need to be sampled. As a consequence, each of the described motion detectors needs to be created for each depth that shall be sampled.

Finally, the response of a template neuron j is defined as

$$R_j = \frac{1}{m}\sum_{i=1}^{m}\left(max\left\{O_d(\Delta_\varphi)O_s(\Delta_s)\right\}_{k=1}^{n}\right) \quad , \tag{2.4}$$

where n is the number of depth samples. The implementation creates motion detectors on-the-fly, m is thus the number of flow vectors.

Table 2.1: Parameters for the template model.

Template Model Setup	
δ	0.05
σ_t	0.5 [m]
σ_r	30 [°]
focal length for template flow generation	2.18
sampling of the depth	$2, 4, 6, 8, 16, 32, 48, 64$ [m]
sampling of the azimuth angle	0 [°]
sampling of the rotation angle	$-35, \ldots, 35$ [°]

Due to the model constraints, only yaw rotations[6] are considered. In addition, sideways movement of the agent is ignored and hence, sampling of the azimuthal angle can be set to zero. The results of Chapter 3 were produced with the parameters defined in Table 2.1.

2.4.1 INTERPOLATION OF THE RESPONSE FIELD

Simply taking the rotation for which the maximally responding neuron codes will yield problematic results. Depending on the sampling of the rotational angle, there might be a systematic over- or underestimation introduced. Regard the following example: sampling is set to be linear from -10 to 10 degrees and the distance between two sampling points is 1 degree. When the exact rotation is 4.6 degrees, the maximally responding neuron most probably codes for 5.0 degrees. As a consequence, the model will over-estimate its own rotation by 0.4 degrees. Thus it is essential to interpolate the response field to get a proper estimate. The response field is the entirety of the template neurons after their response was calculated. Due to the model constraints, this field is one dimensional and only codes for rotational values.

I formulated and implemented the following four of the different possibilities to interpolate the response field. A schematic of the interpolation methods is illustrated in Figure 2.8.

- SIMPLE
  Interpolation is done by taking the position of the maximally responding neuron $n_{\max}$ and integrating over a certain area around that position. The dimension of the interpolation area $\Omega_{n_{\max}}$ is chosen to be approximately 17% of the response field around $n_{\max}$. This value is chosen randomly and results in taking an additional six neurons in each direction of the response field to calculate the interpolated response. The lower and upper bounds l and u for the one dimensional area $\Omega_{n_{\max}}$ need to be calculated carefully in order to avoid problems at the edges of the response field. Thus, $\Omega_{n_{\max}}$ contains all neurons that surround $n_{\max}$.

6 A yaw rotation is the rotation around the agent's y axis

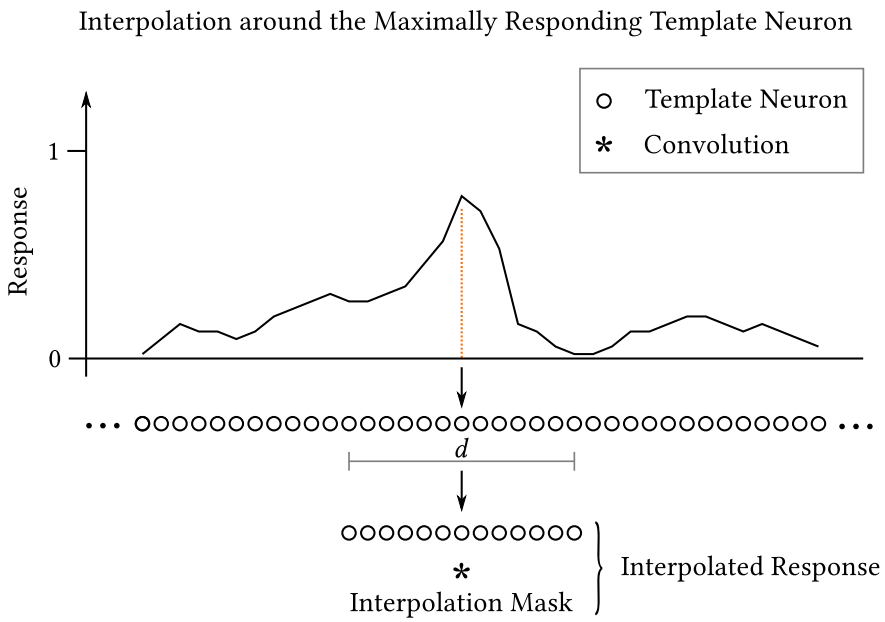


Figure 2.8: Schematic of the interpolation around the maximally responding template neuron. The width d defines the number of template neurons that are interpolated. d is set to 17% of the response field for the methods *simple*, *gauss near* and *DoG*. Method *gauss full* interpolates the entire response field.

Let $\Phi(\alpha) = (\sin(\alpha), \cos(\alpha))^t$ be the function that takes an angle α to its vectorial representation and Φ^{-1} its inverse. Let α_n be the rotational tuning of a neuron n, and w_n its neural activity. The interpolated estimate γ can henceforth be described in terms of a discrete integration

$$\gamma = \Phi^{-1}\left(\sum_{n\in\Omega} w_n \Phi(\alpha_n)\right) \quad . \tag{2.5}$$

- GAUSS NEAR
  In contrast to the *simple* interpolation method, the *gauss near* method smooths the neural activities w_i in the area $\Omega_{n_{\max}}$ with a Gaussian. Let ψ_n be the neural activity of neuron n after discrete convolution of all activities w_i in the area $\Omega_{n_{\max}}$ with a Gaussian g_σ, $\sigma = 1.5$. Consequently, γ is defined as

$$\gamma = \Phi^{-1}\left(\sum_{n\in\Omega} \psi_n \Phi(\alpha_n)\right) \quad . \tag{2.6}$$

- DOG
  The *DoG* interpolation uses a Difference of Gaussian d instead of the Gaussian of method *gauss near* to convolute neural activities w_i in the area $\Omega_{n_{\max}}$. d is defined as

$$d = 2g_{\sigma^1} - 1.5g_{\sigma^2} \quad , \tag{2.7}$$

  where $\sigma^1 = 1.0$ and $\sigma^2 = 1.5$. γ is finally determined according to Equation 2.6.

- GAUSS FULL
  Instead of using only a small area $\Omega_{n_{\max}}$ of the response field, the *gauss full* interpolation uses the whole response field for Gaussian smoothing of the neural activities. The interpolated estimate γ is calculated according to Equation 2.6.

2.4.2 SUBSAMPLING FROM MT TO MST NEURONS

The number of flow vectors used to compute the response can be set independently. However, using a large number of flow vectors may lead to computational limits. The memory consumption and processing power required to calculate the *template model* response is quite large. Thus it is desirable, with respect to computational cost, to have a small number of flow vectors.

Although Perrone showed in [Per92] that a higher number of flow vectors results in a smaller error of translation estimates, I analyzed different input sizes for rotational estimates. It may be preferable to have a small number of flow vectors in certain cases. The results are described in detail in Section 3.2.2. In combination with the computational limits it is necessary to subsample a larger number of flow vectors to a smaller number. For instance, the input images of the virtual input sequence are of size 480×360. Although the MotionAlgo already reduces this size by a factor of 5 emitting 96×72 flow vectors, decreasing the number of vectors may still be required. In addition to this example, subsampling is especially needed for ground truth optical flow: `exrflow` computes the optical flow for the full size of the input image.

I analyzed four different ways to subsample flow vectors. The basic methods are computation of the mean or median flow vector. Both methods were investigated in a standard manner and, inspired by CenSurE, with overlapping interpolation areas. Mean and averaging are very common in computer vision task. The median vector was recently shown to be one of the best operations in optical flow processing available [SRB10].

- MEAN
  mean subsampling is effectively like a box filter applied to the input flow vectors, but reduces simultaneously the number of flow vectors. The procedure can be visualized by placing a coarse grid over the input data and averaging the vectors in each cell. Figure 2.9a depicts the method.

- MEDIAN
  In order to calculate the *median* vector, the flow vectors' angular representation will be calculated first. Subsequently, a median vector with respect to the angular value can be selected. The length of the resulting vector is the average vector length of all considered vectors. Its position is identified with the position of the vector that has the median angular value.

- MEAN OVERLAPPING and MEDIAN OVERLAPPING
  Inspired by CenSurE, I implemented the methods *mean* and *median* subsampling additionally with overlapping areas. Each cell is extended by

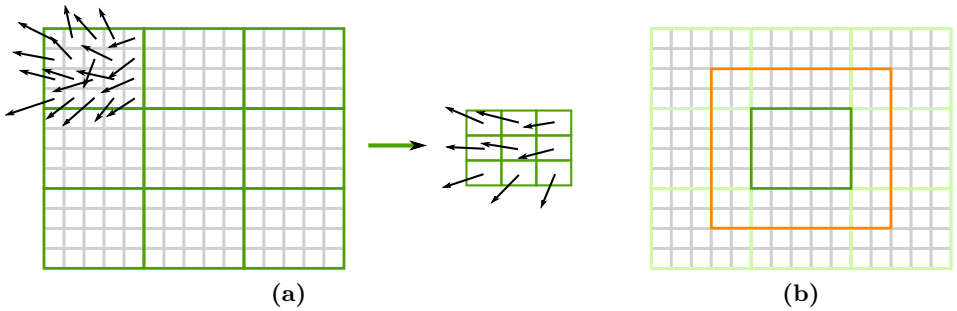


Figure 2.9: Subsampling of flow vectors. **(a)** illustrates the subsampling of 180 to 9 flow vectors. In order to reduce the graphical complexity, only the flow vectors of the first subsampling cell are shown. *mean* subsampling calculates the mean vector of a cell whereas the *median* method selects the median angle and the mean length. **(b)** displays the difference between the normal subsampling cell and overlapping subsampling areas. The orange cell extends the green cell by 40 % in each direction.

40% in each direction, mean and median vectors are computed for the extended cells. The difference between the "regular" methods and the methods with overlapping areas is drawn in Figure 2.9b.

2.4.3 SAMPLING OF ROTATIONAL ANGLES IN THE STANDARD AND FEEDBACK MODEL

There is a difference in rotational sampling between the model which does not use feedback for the calculation and the model which uses feedback. The feedforward-only model linearly samples in a certain range of rotations. In contrast to this, the feedback driven model samples only loosely for higher rotations but densely around zero degrees. Figure 2.10 displays the sampling points for a normal sampling and a dense sampling for rotations in the range $[-35, 35]$ degrees. The justification for the different sampling points is the fact that the feedback driven model will receive different input than the feedforward-only one. Optical flow for the feedback-model will be modulated so that the rotational component is already removed up to a certain error. The remaining rotational part is assumed to be very small and can be analyzed with a higher precision. Outliers need to be taken into account, therefore sampling of higher rotational values is needed.

To be able to compare the results of the model with and without feedback, the number of sampling points was set to 71. Rotational sampling was designed to lie in the range $[-35, 35]$ degrees. To generate sampling points for the dense sampling around zero, the function $y(x) = \exp(\sigma x) - 1$ with $\sigma = 0.125$ was used. Input x is the sampling index, $y(x)$ must be re-normalized in order that $y(m) = m$. The whole procedure is presented in Algorithm 1.

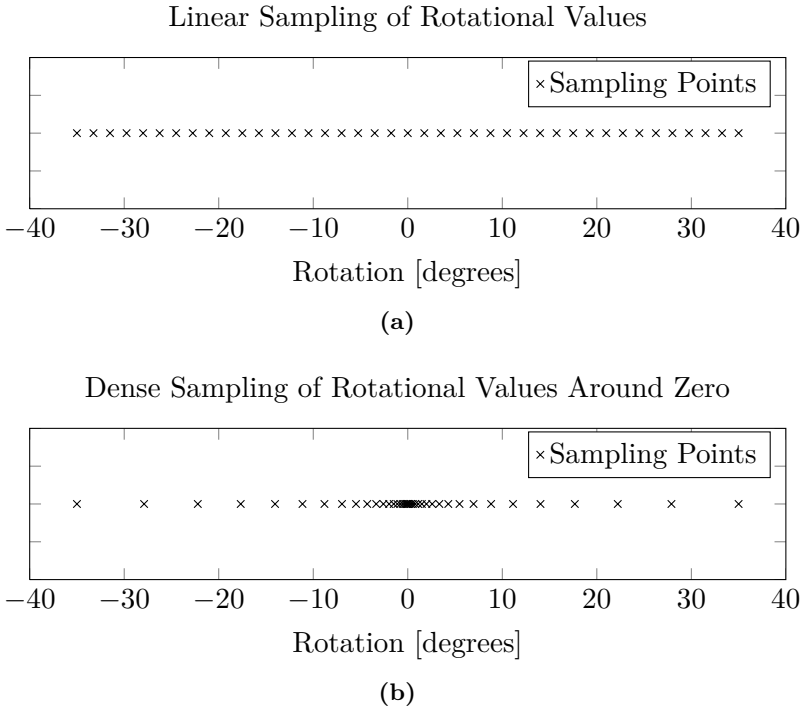


Figure 2.10: Sampling rotational values **(a)** without and **(b)** with feedback. The model without feedback samples linearly, the model with feedback samples densely around zero. For visibility reasons, both figures contain only 41 sampling points. The implementation used 71 sampling points.

input : maximum rotation sample m, odd number of sampling points n
output : List of sampling points $S = \{s_1, \ldots, s_n\}$

$X \leftarrow 1, \ldots, (n-1)/2$;
// generate right-half list S_r
foreach *element* $x_i \in X$ **do** $s_i \leftarrow y(x_i)$;
// re-normalize all values in S_r
foreach *element* $s_i \in S_r$ **do** $s_i \leftarrow (s_i / \max(S_r)) * m$;
// Create final list S
for $i = 1, \ldots, (n-1)/2$ **do**
  $S[i] \leftarrow -s_{\frac{n-1}{2}+1-i}$;
  $S[n-i+1] \leftarrow s_i$;
end
$S[(n-1)/2+1] \leftarrow 0$;

Algorithm 1: Generation of rotational sampling points for the feedback driven model. The returned sampling points are symmetrical around zero degrees.

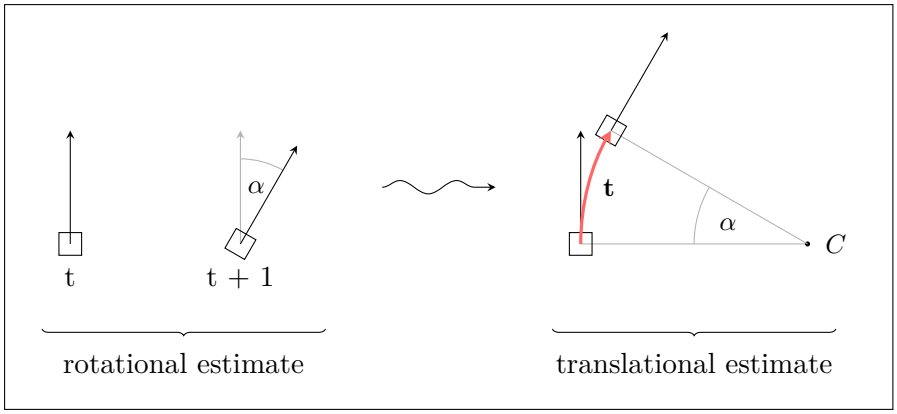
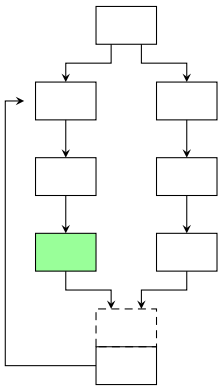


Figure 2.11: Calculation of the self-motion. The curve between two points can be calculated using the estimated rotation α and the agent's constant speed.

2.4.4 ESTIMATE GENERATION



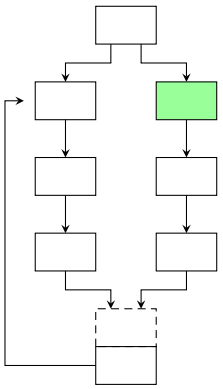
The self-motion estimate for the *template model* path is missing a further element. In addition to the rotational component already estimated, it needs a translational part. Note that this estimate is only used when analyzing the individual path. However, the calculation is the same for the model that employs both paths. In this case, the rotational estimate is extracted from the head direction network that is described further below. The translation can be computed using the rotational estimate and the constraint that an agent, having a constant speed, always moves tangential to a curvilinear path. For instance, after the agent estimated its rotation, the traversed path can be calculated by a circle. The traveled distance on the arc of that circle must reflect the agent's speed whereas the radius of the circle depends on the estimated rotation. Figure 2.11 provides an overview of this example.

In order to calculate the arc length between two positions p_t and p_{t+1} at times t and $t+1$, the speed s of the agent must be known. According to the model constraints, $s = \frac{1}{\text{frame rate}}[m/frame]$. Let β be the estimate for the yaw rotation, then the radius r of the curve is defined as $r = \frac{s}{\beta}$. The angle α between the vectors pointing from the two agent positions to the curve center C (see Figure 2.11) is equal to β. Finally, the translation $\mathbf{t}$ can be defined as

$$\mathbf{t} = r \begin{bmatrix} \cos(\alpha) \\ \sin(\alpha) \end{bmatrix} \quad . \tag{2.8}$$

The final result of the *template model* path is a vector $\mathbf{e} = (\beta, \mathbf{t})$ which contains the estimates for the rotation and translation.

2.5 CENSURE



In order to keep the model in a manageable size, I avoided to use landmarks that require behavioral subsystems. Researching and implementing such a subsystem would go beyond the scope of this thesis. As a replacement, feature detecting and tracking which can be applied on a per-frame basis was used. The following section will describe my implementation in detail.

In [AKB08], Agrawal, Konolige and Blas propose an enhancement to the well known feature detector SURF (Speeded Up Robust Features, [BETVG08]) called CenSurE – Center Surround Extremas. In addition to the improved detector, they introduce a new descriptor for features named MU-SURF (Modified Upright - SURF). The new descriptor opposes some problems inherent to the standard upright descriptor when the input data contains certain frequencies.

SURF uses Haar-Wavelets to approximate the Gaussian function which is used during feature detection. Unlike SURF, CenSurE uses Difference-of-Boxes (DoB) or Difference-of-Octagons (DoO) to approximate a Difference of Gaussians for the detection process. The latter one is more robust to rotational transformations of the input image than DoB, but slightly slower. According to [AKB08], both detectors yield a higher repeatability with respect to detected features than SURF. Haar-Wavelets, DoO and DoG are illustrated in Figure 2.12. In addition to being better than SURF, its repeatability quality is higher than other comparable methods like SIFT ([Low99]) or FAST and its extension FASTER ([RPD10]). Despite the fact that FASTER is indeed faster in feature detection than other known methods at the time of this writing, the higher repeatability is preferable[7]. Due to those findings, and because CenSurE was shown to work with challenging datasets in [KAS07], CenSurE was selected as feature tracking algorithm.

Due to the fact that CenSurE and MU-SURF are not already available in exhaustive libraries like OpenCV[8], I implemented the algorithm from scratch. More details about the technical specifics of the implementation can be found

7 A speed improvement for CenSurE is described in [EMC09]. However, this improvement was not applied to the implementation.

8 `http://opencv.willowgarage.com` (visited on 2012-03-19)

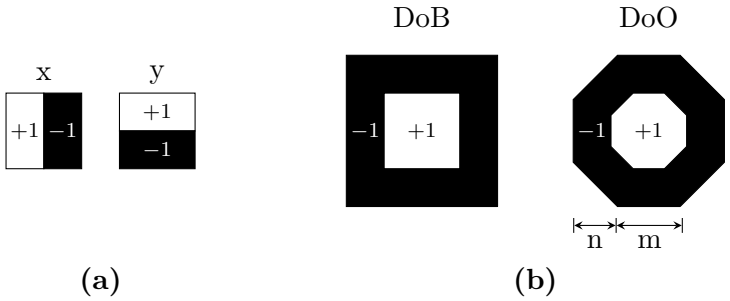


Figure 2.12: (a) Haar Wavelets for x and y direction. **(b)** Difference of Boxes (DoB) and Difference of Octagons (DoO). (cf. [AKB08]). The filters in both (a) and (b) collect the pixel intensities around a central pixel. In (b), the pixel intensities for the outer and inner octagon are weighted: Let O be the outer intensities and I the inner intensities, let a be the area of the inner octagon and b the area of the outer octagon. The filter response is thus defined as $r = \frac{1}{|a|}I - \frac{1}{|b|}O$. This ensures that no energy is induced by the filtering operation. The sizes (m, n) depend on the scale of the filter. (m, n) for the inner and outer octagon are provided in in Table 2.2.

in the Appendix, Section C.1; for all computations presented in chapter 3, the DoO was used.

2.5.1 INTEGRAL IMAGES

The implementation uses integral images – also known as summed area tables – to compute the DoB and DoO. For DoB, the normal (or "regular") integral images are used. The regular integral image $\Upsilon_{I,n}$ for input image I of width w and height h is defined as

$$\Upsilon_{I,n} = \sum_{i=0}^{i \leq w} \sum_{j=0}^{j \leq h} I(i,j) \quad . \tag{2.9}$$

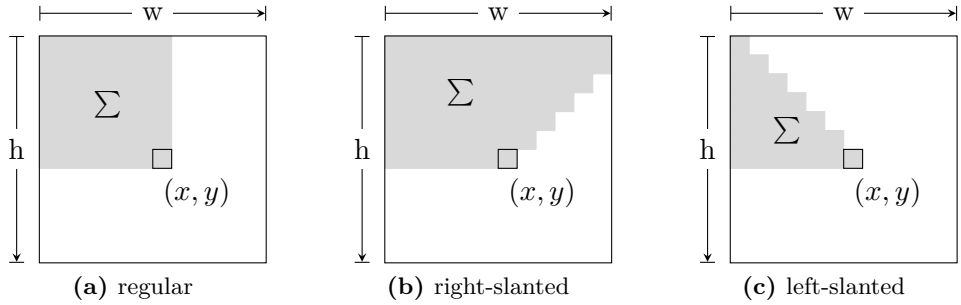
For the computation of the DoO, right-slanted integral image $\Upsilon_{I,r}$ and the left-slanted integral image $\Upsilon_{I,l}$ need to be calculated. They can be recursively defined as

$$\Upsilon_{I,r}(x,y) = \Upsilon_{I,r}(x+1, y-1) + \sum_{i=0}^{i \leq x} I(i,y) \tag{2.10}$$

$$\Upsilon_{I,l}(x,y) = \Upsilon_{I,r}(x-1, y-1) + \sum_{i=0}^{i \leq x} I(i,y) \quad , \tag{2.11}$$

where $\Upsilon_{I,r}(0,0) = I(0,0)$ and $\Upsilon_{I,l}(0,0) = I(0,0)$. Figure 2.13 shows examples of each of the integral images. To reduce the number of memory loads required to compute the integral images, they can be computed all at once in a combined function.

Using integral images ensures that the calculation of DoB or DoO are realizable in constant time. *Proof:* Given integral images $\Upsilon_{I,l}$, $\Upsilon_{I,r}$ and $\Upsilon_{I,n}$. To



(a) regular **(b)** right-slanted **(c)** left-slanted

Figure 2.13: Integral images. The gray areas are summed up. The regular integral image integrates over a rectangular area up until pixel (x, y) whereas the right-slanted integral image adds all pixels that are right-next to the pixel in all previous lines. The left-slanted integral image adds one additional pixel in each next row.

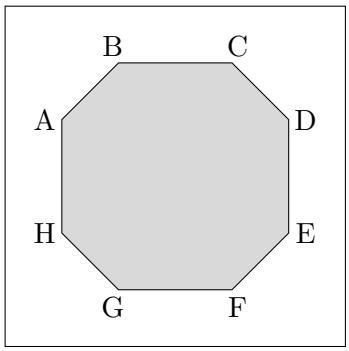


Figure 2.14: Integration of the octagonal area $A, \ldots, H$ requires only constant time when using slanted integral images.

compute the integral of the octagon shown in Figure 2.14, first split the octagon into an upper trapezoid U of coordinates A, B, C, D, a lower trapezoid L of coordinates H, E, F, G and the rectangle R surrounded by A, D, E, H. Using equations 2.9 to 2.11, it follows that

$$U = \Upsilon_{I,l}(D) - \Upsilon_{I,l}(C) - \Upsilon_{I,r}(A) + \Upsilon_{I,r}(B) \qquad \in O(1) \qquad (2.12)$$

$$L = \Upsilon_{I,r}(F) - \Upsilon_{I,r}(E) - \Upsilon_{I,l}(G) + \Upsilon_{I,l}(H) \qquad \in O(1) \qquad (2.13)$$

$$R = \Upsilon_{I,n}(E) - \Upsilon_{I,n}(D) - \Upsilon_{I,n}(H) + \Upsilon_{I,n}(A) \qquad \in O(1) \quad . \qquad (2.14)$$

Therefore, $U + L + R \in O(1) + O(1) + O(1) = O(1)$. This calculation must be done for the outer and the inner octagon, but this is still $\in O(1)$. □

2.5.2 FILTER RESPONSES AND DESCRIPTORS

To detect features in an image I, the previously described integral images are computed for the gray scale variant I_g of I. DoOs are subsequently computed not only for one but 7 different scales for each pixel. The exact dimensions of the outer and inner octagons are given in Table 2.2 for each scale. The result is a *scale space* S of dimensionality $w \times h \times 7$.

Table 2.2: Inner and outer sizes for the Difference of Octagons. (cf. [AKB08])

scale	1	2	3	4	5	6	7
inner (m, n)	$(3, 0)$	$(3, 1)$	$(3, 2)$	$(5, 2)$	$(5, 3)$	$(5, 4)$	$(5, 5)$
outer (m, n)	$(5, 2)$	$(5, 3)$	$(7, 3)$	$(9, 4)$	$(9, 7)$	$(13, 7)$	$(15, 10)$

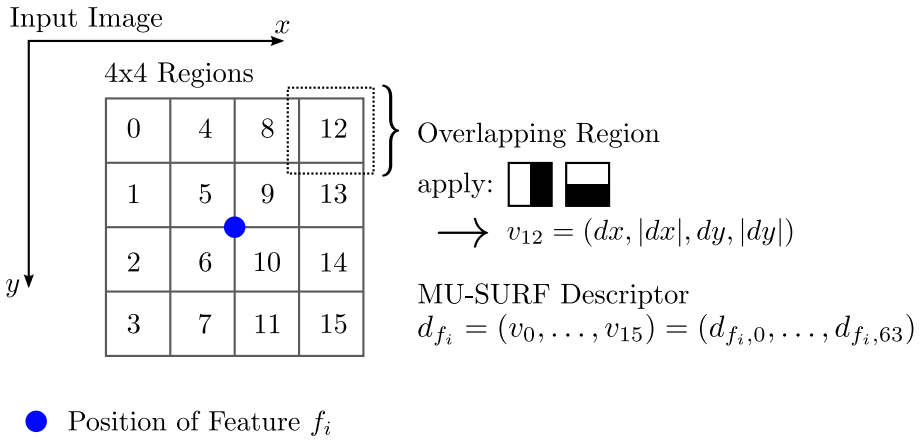


Figure 2.15: MU-SURF generation from a feature position f_i. To each of the 16 subregions, Haar-Wavelets in the x and y direction are applied to form subregion-description vectors v_i. They are finally concatenated to create d_{f_i}.

In the next step, each of the entries $s \in S$ is checked if it conforms to certain criteria:

1. $s > 30$

2. s must be a local extremum in a $3 \times 3 \times 3$ neighborhood. This includes filter responses of other scales.

3. s must have a Harris Cornerness ([HS88]) > 10.

When a value s fulfills all criteria, it is marked as a feature f_i and a MU-SURF descriptor d_{f_i} is generated for f_i. The descriptor d_{f_i} is created by taking a square region R of a certain scale-dependent size, which itself is split into 4×4 subregions. For each of those subregions $R_0, \ldots, R_{15}$, Haar-Wavelet responses dx and dy are computed for the directions x and y, respectively. To form the final MU-SURF descriptor d_{f_i}, all subregion description vectors, which are of the form $v_i = (dx, dy, |dx|, |dy|)$, are concatenated to yield $d_{f_i} = (v_0, \ldots, v_{15}) = (d_{f_i,0}, \ldots, d_{f_i,63})$. The descriptors of all features of a frame are stored within a descriptor table. This procedure is depicted in Figure 2.15. The difference between the MU-SURF descriptor and the normal upright-SURF descriptor is that each subregion R_i is extended slightly so it overlaps with its neighboring subregions.

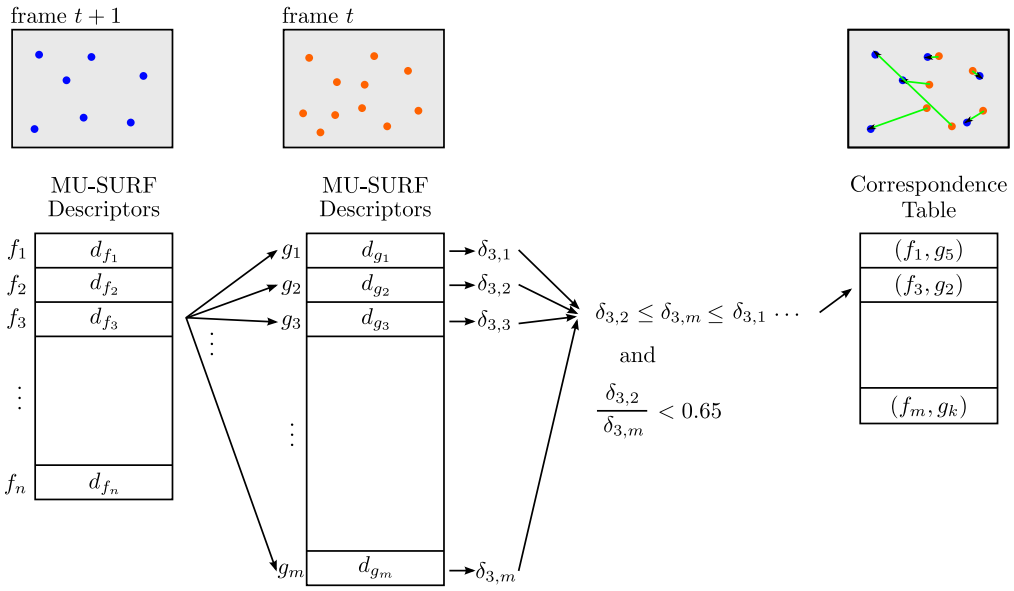


Figure 2.16: Feature matching between two frames. Shown is the matching of feature f_3 to all features found in the previous frame. At first distances $\delta_{l,k}$ are calculated. Next, the smallest two distances are compared to each other and if the distances suite the required criteria, a match is stored to the correspondence table. The value 0.65 was taken from the OpenCV implementation of SURF. Blue and orange dots are exemplary feature positions, green lines between such features are correspondence lines.

2.5.3 TRACKING

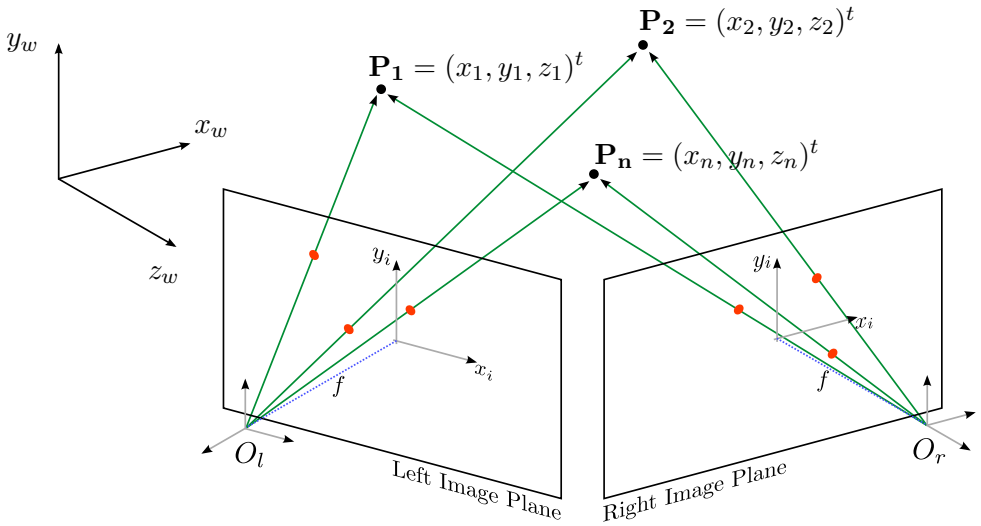
All features f_i and their corresponding descriptors d_{f_i} are stored in a descriptor table for the frame they were found in. To track features in two frames at times t and $t+1$, entries in both descriptor tables are compared one by one.

Let $f_i, i = 1, \ldots, m$ be the features of frame at time $t+1$ and $g_j, j = 1, \ldots, n$ the features of the frame at time t and d_{f_i} and d_{g_j} their corresponding descriptors. The distance $\delta_{l,k}$ between two features f_l and g_k is defined as the Euclidean distance of the MU-SURF descriptor components

$$\delta_{l,k} = \sqrt{\sum_{q=0}^{q=63} (d_{f_l,q} - d_{g_k,q})^2} \quad . \tag{2.15}$$

Distances between all features f_i and $g_j, i = 1, \ldots m, j = 1, \ldots, n$ are calculated and the two smallest distances $\delta_{r,s}$ and $\delta_{t,u}$ taken for further comparison. If, and only if those two distances are dissimilar enough, the two features with the smallest distance are marked as a corresponding match, or a so called *tracked feature*. To assist the mathematical description, Figure 2.16 contains an example outline of the just mentioned approach.

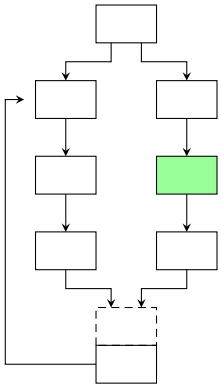
The CenSurE feature matching and tracking yields a list of matches for each pair of frames. The list contains features and their positions in each frame. Features of two consecutive frames can subsequently be used to estimate the ego-motion with the help of the epipolar geometry.



x_w, y_w, z_w: world coordinates
$\mathbf{P}$: point in 3-space
x_i, y_i: image plane coordinates
$O_{l,r}$: left, right camera origin
f: focal length
●: projected point on image plane

Figure 2.17: Overview of the epipolar geometry. The epipolar geometry describes the relationship of the camera coordinate systems which originate at O_l and O_r. Due to visibility reasons, the epipoles and epipolar plane were left out.

2.6 POSE ESTIMATION USING TRACKED FEATURES



The "intrinsic projective geometry" ([HZ00]) between two views of the same environment is called epipolar geometry. It describes the transformations that are required to turn projected points from one view into points of another view. Figure 2.17 helps to understand this kind of relationship. Points $P_1, \ldots, P_n$ in 3D space are projected perspectively onto the right and left image planes. With the help of the fundamental matrix $\mathbf{F}$, a 3×3 matrix of rank 2, it is possible to show a geometric relationship of point correspondences. The reverse way works as well. It is possible to estimate $\mathbf{F}$ from given point correspondences. Having assessed $\mathbf{F}$, the translation and rotation to transform one view to another can be extracted. To obtain the transformation, the essential matrix $\mathbf{E}$, which is computed from $\mathbf{F}$ with the help of the camera calibration matrix $\mathbf{K}$, is required. An extensive description of the epipolar geometry is given in [HZ00]. However, I will present the essential steps to compute $\mathbf{F}$ and $\mathbf{E}$ and how to acquire the rotation and translation from $\mathbf{E}$.

2.6.1 ASSESSMENT OF THE FUNDAMENTAL MATRIX $\mathbf{F}$

Longuet-Higgins described a method to establish the essential matrix $\mathbf{E}$ directly from point correspondences in [LH87]. This method only works for calibrated cameras, though. The fundamental matrix $\mathbf{F}$ on the other hand can be calcu-

lated for uncalibrated cameras. In order to keep the model open for variations in the camera model, **F** is estimated and subsequently **E** is calculated.

To compute **F**, an overdetermined system of equations has to be solved. The method of choice is the adaptive RANSAC (Random Sample Consensus), an extension to RANSAC presented in [HZ00]. The adaptive RANSAC is shown in Algorithm 2. Before using RANSAC, all input coordinates need to be normalized. The normalization will yield coordinates whose mean distance to the centroid of all features is $\sqrt{2}$.

input : data d, model fitting function f, distance function δ, number of required samples s, distance measure t
output : number of inliers n_b, estimated model M_b

$N \leftarrow 1$;
$p \leftarrow 0.99$;
$i \leftarrow 0$;
$m \leftarrow$ number of elements in d;
$maxIter \leftarrow 1000$;
// M_b will contain the best model so far
$M_b \leftarrow$ **nil**;
// n_c will contain the number of inliers of M_b
$n_b \leftarrow 0$;
while $N > i$ **do**
  $\sigma \leftarrow$ select s random samples from d;
  // the next line will create a candidate model M_c
  $M_c \leftarrow f(\sigma)$;
  // The distance function will yield the number of inliers
  $n_c \leftarrow \delta(M_c, d, t)$;
  if $n_c > n_b$ **then**
    $n_b \leftarrow n_c$;
    $M_b \leftarrow M_c$;
    // Calculate the quality of the new M_b
    $q \leftarrow 1 - \left(\frac{n_b}{m}\right)^s$;
    $N \leftarrow \frac{log(1-p)}{log(q)}$;
  end
  // prevent infinite loop
  $i \leftarrow i + 1$;
  if $i > maxIter$ **then** exit loop;
end
return M_b, n_b

Algorithm 2: The adaptive RANSAC algorithm.

RANSAC selects $s = 8$ feature correspondences from the set of all correspondences d in each iteration. Note that **F** is of rank 2 and only seven correspondences are thus required, but most implementations use eight. It thereupon calls the fitting function f to fit a model to the selected data. The fitting function is

Table 2.3: Parameters for the adaptive RANSAC algorithm. The algorithm estimates the fundamental matrix $\mathbf{F}$. s elements are required for this task. The algorithm is either iterated for at most 1000 times or when the quality of a candidate of $\mathbf{F}$ has a certain quality p. The measure to compute p is the Sampson distance. A distance threshold t is required to compute this distance. An outline of the algorithm is presented in Algorithm 2.

RANSAC Setup	
minimum number of samples s	8
maximum iterations	1000
quality measure p	0.99
distance threshold t	0.00001

implemented using the singular value decomposition (SVD) approach, which is described in [HZ00], to assess a candidate M_c for $\mathbf{F}$.

The next step is to measure the quality of M_c. This is achieved by calculating the Sampson distance between the (normalized) feature coordinates and the coordinates which were projected with the help of M_c. Let $\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^2$ be two point correspondences and $\mathbf{Y}_1$, $\mathbf{Y}_2 \in \mathbb{R}^3$ the correspondences with homogeneous coordinates. The Sampson distance Δ_S is defined as

$$\Delta_S = \frac{\mathbf{Y}_1^t M_c \mathbf{Y}_2}{f1 \cdot f1 + f2 \cdot f2} \quad , \tag{2.16}$$

where $f_1 = M_c\mathbf{y}_1$, $f_2 = M_c\mathbf{y}_2$ and $a \cdot b$ is the dot product of a and b. The Sampson distance is discussed in length in [HO06]. The distance threshold t is set to $t = 0.00001$, features for which $|\Delta_S| < t$ holds are identified as inliers. Remaining features are outliers. If the number of inliers is large enough, $\mathbf{F}$ is identified as a M_c and RANSAC is stopped, otherwise it will continue with the next iteration.

The fundamental matrix $\mathbf{F}$ which results from RANSAC needs to be re-normalized because the input data was previously normalized. For a detailed explanation of the normalization and renormalization step, I refer to [Har97]. The parameters which were used to produce the results of Chapter 3 are presented in Table 2.3.

2.6.2 EXTRACTION OF THE TRANSLATION AND ROTATION

The essential matrix $\mathbf{E}$ is required to compute the translational and rotational estimates. Given $\mathbf{F}$ and the camera calibration matrix $\mathbf{K}$, $\mathbf{E}$ is defined as

$$\mathbf{E} = \mathbf{K}^t\mathbf{F}\mathbf{K} \quad . \tag{2.17}$$

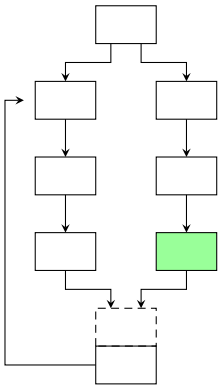
$\mathbf{K}$ is a 3×4 matrix containing the intrinsic calibration for a camera. For example, the calibration matrix for the default camera used in `blender` (until version 2.61) is defined as

$$\mathbf{K}_b = \begin{bmatrix} -a_u & 0 & u_0 & 0 \\ 0 & -a_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad , \tag{2.18}$$

where $a_u = f\frac{w}{2}$, $a_v = -f\frac{h}{2}$. w is the image width, h is the image height. $u_0 = \frac{w}{2}$ and $v_0 = \frac{h}{2}$ locate the image plane's principal point. f is the focal length, which must be computed from `blender`'s internal sensor size of 32 mm and the "lens focal length" l, which can be set by the user. The default value of l is 35 mm. Note that $\mathbf{K}_b$ already takes care of the fact that `blender`'s camera model is looking along the negative z axis. In contrast to this, literature describing epipolar geometry usually assumes the camera to look into the positive z direction. If this difference is not taken into account , unexpected sign flips may occur.

Given $\mathbf{E}$, it is possible to determine the translation and rotation. In [WT00], Wang and Tsui show that the SVD of $\mathbf{E}$ yields eight different solutions. The SVD approach was initially described in [Har92], but only four possible solutions were identified. Ling, et. al., who use the decomposition of $\mathbf{E}$ for camera motion tracking, support the findings of Wang and Tsui in [LCB11].

ESTIMATE SELECTION



The SVD of $\mathbf{E}$ yields candidate solutions for the translation and rotation. Decomposing $\mathbf{E}$ gives two different translation vectors $\mathbf{t}_1, \mathbf{t}_2 \in \mathbb{R}^3$, which are equal except for their sign. In addition, the decomposition yields two different rotation matrices $\mathbf{R}_1, \mathbf{R}_2 \in R^{3\times3}$. According to [WT00], $\mathbf{R}_3 = -\mathbf{R}_1$ and $\mathbf{R}_4 = -\mathbf{R}_2$ are possible solutions as well. Consequently, a set of eight possible solutions $S = \{(\mathbf{t}_i, \mathbf{R}_j), i = 1,2, j = 1,\ldots,4\}$ can be established. From the eight possible solutions, only one $(\mathbf{t}, \mathbf{R}) \in S$ is valid. It can be selected by calculating the 3D point from a feature correspondence. Solely one of the solution candidates in S will yield a 3D point that lies in front of both image planes.

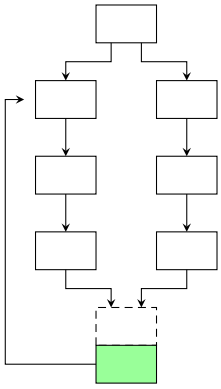
Another method to determine the correct solution is a simple heuristic that depends on the model constraints. The camera is moving along the camera's negative z axis, therefore the translational vector with a negative z component is the correct one. The correct rotation matrix is selected according to the following observation: Three of the four rotation matrices will contain rotations around at least one of the axis by 90 degrees. Hence, it is possible to identify the correct solution as the matrix containing the least overall rotation. The overall rotation is the sum of all rotations around each axis. It can be computed, for example, by converting a rotation matrix to Euler angles.

The implementation uses the constraint-dependent method to select the correct estimate. Ultimately, the selection is forwarded to the fusion module, where it is combined with the result of the *template model* path.

2.7 ESTIMATE FUSION AND THE HEAD DIRECTION NETWORK

Rotational estimates from to individual paths need to be fused and the result needs to be stored and must be retrievable. This shows that there is the need for a subsystem which holds an estimate and is able to alter its state according to new input. The following section contains a description of this subsystem and will continue with the fusion mechanism, which merges the two estimates from the *template model* and the *epipolar* path.

2.7.1 AN AMARI-TYPE HEAD DIRECTION NETWORK



Continuous Attractor Neural Networks (CANNs or CANs) are often used to simulate hippocampal head direction, place or grid cells (e.g. [BF09], [Rol07])[9]. They are networks of cells which are recurrently connected. Proper tuning of the recurrent weights leads to the appearance of a so called activity packet. The activity packet is a small area of the network which exposes neural activity even when external stimuli are absent [BT07]. Such an activity packet can be shifted by new stimuli, so that other areas of the network become active while the old position becomes inactive ([FWW09], [ZY10]). Hence, CANs are believed to play a major role in working memory. CANs were mathematically analyzed primarily by Shun-Ichi Amari ([Ama90], [Ama91], [WHA08], [WA05]). Schematics of a head direction network are given in Figures 1.1 and 2.18a.

In a nutshell, CANs achieve all requirements for the subsystem. They both hold certain states over time without input and change their states as soon as required. In fact, there exist different complex models of the head direction cell network, e.g. [XHS02], which utilize CANs.

MATHEMATICAL FORMULATION OF THE HEAD DIRECTION NETWORK

I defined a very simple head direction network which fulfills all requirements. The equations are based on RatSLAM's PoseCells and [BF09], a grid cell simulation for path integration which was shown to be very accurate for long distances. Each neuron is connected to nearby cells. Due to the fact that the neurons code for a circular input space, the neuron which codes for zero degrees has the neuron coding for 360 degrees as one of its direct neighbors.

Let $w_{i,j}$ be the weight from neuron j to i and θ a global inhibitory constant. Let y_j be the axonal potential of neuron j, the dendritic potential x_i of neuron i is hence defined as

$$\frac{dx_i(t)}{dt} = -x_i(t) + \sum_j w_{i,j} y_j(t) - \theta \quad . \tag{2.19}$$

9 There is a strong argument in the research community, how hippocampal networks in the rat and primate brain should be modeled. In addition to CANs, there are models describing dynamics of single cells with the help of e.g. interference, spin-glass models, etc. (for example see [ZYT+09], [FT06], [OB05], [MGG10]). They are not necessarily used to simulate head direction networks, though.

After the dendritic potential was calculated for a time t, all $x_i(t) < 0$ are set to zero. Afterward, the dendritic potentials are normalized according to

$$x_i(t) = \frac{x_i(t)}{\sum_j x_j(t)} \quad . \tag{2.20}$$

The axonal potential y_j is defined as

$$y_i(t) = tanh(x_i(t)) \quad . \tag{2.21}$$

In all simulations, θ was set to 0.002. I found this value to stabilize the activity packet in different simulations. Approximation of the differential equation was accomplished by transferring the continuous-time Equation 2.19 to discrete time. Let $\rho = \frac{dt}{\tau}$, the discrete form of Equation 2.19 is thus defined as

$$x_i(t+1) = (1-\rho)x_i(t) + \rho\left(\sum_j w_{i,j}y_j(t) - \theta\right) \quad , \tag{2.22}$$

where t is the discrete time step. Consequently, $y_i(t+1)$ must be computed from $x_i(t+1)$. The results presented in Chapter 3 were calculated with $dt = 1.0$ and $\tau = 1.0$.

Recurrent weights $w_{i,j}$ were defined using a Difference of Gaussian, yielding an on-center off-surround function. This recurrent connection will excite neurons which are near to an active neuron, but inhibit neurons which are further away. The inhibitory part is required to have a stable activity packet. If there were no inhibition of neurons in the far field, the activity packet would expand to encompass the whole neuronal field. Let g_σ be the Gaussian function, the weight $w_{i,j}$ from neuron j to i is defined as

$$w_{i,j} = g_{\sigma^1}(|i-j|) - g_{\sigma^2}(|i-j|) \quad , \tag{2.23}$$

with $\sigma^1 = 2.0$, $\sigma^2 = 5.0$.

Using Equations 2.20 to 2.23, the head direction network will expose a stable activity pattern at a neuron i after only 10 iterations. For this to happen, neuron i must be initialized to have a dendritic potential of one, while all other neurons $j \neq i$ must initially have a dendritic potential of zero.

To shift the activity packet, a center shifted variant of Equation 2.23 is used. The shift depends on the amount the packet should be shifted and the number of cells that are used in the head direction network. For example, the activity of a network which uses 360 cells should be shifted. The shift has to reflect a change of $r = -5$ degrees. The weights $w^s_{i,j}$, required to shift the activity packet, are thus defined as

$$w^s_{i,j} = g_{\sigma^1}(|i-j|+r) - g_{\sigma^2}(|i-j|+r) \quad . \tag{2.24}$$

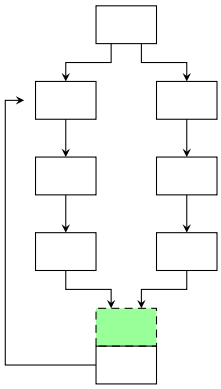
Note that r must be increased as soon as larger numbers of neurons are used to simulate the network and vice versa. To apply $w^s_{i,j}$ to the head direction network, Equation 2.22 is calculated using $w^s_{i,j}$ instead of $w_{i,j}$. The implementation will only work for a certain range of rotations. This is because the shifting function is not wide enough to cover high angular values. To counteract this problem, a

different shifting function needs to be defined. Due to the fact that large angular motions were not analyzed with the model, the shifting function presented in Equation 2.24 was used. The shifting procedure is illustrated in Figure 2.18b. All simulations employed 360 head direction cells.

STATE INTERPOLATION

The same problem which was described in 2.4.1 for the *template model*, appears in the head direction network. Instead of using the neuron that discharges maximally as head direction estimate, the neuronal field needs to be interpolated. The interpolation follows Equation 2.5, where the set $\Omega_{n_{\max}}$ contains the maximally responding neuron $n_{\max}$ and 7 of its neighbors in each direction. The w_n of Equation 2.5 are the neural activities of the head direction cells in $\Omega_{n_{\max}}$.

2.7.2 FUSING ESTIMATES IN TWO DIFFERENT WAYS



I investigated two different possibilities to fuse the rotational estimates from the two processing paths. The first method merges the estimates according to a weighted average, the second method combines two shifting functions.

To calculate the weighted average, it is necessary to define the weights. They are confidence measures about how certain each processing path is about its estimate. For example, the *epipolar* path will have a low confidence value for its estimate when there were many features in the new image, but only few were tracked in two frames. The *template model* path will have a high confidence only in the case when the optical flow estimation yields a high confidence. The instantaneous confidence measures are defined in the following way:

- The confidence of the *epipolar* path is defined as

$$c_e = \frac{t}{n} \quad , \tag{2.25}$$

  where t is the number of tracked features between the two most recent frames, n is the number of features found in the current frame.

- The MotionAlgo yields confidence measures for each flow vector of its output. Therefore, the confidence of the *template model* path can be defined as

$$c_t = \frac{\sum_{x=1}^{w} \sum_{y=1}^{h} c_M(x,y)}{wh} \quad , \tag{2.26}$$

  where $c_M(x,y)$ is the confidence of the flow vector at position (x,y), w is the width of MotionAlgo's output and h its height.

  If the MotionAlgo was not used to estimate optical flow, the confidence measure c_t is defined according to

$$c_t = \frac{n-z}{n} \quad , \tag{2.27}$$

  where n is the number of all flow vectors and z is the number of flow vectors v_i with $|v_i| = 0$. This is usually the case when a pixel has infinite

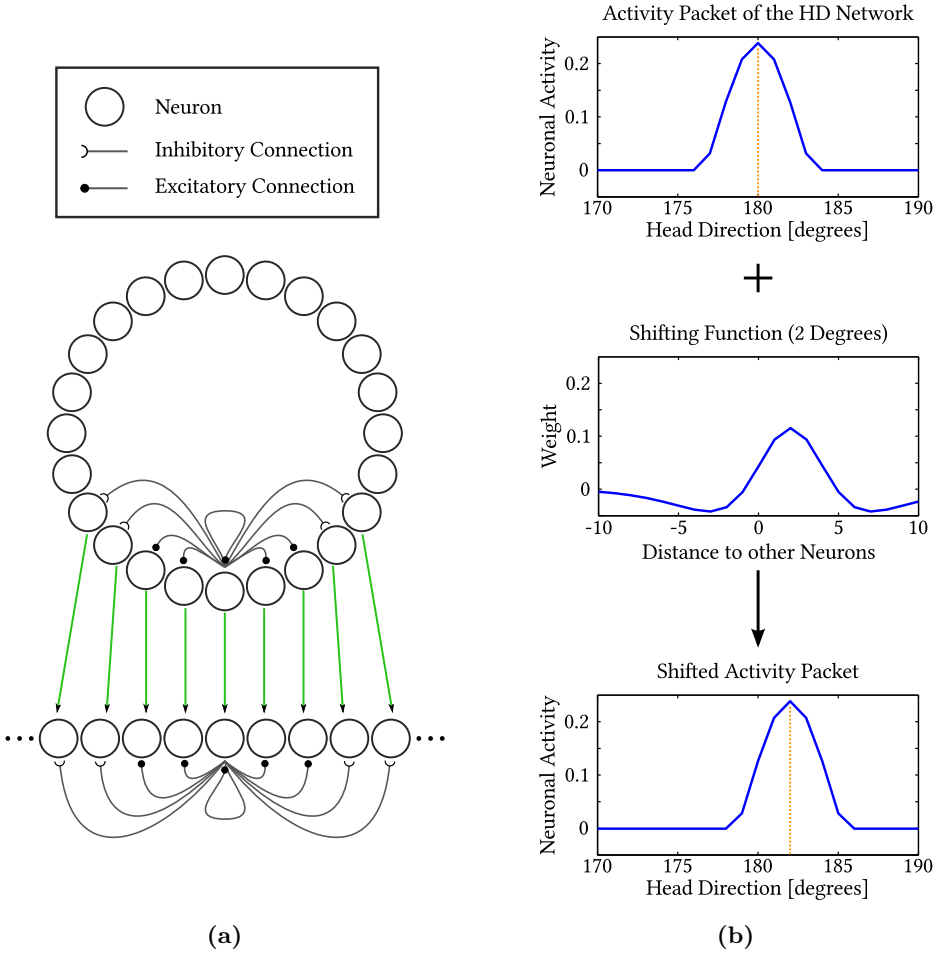


Figure 2.18: (a) Schematic of the head direction network. The circular head direction network can be illustrated as a linear network. Only exemplary connections of one neuron are shown in the Figure. **(b)** Activity packet shifting. The activity packet of the head direction network is shifted by two degrees to the right. The shifted packet is stable after one iteration of the network. Note that only the responses of a small number of cells around the activity packet are shown.

depth and thus, optical flow could not be computed. This happens, for example, in 3D models when the camera is looking at no structure but into "void".

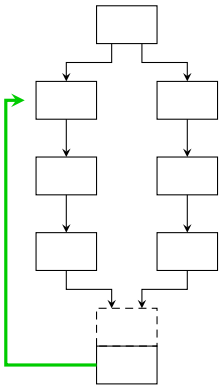
The instantaneous confidence measures of the last three frames are used to calculate the gliding average confidence measures c_e^g and c_t^g. Utilizing the confidence measures c_e^g and c_t^g, it is possible to define the two methods to merge estimates:

- WEIGHTED AVERAGE
  Let r_e and r_t be the rotational estimate of the *epipolar* and *template model* path in degrees. r_e and r_t are thus transformed to unit vectors according to a function Φ:
  $$\Phi(r) = \begin{pmatrix} sin(r) \\ cos(r) \end{pmatrix} \quad . \tag{2.28}$$
  The fused result r_f can thus be defined as
  $$r_f = \Phi^{-1}\left(c_e^g \Phi(r_e) + c_t^g \Phi(r_t)\right) \quad . \tag{2.29}$$
  A corresponding function w_f needs to be created according to Equation 2.24 to shift the activity packet of the head direction network. As soon as w_f is created, it needs to be applied to the network.

- SHIFTING FUNCTION
  Instead of a vector addition, the *shifting function* method creates functions w_e and w_t according to the rotational estimates r_e and r_t and Equation 2.24. Hence, w_f can be defined by
  $$w_f = c_e^g w_e + c_t^g w_t \quad . \tag{2.30}$$
  This function can be applied directly to the head direction network without any further calculation.

In summary of this section, the head direction network of the proposed model is a network of recurrently connected cells which expose an activity packet. The activity packet can either be shifted according to a precomputed function, or by an additive application of two different functions.

2.8 FEEDBACK



Using the head direction (HD) network and a prediction signal, it is possible to create a feedback for upstream modules. After a prediction signal was applied to the HD network, the activity packet will be shifted. The change in the HD network state can be expressed as an angular difference Δ between the old state, and the new state. With the help of the feedback Δ, it is possible to cancel out the rotational component. This is based on the fact that optical flow does not depend on the depth of the surrounding environment. Of course the approach does not guarantee the complete separation of the rotational component of optical flow. The remaining rotational component that is coded in the optical flow

will be very small when Δ is near the ground truth value, though, and can be analyzed with high precision. Sampling rotational angles with a high precision is described in Section 2.4.3.

The instantaneous optical flow is defined according to [LP80], with the difference that Longuet-Higgins and Prazdny used another camera setup. Whereas their camera is looking along the positive z axis, mine is looking along the negative axis. The outcome is a sign flip in the initial formulation, which yields two sign flips in the rotational component of optical flow.

Let X and Y be the coordinate of the optical flow on the retinal plane. Let ω_x, ω_y and ω_z be the rotations around the x, y and z axis, respectively. The optical flow for rotations around x, y and z is thus defined as

$$u = \frac{1}{f}\begin{pmatrix} XY\omega_x - (f^2 + X^2)\omega_y + fX\omega_z \\ (f^2 + Y^2)\omega_x - XY\omega_y - fX\omega_z \end{pmatrix} \quad , \tag{2.31}$$

where f is the focal length. Under the model constraints that the agent is only allowed to carry out rotations around the y axis, the equation can be simplified to

$$u = \frac{1}{f}\begin{pmatrix} (f^2 + X^2)\omega_y \\ XY\omega_y \end{pmatrix} \quad . \tag{2.32}$$

Let v be the optical flow vector at a position (X, Y), determined either using ground truth data or with the help of the MotionAlgo. Let u be the rotational flow component according to Equation 2.32, $\omega_y = \Delta$, at (X, Y). Hence $\nu = v - u$ is the optical flow that needs to be processed in the *template model* path.

Although feedback could have been incorporated to the *epipolar* path as well, the focus was set on the *template model* path. Including feedback to the *epipolar* path would enable the rejection of outliers in advance to estimating the fundamental matrix. The rejection would have to be based on the assumption that the feedback signal is sufficiently precise. For each correspondence, which was found with the help of CenSurE, a measure could be defined which determines how well it reflects the rotational feedback. If the measure falls below a certain threshold, the correspondence could be removed before having an influence on the fundamental matrix estimation.

A feedback like the one described in this section was previously not described in the literature – neither for one of the paths independently, nor in combination with a fusion system. Thus, feedback was analyzed separately for the *template model* and for the system which uses fused estimates.

2.9 ERROR METRICS

In order to quantify errors in estimations of translation and rotation, a list of different error metrics were used. An estimation error occurs, for example, when the agent rotates 5.0 degrees to the right, but the algorithm computing the estimate yields a value of 4.5 degrees.

2.9.1 REFERENCE SYSTEMS

Defining the reference systems is necessary before introducing the error metrics. To compute the positional difference of two points, it is either possible to compute the error with respect to a global reference point, or by taking one of the two positions as reference. This is also true for rotations.

The two different reference systems are referred to by *local* or *global* reference system, the errors are thus called *local error* or *global error*. Rotational, translational, and positional error metrics can be calculated using either the local or the global reference system.

LOCAL ERRORS

Local errors describe the instantaneous errors that occur between two estimates. An agent which rotates between two frames by 5.0 degrees but estimates its rotation at 4.5 will have an instantaneous error of 0.5 degrees.

GLOBAL ERRORS

To understand accumulative effects of estimation errors, it is necessary to compute global errors. As an example, consider an agent moving on some path. The agent has a constant local error, which causes an overestimation to the right for the translational estimate by 0.05 m. Over time and although the local error stays constant between different consecutive positions, the global error will increase steadily.

2.9.2 ANGULAR ERRORS

Angular errors for rotation estimates were calculated according to the quantification of rotational bias described in [TTH96]. Let $\mathbf{R}$ be the ground truth rotational matrix and $\bar{\mathbf{R}}_\mathbf{i}$ be an estimated rotation matrix[10]. The "difference rotation" [TTH96] matrix is defined as $\Delta\mathbf{R} = \mathbf{R}^t\bar{\mathbf{R}}_\mathbf{i}$, where ^{t} is the transpose operator. The angle θ_i, which takes $\bar{\mathbf{R}}_\mathbf{i}$ to $\mathbf{R}$, is defined as

$$\theta_i = \cos^{-1}\left(\frac{tr(\Delta\mathbf{R}) - 1}{2}\right) \quad , \tag{2.33}$$

where $tr(\mathbf{B})$ is the trace of a matrix $\mathbf{B}$.

To compute the average angular error $\tilde{\mathbf{R}}$ of the set of estimation matrices $Q = \{\bar{\mathbf{R}}_\mathbf{1}, \ldots, \bar{\mathbf{R}}_\mathbf{n}\}$, $\Delta\tilde{\mathbf{R}}$ must be computed for $\tilde{\mathbf{R}}$ instead of an $\bar{\mathbf{R}}_\mathbf{i}$. Note that extracting Euler angles or quaternions from rotation matrices gives solutions which are not necessarily unique. Therefore it is preferable to compute the average rotation matrix directly from all elements of Q. 3D Rotation matrices are elements of Lie Group SO(3) [Sel96], on which the addition theorem is not defined. To convert a matrix $\mathbf{B} \in SO(3)$ to a corresponding form $\boldsymbol{b} \in so(3)$, the matrix logarithm can be used [Moa02]. so(3) is the three dimensional Lie Algebra, on which the addition theorem for matrices is defined. The resulting

10 If a processing path yields only a rotational value around the y axis, a corresponding rotation matrix needs to be computed first.

matrix $\mathbf{b}$ is skew-symmetric and contains angular velocities, which can be averaged element-wise. The inverse function of the matrix logarithm, which gives $\mathbf{B} \in SO(3)$ for a $\mathbf{b} \in so(3)$, is the matrix exponential. The matrix logarithm and matrix exponential will be henceforth denoted as ${}^{M}\log$ and ${}^{M}\exp$. Constructing the average rotation matrix $\tilde{\mathbf{R}}$ for all $\bar{\mathbf{R}}_\mathbf{i} \in Q$ is thus defined as

$$\tilde{\mathbf{R}} = {}^{M}\exp\left(\frac{1}{n}\sum_{i=1}^{n} {}^{M}\log\left(\bar{\mathbf{R}}_\mathbf{i}\right)\right) \quad . \tag{2.34}$$

Using equation 2.33, the standard deviation $\sigma_{\tilde{R}}$ of the mean angular error $\tilde{\mathbf{R}}$ is defined as

$$\sigma_{\tilde{R}} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\theta_i^2} \quad . \tag{2.35}$$

2.9.3 TRANSLATION ERRORS

Errors for the translation estimation were calculated in form of the angular difference between the ground truth translational vector (or directional vector) and the estimated direction. This is due to two basic reasons: 1) the model constraints require the agent to travel at a constant speed and 2) the two processing paths will produce estimates that need to be scaled. For example, the epipolar geometry yields unit vectors for the directional vector due to a renormalization step. Using the constrained speed the translation vector can be scaled to the correct length.

Directional vectors are two-dimensional because the model is restricted to allow movement only in the xz plane. Let $\mathbf{t} = (t_x, t_z)^t$ be the estimated translation vector and $\boldsymbol{\text{ø}} = (\tau_x, \tau_z)^t$ the ground truth vector. Hence, the error ξ between $\mathbf{t}$ and $\boldsymbol{\text{ø}}$ is defined as

$$\xi = \cos^{-1}\left(\mathbf{t} \cdot \boldsymbol{\text{ø}}\right) \quad , \tag{2.36}$$

where $a \cdot b$ is the dot product of two vectors a and b. The calculation could as well be written

$$\xi = \sqrt{\left(\tan^{-1}\left(t_z, t_x\right) - \tan^{-1}\left(\tau_z, \tau_x\right)\right)^2} \tag{2.37}$$

but then the cyclic space of the input data, which must lie in $[-\pi, \pi]$, needs to be enforced. Although most computer languages provide a function for $\tan^{-1}$, usually named `atan` (or with quadrant checking `atan2`), the implementations and accepted input spaces might differ. Thus, documentations for `atan` and `atan2` need to be read carefully, and input data might need a transformation.

2.9.4 POSITION ERRORS

The error with respect to the agent position is called position error and is the Euclidean distance between the ground truth and the estimated position. An agent located at $(4.5, 3.0)$ m with respect to a global reference point, believing that it is at $(4.6, 2.8)$ m, has a position error of approximately $0.22\,\text{m}$.

Let $\mathbf{p} = (p_x, p_z)$ be the position estimate and $\mathbf{æ} = (\rho_x, \rho_z)$ the ground truth position. The error ζ_t between $\mathbf{p}$ and $\mathbf{æ}$ at time t is defined as

$$\zeta = \sqrt{(\rho_x - p_x)^2 + (\rho_z - p_z)^2} \quad . \tag{2.38}$$

The mean position error κ_t at time t of an agent traveling on a path is

$$\kappa_t = \frac{1}{n} \sum_{i=1}^{n} \zeta_i \quad , \tag{2.39}$$

where n is the number of estimates collected until t.

3 RESULTS

Each of the processing paths *epipolar* and *template model* were thoroughly analyzed separately to understand the estimation errors that are introduced. An exhaustive examination of the results of the *template model* path was especially needed, as it guided the selection of the different interpolation and subsampling strategies. The distinct strategies are described in Sections 2.4.1 and 2.4.2. Next, the two fusion techniques which I defined in Section 2.7.2 were studied. Finally, the system that employs feedback was considered. For this system, results are shown for the fused estimates together with the individual data of the *template model* path.

Therefore, this chapter can be regarded as a step by step analysis of the different building blocks of the model that I suggested in chapter 2. It leads to the best parameter and function set for the submodules and their composition to solve the problem of self-motion estimation from visual input.

SETUP PARAMETERS Parameters for the template model and the MotionAlgo were fixed for each experiment, if not otherwise specified. Exact values for these two submodules can be found in Tables 2.1 and A.1. Additionally, settings for the adaptive RANSAC were kept constant for all trials. They are listed in Table 2.3. Optical flow was stored and processed according to the Middleburry flow format, which has $(-1, -1)$ in the top left corner of the image plane and $(1, 1)$ in the bottom right corner as long as the image plane is quadratic. If the ratio of width and height was not $1 : 1$, the top left corner was set to $(-1, -r)$ and the lower right to $(1, r)$, where $r = \frac{h}{w}$, h denotes the image height and w the image width.

3.1 ESTIMATION ERRORS DUE TO NON-EXACT FEATURE POSITIONS

Errors in the estimation of the spatial position of an agent can have diverse reasons. For instance, the limited precision in numerical representation of numbers can cause errors. This is usually imposed by the data type and programming language that has been chosen for an implementation. Another source of errors is the application of numerically critical arithmetic operations such as division

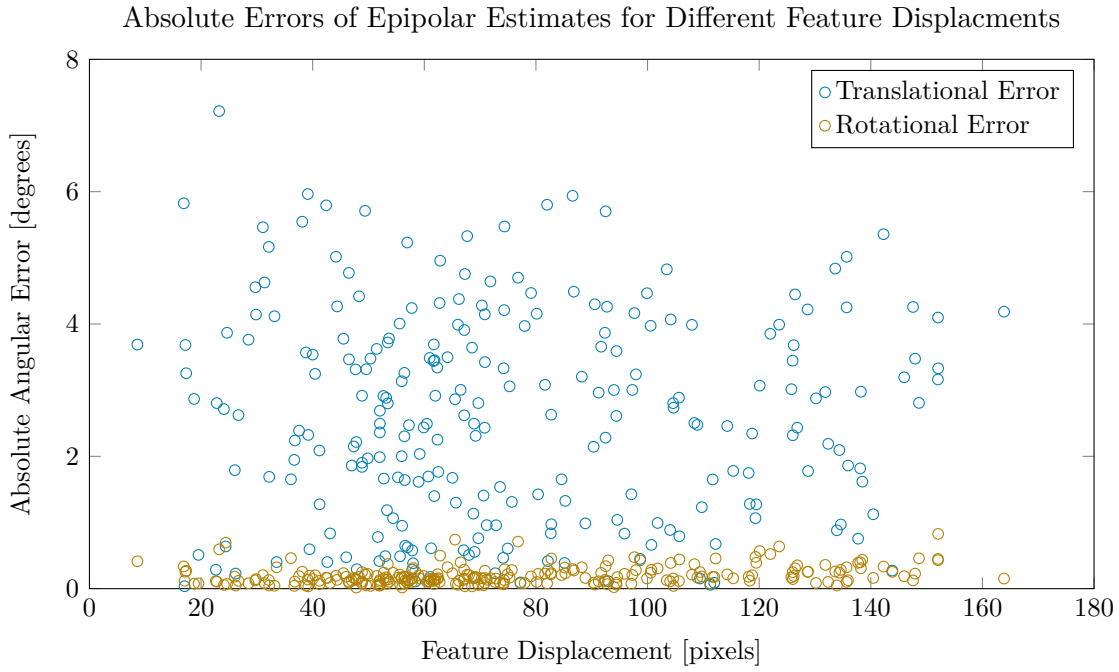
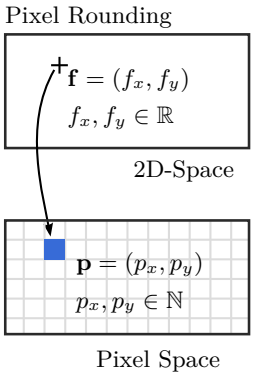


Figure 3.1: Rotational and translational estimation errors for different feature displacements. The input data consists of randomly scattered points in 3D space. They were perspectively projected onto the image plane; exact positions were rounded to pixel coordinates. Different displacements between features originate from different depths in 3D space and different translational and rotational values for the camera. Translation, rotation, and feature point positions in 3D space were randomly selected for 250 trials.



by a small number. The errors that occur from the various sources can be expressed by using the error metrics which I defined in Section 2.9.

In my implementation of the *epipolar* path, the inaccuracy of feature positions introduce the most errors. The CenSurE application, for example, yields feature positions in the form of pixel coordinates of the input image. A higher precision, e.g. sub-pixel accuracy, is not available for virtual data at the moment. The error that is imposed by pixel coordinates was quantified in the following way. Analytical feature positions were rounded to pixel coordinates in order to simulate the output of CenSurE, the image plane width and height were set to 480×360. The agent's translation $\mathbf{t}$ and rotation φ between two frames were selected according to the independent uniform distributions $\mathbf{t} = (t_1, t_2)^t$, $t_i \sim U(-10, 10)$ m and $\varphi \sim U(-10, 10)°$. Consequently, different feature displacements are created this way. The displacement d of a feature f between two frames at times t and $t + 1$ is defined as the Euclidean distance of that feature between the frames. Errors in the estimation of the rotation and translation were subsequently calculated. Figure 3.1 shows the results for 250 trials. The errors do not show a dependence on the displacement d, although a slight increase of the error in the rotational estimate could be assumed for larger displacements.

I collected another set of data to further investigate the rotational error. The results were generated by an agent which travels on a circular path of radius $r = 7.5\,\text{m}$ through a point cloud for 40 seconds (s). The point cloud is described in Section 2.2.1. The agent traverses a distance of $1\,\text{m/s}$, data was collected for the different frame rates $\{1, 2, \ldots, 20\}$. This ends in smaller rotations and translations between two frames for higher frames. Analytical feature positions were rounded to pixel coordinates again. The simulation was carried out 100 times. Each time, the point cloud was randomly generated. The results are depicted in Figure 3.2(a). The absolute angular errors stay approximately constant for different frame rates. This supports the previous findings which were shown in Figure 3.1.

A demonstration of the impact of the error in percent is drawn in Figure 3.3. The Figure shows the ground truth trajectory as a gray curve, the estimated trajectory of an agent using 1 FPS is drawn as a red curve while the estimated trajectory of an agent using 10 FPS has a blue color. The agent using the high frame rate displays a tendency to overestimate its rotation. This results in an estimated trajectory that is narrower than the ground truth trajectory. Note that Figure 3.3 needs to be interpreted carefully. Due to the higher frame rate, the green trajectory is made of 401 data points, whereas the red trajectory contains only 41 data points.

3.2 EVALUATION OF THE TEMPLATE MODEL

The template model has several spots which introduce errors. For example, the choice of a subsampling strategy influences the result. Subsampling is required to reduce the number of flow vectors which are used as input. I described different subsampling strategies in Section 2.4.2. Although there are a variety of descriptions and explanations of results of template models available in the literature (e.g. [PS98]), detailed information about the influence of certain steps during the computation is either not given at all or badly explained.

Parameters for all experiments are listed in Table 2.1. Data was collected from an agent moving on a circular path with radius $r = 7.5\,\text{m}$. The agent traveled for $40\,\text{s}$ with a speed of $1\,\text{m/s}$.

3.2.1 INTERPOLATION STRATEGY FOR ESTIMATE SELECTION

The estimate of rotation is determined by the interpolation strategy. For instance, instead of using the rotation for which the maximally responding neuron codes, a certain area surrounding this neuron is integrated. The different methods which I considered are described in Section 2.4.1. The input data were 10×10 analytical optical flow vectors. Figure 3.4 shows the results for each method and different frame rates. In addition, the estimate of the maximally responding neuron is plotted. Quite interesting are the good results of the *gauss full* method for a frame rate of 4 FPS. The method interpolates the template

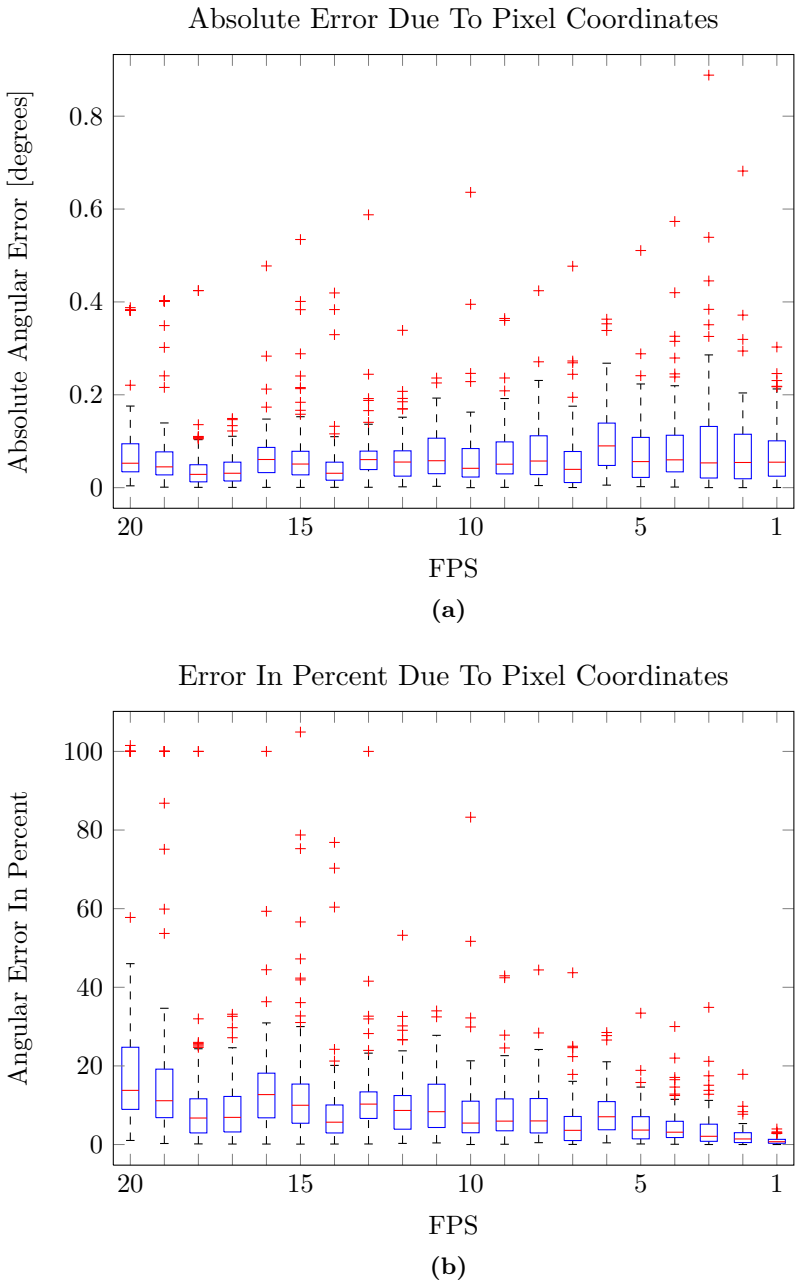


Figure 3.2: Detailed analysis of the rotational error. **(a)** Absolute error, introduced by rounding feature positions to pixel coordinates. **(b)** Error in percent, imposed by rounding exact feature positions to pixel coordinates. Both datasets were collected by using random point clouds as input data. Points were perspectively projected onto the image plane and tracked between frames. The datasets consist each of 100 simulations for each frame rate. The agent traveled for 40 s with a speed of 1 m/s on a circular path with radius $r = 7.5\,\mathrm{m}$ through the point cloud.

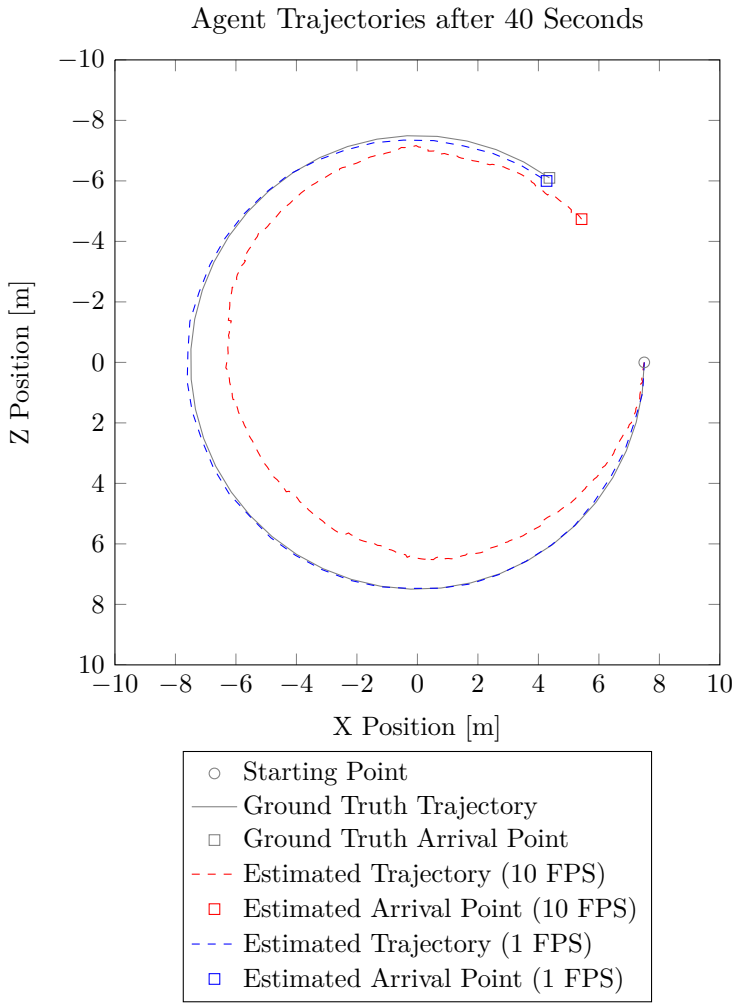


Figure 3.3: Estimated trajectories for two different frame rates and ground truth trajectory.

Figure 3.4: Response field interpolation results. The mean values for different frame rates are shown as boxes. The boxes are connected in order to improve visibility.

model's entire response field. Nevertheless, the estimate is worse than the results of the other interpolation strategies for higher frame rates.

The results of methods *simple* and *DoG* are interesting for a frame rate of 5 FPS. Both methods yield a very small error for this frame rate. The error is larger for most of the other frame rates which were considered. This is especially true for the *DoG* interpolation method. Its error in the estimation of the rotation is worse for all other frame rates.

Interpolation with the *gauss near* method is best for high frame rates. In fact, Figure 3.4 shows that *gauss near* is the only interpolation strategy that yields increasingly better results for higher frame rates. Due to the fact that the targeted virtual scene provides 10 FPS, *gauss near* was employed as the interpolation strategy for all simulations in the following sections.

3.2.2 INPUT SIZE

The number of optical flow vectors affects the estimation quality. Typically one would assume that a higher number of vectors effects less error in the estimation. Although Perrone showed in [Per92] that the template model achieves a higher accuracy for estimates of the translation for a higher number of vectors, the subject is not studied well enough. Therefore I examined the rotational estimates when using different numbers of input flow vectors. Input data was analytical optical flow in a variety of input sizes. The results, which are illustrated in Figure 3.5a, reveal that the estimation errors stay approximately constant for

each input size. However, they contradict the assumption that a higher number of flow vectors must always be beneficial. Especially outstanding are the results for 30×30 and 40×40 flow vectors, as they yield the best estimates with respect to the mean rotational error for a global reference system. This is despite the fact that the number of vectors is quite small. Interesting in addition to these findings are the results for an input size of 10×10. The good results for this size were quite unexpected.

Figure 3.5b shows the mean angular errors and standard deviations for the data shown in Figure 3.5a. The standard deviation declines with the use of more flow vectors. 30×30 flow vectors yield the best result with respect to the mean angular error. The input sizes 30×30 and 40×40 are not significantly different to each other according to Welch's t-test with a significance threshold of 5%. 30×30 flow vectors will thus be used as the input for subsequent simulations. The reduced number of flow vectors leads to a drastically smaller consumption of both time and memory for the computation of the template model response.

3.2.3 SUBSAMPLING STRATEGY

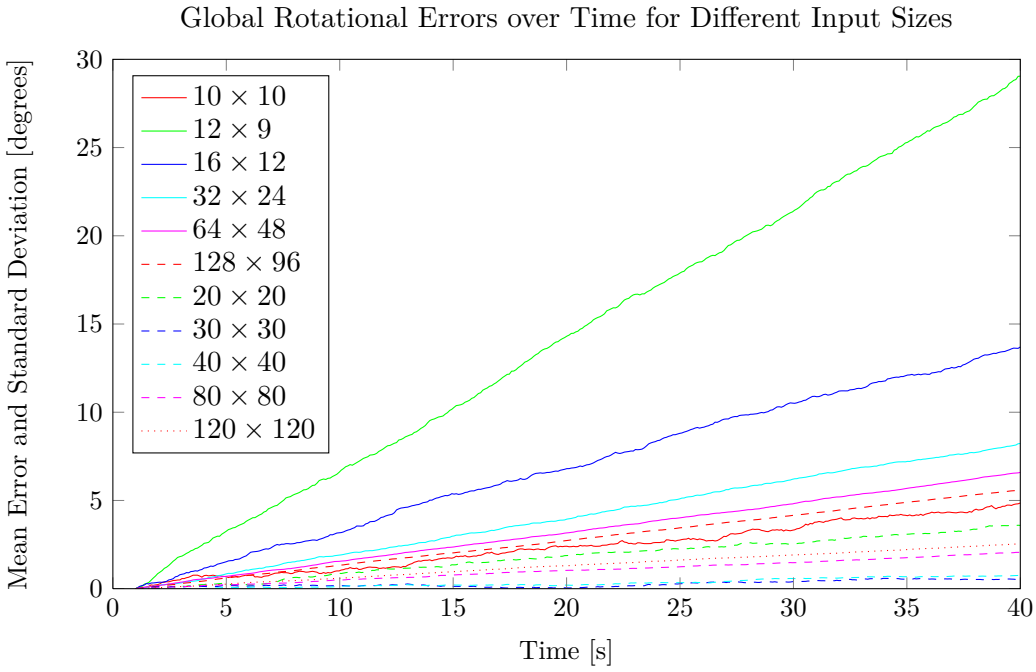
Reducing the number of flow vectors is an important step. The previous section pointed out that a larger number of flow vectors not necessarily leads to better estimations. In addition, a large number of optical flow vectors may reach computational limits. For instance, the ground truth optical flow of the virtual sequence is of size 480×360. Matlab appallingly consumes memory and time to compute the template model's response. Hence, I proposed different methods to subsample flow vectors in Section 2.4.2.

Note that the term *subsampling* is not fully correct. The process is an interpolation of flow vectors, but the term *subsampling* is used anyway to distinguish this step from the interpolation of the response field.

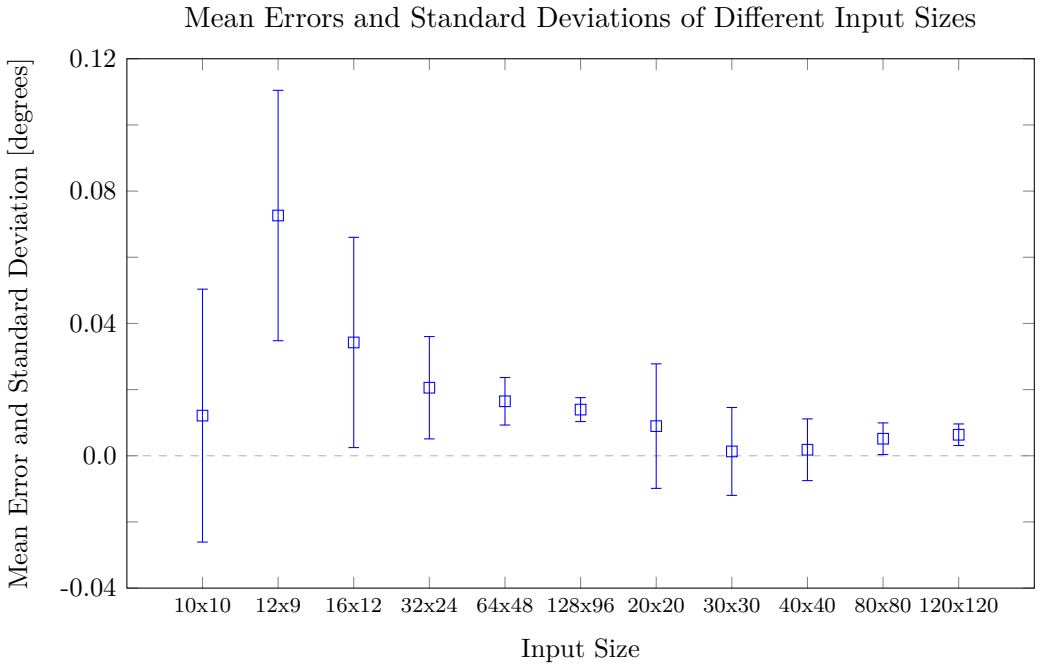
Figure 3.6 shows the results for the different subsampling strategies. Outstanding is that the *mean* subsampling strategy is always better than the median flow vector for all shown input sizes. Again, 30×30 flow vectors resulted in the estimate with the least mean errors. The best subsampling strategy is almost always the mean vector without overlap. Due to the fact that the mean vector of overlapping areas is best for 30×30 flow vectors this method was used throughout the following simulations. In addition, the difference is insignificant according to Figure 3.6, Note that 30×30 was the default size of flow vectors for most simulations because it had the least mean angular error when compared to other numbers of input sizes. This is shown in Section 3.2.2.

3.3 EVALUATION OF DIFFERENT FUSION METHODS

Fusing the results of both processing paths is required to update the head direction network. The activity packet of the network needs to be shifted in such a way to reflect the combined information of the two paths. Fusion can be accomplished in two different ways, described in Section 2.7. The first method is



(a)



(b)

Figure 3.5: Analysis of the rotational errors. Displayed are errors of the rotational estimates of the template model. The input was analytical flow of different input sizes. **(a)** Rotational errors over time with respect to a global reference frame. The displayed curves suggest that the under- or overestimation of the rotation stays approximately constant over time, as the curves exhibit an almost fixed slope. Magnitudal differences in the error lead to the different inclination of the curves. The curves themselves are a result of the accumulation of errors over time. **(b)** Mean angular errors and standard deviations.

.

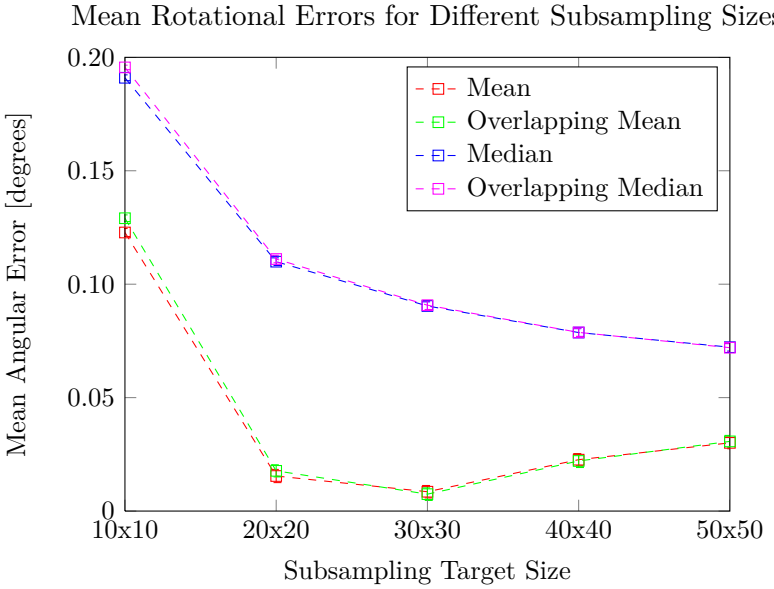


Figure 3.6: Mean rotational estimation with respect to a local reference frame for different input sizes and subsampling strategies. Standard deviations were left out to improve visibility. In addition, all mean values are connected by dashed lines.

to calculate the averaged mean. The second method introduces a change of the recurrent weights of the neurons in the network. The virtual sequence was used to examine both procedures. Features were detected and tracked with the help of CenSurE. The fusion was tested for both ground truth optical flow generated with the help of `exrflow` and optical flow estimated with the MotionAlgo. As a consequence of Sections 3.2.2 and 3.2.3, the flow was subsampled to a size of 30×30.

The results indicate no significant difference of the methods. Figure 3.7 shows the mean angular error and their standard deviations. The two methods are named *mean average* and *head direction* throughout this section. In addition to the fused results, the distinct estimates of the two processing paths are included. Note that the *epipolar* path is independent from the optical flow that was used. Therefore, the Figure is split into three logical groups: 1) contains only the estimate of the *epipolar* path; 2) contains the results of the *template model* path and the fused results for estimated optical flow; 3) contains the results for ground truth optical flow.

Supplementary to this finding, there is virtually no difference discernible for the two types of optical flow in Figure 3.7. Table 3.1 shows the exact numerical values to broaden the insight into the Figure. The logical grouping of the Table follows the Figure. Obviously, the *template model* path works slightly better for ground truth flow data. In contrast to this, mean angular errors of fused results for ground truth flow are worse than the errors for estimated optical flow. Nevertheless, the differences are negligible.

As a consequence of the findings, the simulations of the next section will use the *head direction* method. Estimated optical flow yields only marginally worse

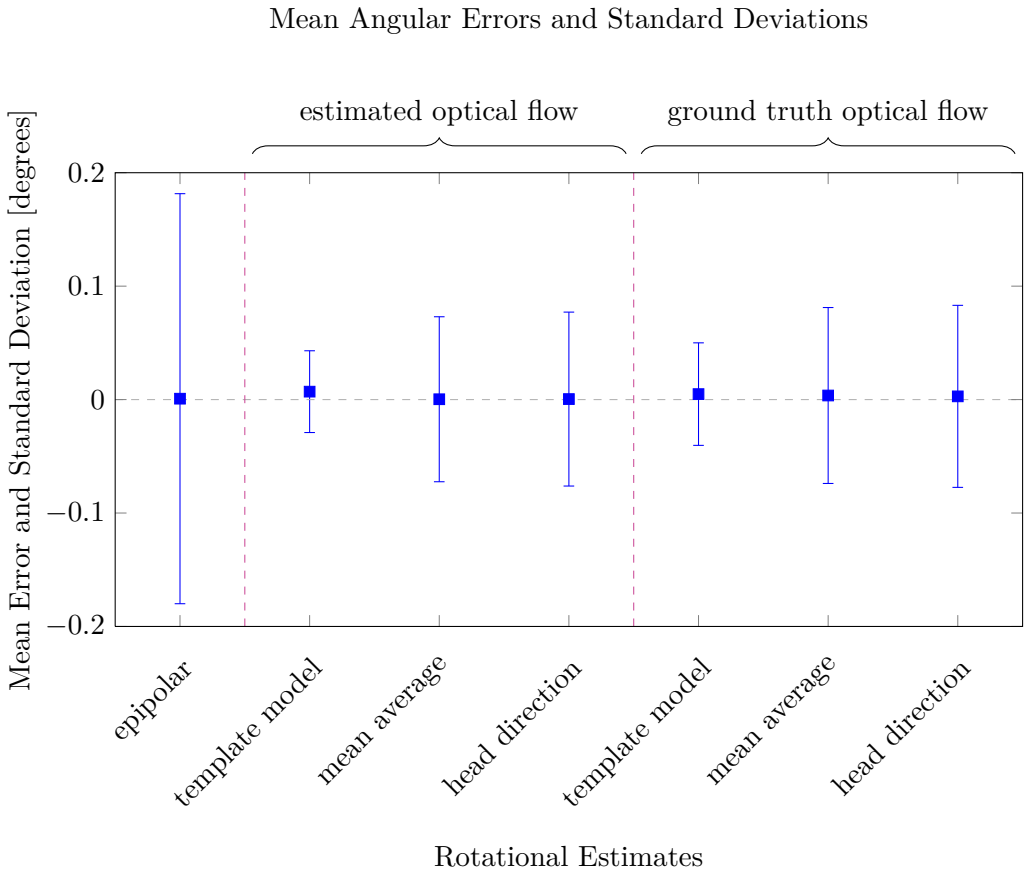


Figure 3.7: Mean angular errors for rotational estimates and their standard deviation. Estimates from the *epipolar* path were independent from the optical flow. Two different sets of optical flow were used to compute the results of the template model. Both estimates were fused either according to an averaged mean or by changing the head direction network's recurrent weights.

Table 3.1: Numerical values for the data displayed in Figure 3.7. The grouping is the same as in the Figure. The first group contains solely the epipolar estimate. It is independent from the used optical flow. The second group contains the different results for estimated optical flow. The third group presents the data for ground truth flow.

epipolar	template model	mean average	head direction	template model	mean average	head direction
Mean Angular Error [°]						
0.0008	0.0070	0.0003	0.0005	0.0048	0.0036	0.0028
Standard Deviation [°]						
0.1807	0.0360	0.0727	0.0767	0.0452	0.0775	0.0802

results for this method than ground truth optical flow. The implementation is slightly less verbose for this method, though.

3.4 ESTIMATE IMPROVEMENT DUE TO FEEDBACK

Feedback requires the template model to have a different sampling of rotational angles. If there is no feedback, the model samples linearly in the range of $[-35, 35]$ degrees. By contrast, the sampling of rotational angles is dense around zero degrees for the model that employs feedback. Linear and dense sampling are described in Section 2.4.3. Consequently, the model analyzed in this section uses dense sampling when feedback is applied.

The prediction signal is a random value drawn from the normal distribution. The mean of the normal distribution is set to the ground truth rotation ϕ. In order to analyze the systems reaction to different amounts of noise, the standard deviation was set variably. Hence, the prediction signal p can be defined as

$$p = \phi + \frac{\phi}{\nu} r \quad , \tag{3.1}$$

where r is a random value drawn from the standard normal distribution and ν is the amount of noise in percent. Note that the model that does not incorporate feedback is not influenced by this noise in any way. The model which uses feedback is called the feedback-model, the model that ignores feedback the feedforward-model. If not otherwise specified, both models fuse the estimates of the two processing paths.

The input to the models was optical flow estimated with the MotionAlgo. This is due to the findings in Section 3.3. At first, results were analyzed for 30×30 flow vectors. Then, the number of flow vectors was increased.

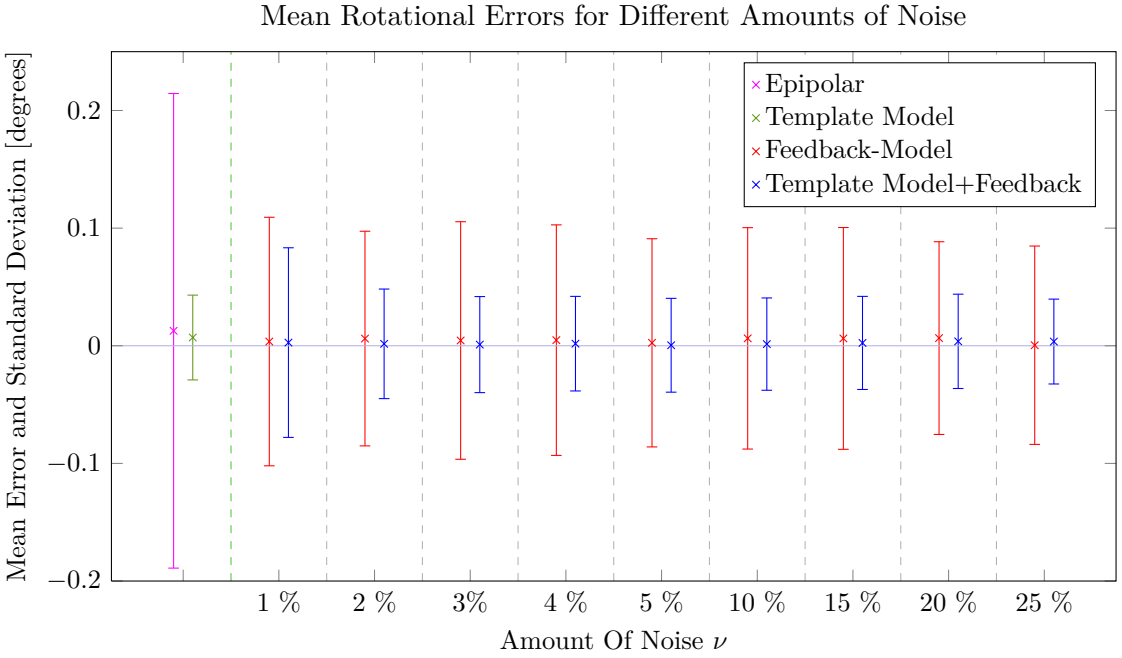


Figure 3.8: Results of the fused model and the *template model* path without fusion when using feedback. The results for the fused model without feedback were almost identical to the results using feedback, so they are not shown here. The first two error bars show the mean rotational errors and standard deviations for the *epipolar* and *template model* paths for comparison reasons. As they do not incorporate a feedback signal, they are not influenced by any feedback signal noise.

In order to make comparisons more comprehensible, different results are shown in all Figures. For instance, Figure 3.8 contains results for the feedback-model together with results for the individual paths of the feedforward-model. In addition, the results of the feedback-model for the individual *template model* path is shown.

The model does not exhibit a bias towards noise. In fact, the results for different amounts of noise are almost identical. For example, there are no significant differences of the errors of the rotational estimates shown in Figure 3.8 according to Welch's t-test with a significance level of 5 %. The Figure shows slight tendencies, though. For instance, the mean angular error for the feedback-model of the individual *template model* path is nearer to zero than the feedforward-model. On the other hand, the standard deviation of the error seems to be slightly increased. Note that the findings of Section 3.3 still hold for the feedback-model.

Next, I increased the number of flow vectors that were used as input. The number of 30×30 flow vectors is biologically not nearly plausible. Thus, it is necessary to study the results for higher amounts of flow vectors. As a consequence, the simulation was repeated for the different sizes of 30×30, 40×40, $\ldots$, 70×70 and finally 96×72 flow vectors. The last input size originates from

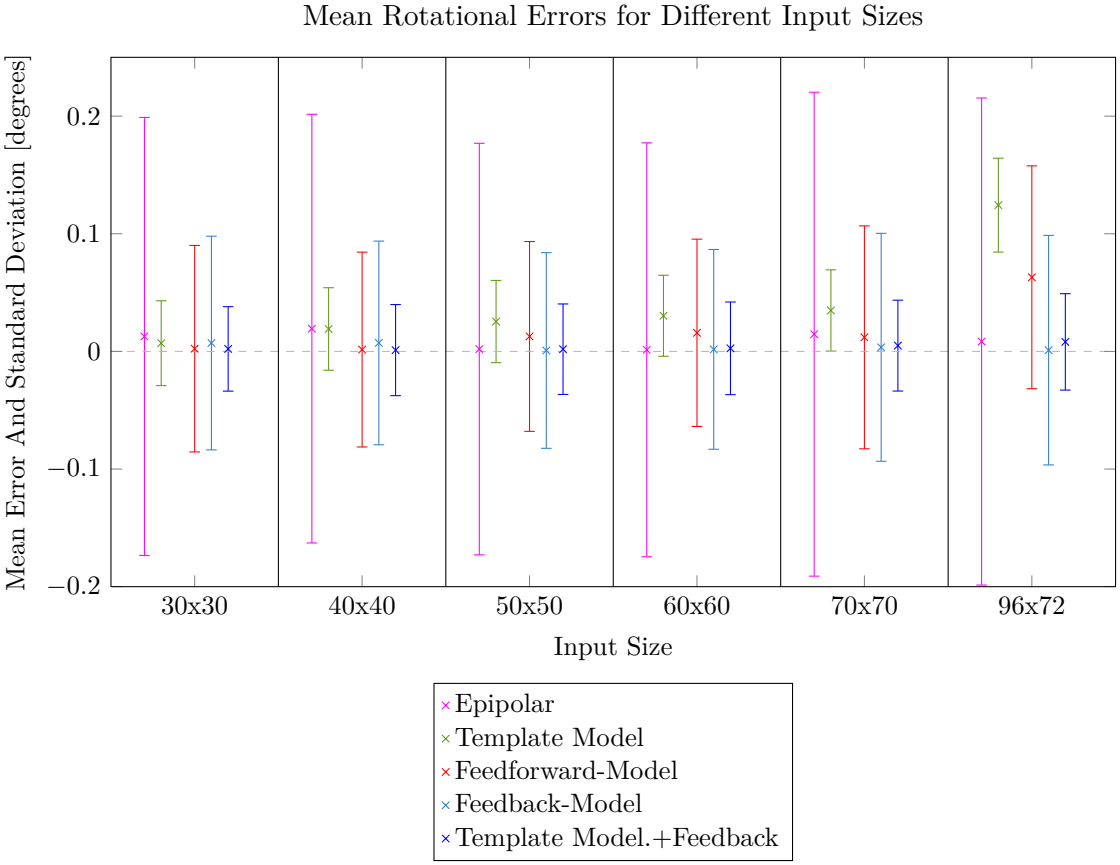


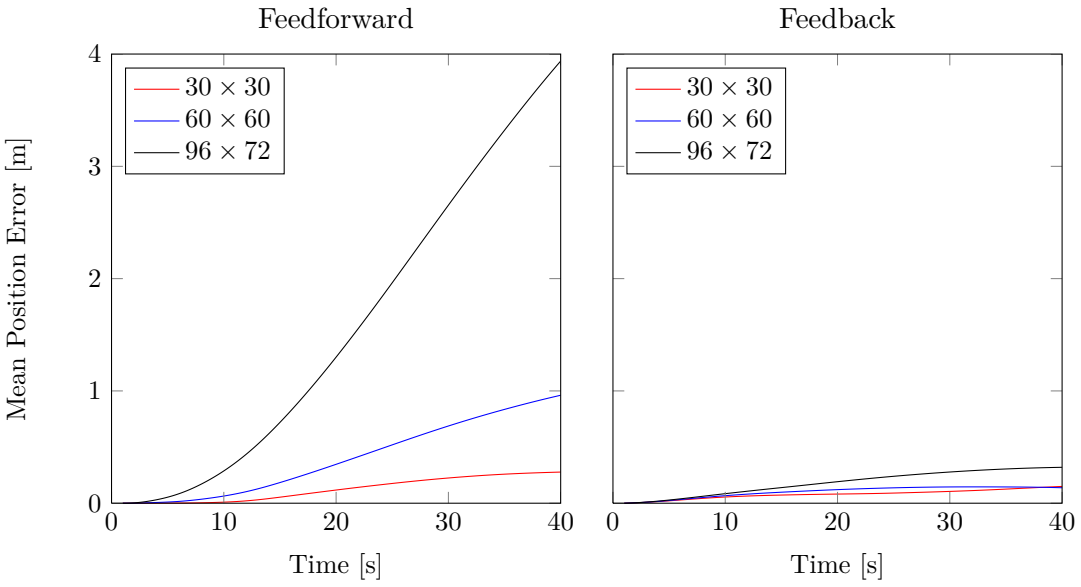
Figure 3.9: Results of the models for different input sizes with and without feedback. The first two bars in each group are the *epipolar* and the *template model* estimate without any feedback. The *template model* path with feedback is abbreviated as "Template M.+Feedback" due to the lack of space. It is significantly different from the *template model* path without feedback for $\geq 40 \times 40$ input flow vectors.

MotionAlgo's output size for model area MT. MotionAlgo subsamples the original resolution of the virtual sequence by a factor of five. Due to the fact that the virtual sequence has a resolution of 480×360 pixels, the corresponding MT size is 96×72. Subsampling of flow vectors was turned off in the feedback model when 96×72 were used. It was enabled for all other sizes. For all simulations, the noise level was set to 20%.

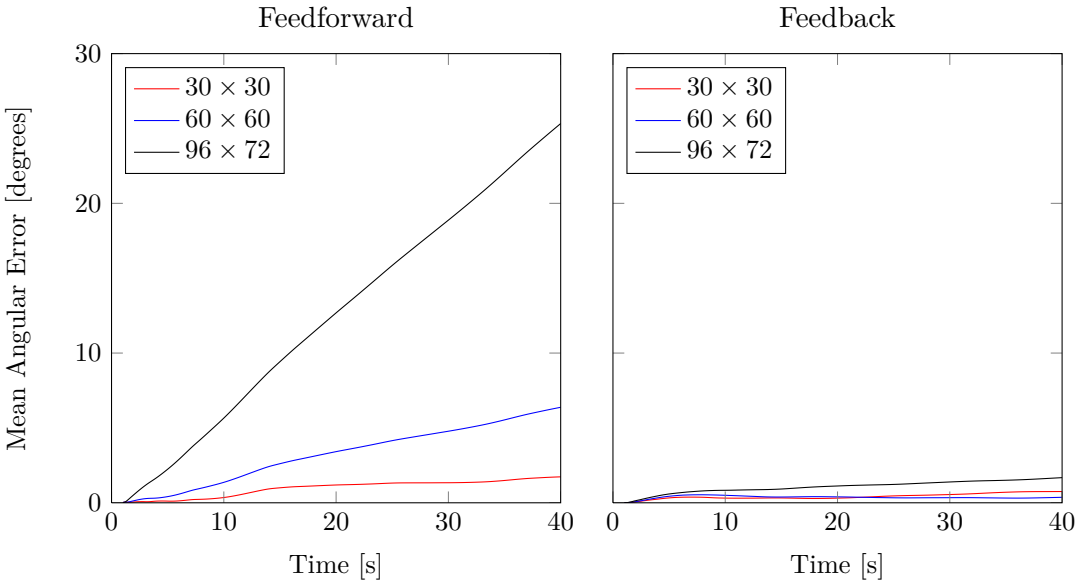
The feedback-model benefits from the feedback. This result is not statistically significant, the data shows a tendency, though. Figure 3.9 illustrates the findings. The Figure shows the data for the different input sizes. For each input size, the rotational estimates of the individual processing paths, the feedforward-model, the feedback-model and the feedback-model for the individual *template model* path are shown. For 30×30 input flow vectors, the findings of Figure 3.8 are confirmed. The mean angular errors are close to zero. The next group of Figure 3.9 shows the outcome for 40×40 flow vectors. Both individual paths

without feedback show a slight increase of the mean angular error. The standard deviations stay approximately the same when compared to the 30×30 results. The decline in the estimation quality of the *template model* path increases with the use of 50×50 flow vectors. Mean angular errors stay near to zero for the other displayed estimation errors. In the groups for 60×60 and 70×70 flow vectors, the error in the estimation of the *template model* keeps getting worse. Note that the mean angular error for the feedback-model continues to stay near to zero. In the last group of Figure 3.9, the estimates for 96×72 flow vectors are illustrated. The *template model* estimate is the worst of all groups. Apart from the difference between the individual *template model* path and the feedback-model for this individual path, the most interesting finding is that the mean angular error of the feedback-model is still approximately zero. Thus, there is a tendency visible in the data. The feedback-model seems to benefit from the feedback when larger numbers of flow vectors are used. This finding is not supported by Welch's t-test, though.

In contrast to these finding, a statistically significant difference exists between the feedforward-model and the feedback-model of the individual *template model* path. A significant difference according to Welch's t-test with a significance threshold of 5 % can already be shown for the comparison of the results for 30×30 and 40×40 flow vectors. 96×72 flow vectors show the strongest implication. Figure 3.10 amplifies this finding. The mean errors over time of the translational estimate are shown in Figure 3.10a. Below, the mean errors over time of the estimation of the rotation are shown in Figure 3.10b. An overwhelming improvement of the estimate of the agent's spatial location and rotation is shown for 96×72 flow vectors. Hence, the feedback model helps to improve the self-localization task.



(a)



(b)

Figure 3.10: Mean errors over time of the feedforward and the feedback-model for three different input sizes. The y axes are scaled to the same size to improve the comparability. **(a)** Mean errors with respect to a local reference frame of the position estimate. **(b)** Mean errors with respect to a global reference frame of the rotational estimate.

4 DISCUSSION AND CONCLUSION

The results of Chapter 3 need further explanation. For example, the findings for the different interpolation strategies of the *template model* response field need clarification. There seems to be a coincidence of the results of some of the methods. Hence, I will debate the findings and give reasons for their occurrence in order to limit the scope of interpretation. In addition, I will discuss different parts of the model with respect to their justification or comparison to other models.

4.1 DISCUSSION

The estimation of the self-motion with the help of the epipolar geometry yields approximately constant errors. Figures 3.1 and 3.2a do not show a dependence of the error on the feature displacement. This is due to the fact that the error which is introduced by rounding exact positions to pixel coordinates is limited by the size of a pixel. Hence, the error is not arbitrarily large but confined to the extents of one pixel. A direct consequence are the errors in percent.

Pixel coordinates impose estimation errors onto high frame rates. A high frame rate will result in small amounts of rotation and translation between frames. Precisely this problem is depicted in Figure 3.3. The agent which uses a high frame rate acquires many data points, but the transformation from one spatial position to the next is small when compared to the agent that uses a low frame rate. Paying attention is thus needed when interpreting trajectory curves with different numbers of data points per curve.

As already noted, the results of the interpolation methods need clarification. I mentioned that the *DoG* and *gauss full* methods expose certain frame rates for which they work best in Section 3.2.1. Additionally illustrated in Figure 3.4 is the estimate of the maximally responding neuron. Comparing the results of the single neuron to, for example, the *gauss full* method for a frame rate of 4 FPS shows that their errors coincide. A possible reason is that the rotation between two frames at 4 FPS is best represented by an interpolation of the complete response field. Other rotations, which are reflected by other frame rates, do not coincide with this. In addition to this, different rotational values may possibly fit best to the parameters of the examined methods. For instance the *DoG* method scores well for 5 FPS because the Difference of Gaussian reflects the

available response field for this method. As the *gauss near* method does not expose such a frame rate, I assumed that it is not susceptible to certain states of the response field.

The performance of the *mean* subsampling method is interesting. The efficiency seems to contradict the findings in [SRB10]. The authors found the median filtering to be the best operation available to remove noise and outliers in optical flow. However, it did not turn out to be the best in my simulations. The results of the simulations are depicted in Figure 3.6. The reason for the good outcome of the *mean* method is due to the already smooth input data, the median vector does not necessarily reflect the vectors in the cell that is subsampled. The internals of the MotionAlgo already produce smoothed data, there are almost no sharp edges of optical flow with the exception for outliers. Hence, the best representation of a certain patch of the flow vectors is possibly the mean vector. In addition, it can be assumed that the optical flow which is generated with `exrflow` from the virtual scene forms a smooth manifold. Thus, the same justification applies.

Although Perrone and Stone found that more flow vectors lead to a higher accuracy, I found contradictory results. According to Figure 3.5b, even small numbers of input flow vectors yield a mean rotational error that is near to zero. On the other hand, I could affirm that the standard deviation of the errors declines for larger numbers of flow vectors. Possibly, the *template model* either over- or underestimates the rotation to the approximately same extent. This would lead to the described large standard deviations, but would ultimately lead to a small mean angular error. In addition, the agent could estimate its spatial position correctly due to a neutralizing effect. For instance, two consecutive estimates with the exact opposing over- and underestimations might lead to the correct overall estimate.

The *epipolar* and *template model* path seem to under- and overestimate diametrical to each other. The results of the simulation for the different fusion methods suggest this notion. For example, the mean angular error for the fused results is smaller than the mean angular error for the individual *epipolar* path. The difference is vanishingly small, though.

Noisy prediction signals do not impose errors on the rotational estimates. In fact, this is true for the feedforward-model as well as the feedback-model for the individual *template model* path. I assume that dense sampling of rotational angles around zero has the main effect. The amount of noise is not large enough to let optical flow remain that contains high rotational values. Thus, the dense sampling can precisely examine the remaining part. This conclusion is supported by the fact that the mean angular errors for both feedback-models are smaller than for the feedforward-models. Thus, the system is not vulnerable to noise in the inspected quantities.

Although the data is not always statistically significant, a tendency that feedback is profitable can be spotted. Removing the rotational component of optical flow due to feedback improves the estimate. This is especially the case for the feedback-model for the *template model* path, the impressive consequence can be regarded in Figure 3.10. The findings for the other models are influenced due to the large standard deviations of the estimation errors. For example, the standard deviation of the *epipolar* path shown in Figure 3.9 is huge when compared

to the other models. Thus, there is no statistically significant difference discernible between this individual path and the other models with the exception of the *template model* path. Consequently, no statistically affirmed statement can be made. This is equally the case when comparing the feedback-model and the feedforward-model. Nevertheless, an explicit improvement is visible in Figure 3.9.

There are some drawbacks of the feedback model, though. For instance, the dense sampling most certainly leads to errors when the feedback signal fails. Subsequently to the failing, optical flow cannot be relieved from most of the rotational component. The *template model* might consequently systematically over- or underestimate the rotation. Take for example optical flow which contains a high rotational value when dense sampling around zero is given. The value might fall exactly in between two sampling points that span a large range of different rotational estimates. A precise rotational estimate will thus be impeded. Therefore it is necessary to find out the different possibilities for the origin of the prediction and thus the signal. It may be possible that the reduction of optical flow is carried out in two subsequent steps in two different areas of the brain. This idea is supported by the finding that stimuli in area VIP (Ventral Intraparietal Area) are coded head-centric [ZHB04]. VIP is believed to work similar to area MST, but neuronal responses are multimodal. A direct path from MST to VIP would possibly circumvent the integration in a head direction network, though.

The prediction signal might have different places where it is applied. My implementation incorporates it after the head direction network was updated according to the fused result. Thus, coincidence detectors to detect a change in the head direction network could be used to make this step biologically plausible. Such detectors are known to exist in the brain and can be easily implemented with a sparse number of neurons. Other possible junctions for the prediction signal are either directly at the optical flow estimation or during the fusion. Both methods were considered for feasibility but not further analyzed.

A comparison of my model to the Kalman filter seems natural. In fact, both the model and the Kalman filter employ a prediction-correction approach. One of the differences is the creation of the prediction signal. Whereas the Kalman filter creates its prediction from the estimated system model, the prediction that is used in the proposed model may have various sources as just discussed. On the other hand, different parts of the Kalman filter can be directly related to components of my model. For example, the Kalman gain corresponds to the different confidence measures used. My model has an advantage to the Kalman filter, though. Lifelong service is one of the designated goals in robotics research. Using a Kalman filter for pose estimation on a robot that is used on longer terms imposes different problems. For example, to calculate the *a posteriori* state estimate and error covariance matrix at a certain time, all observations that were made up until this moment are required. Over long periods, an increase in memory and processing power consumption is the consequence. In contrast to this, my model uses an approximately fixed size of memory and processing power.

In summary, most of the results can be directly explained. However, some of the findings will need further investigation. For instance, the results of the different input sizes need to be analyzed on the basis of more simulations.

4.2 CONCLUSION

Estimating the self-motion from visual input with high accuracy is a difficult task. In order to face the problem, I proposed a biologically inspired model. The model uses feedback to process visual input in order to calculate estimations of the rotation and translation. I was able to show that a feedback can help to improve the precision of the estimate when optical flow is used as input. In addition, I demonstrated that a simple network of head direction cells is capable of fusing different sensory inputs. With the help of the head direction network, a prediction signal can be used to calculate the feedback. Hence, the research questions are answered. With the help of a head direction network, different sensory inputs can be fused and subsequently used as a feedback in order to improve the estimation quality.

5 OUTLOOK

The model and its results directly lead to new research. For instance, some of the presented results need to be re-investigated. The simulations should be repeated with different trajectories. This might give more insight into the results for, e.g. the different numbers of input vectors. On the other hand, the constraints that were imposed on the model need to be successively removed.

Extending the model will require a lot of simulation time. Hence, it will be necessary for efficiency to find acceleration structures which reduce the computational cost of the model and its modules. It might be necessary to rewrite different parts, or to port them to massively parallel architectures like CUDA or OpenCL. Doing so will most probably disclose new and exciting problems and solutions in the domain of parallel computation of neural networks.

Aside from these technical enhancements, the model itself could be extended in many different ways. One such way would be to drive the model into a direction where the combination of different processing paths is selected by estimation quality. To illustrate this idea, assume a model that uses the same processing paths as my model – *epipolar* and *template model* estimation – but keeps track of the distance and rotation covered over time and adaptively incorporates estimates not until their assumed error falls below some certain quality in percent. For instance, the findings in Section 3.1 suggest that the *epipolar* path is only used after large translations or rotations because the ratio between error and estimate will be beneficial towards the estimate. The path could thus work as a sort of recalibration. In addition, the integration of distances could be accomplished using more complex, biologically inspired neural field networks that are currently used in research to simulate place cells and grid cells (e.g. [BF09]). Other examples to enhance the model would be to use additional paths to estimate self-motion (e.g. [TS96]) or to use different camera models.

Real world data should be considered to be tried with the model. It was shown only recently in [MK11] that even small discrepancies between real footage and a physically correct simulation of the same scene lead to significantly different flow estimates. Hence, it should be examined if the model is robust against those differences.

Furthermore, it is necessary to keep track of new findings in neuroscience. They need to be incorporated into the biologically inspired model of visually driven navigation. Thus, it may be possible to validate or falsify the findings.

Supplementary, it might lead to proposals of new biological models and to new research directions in neuroscience.

ADDITIONAL SETUP PARAMETERS

Table A.1: Setup parameters for the MotionAlgo application.

MotionAlgo Setup	
image width	determined by reading input files
image height	determined by reading input files
detection algorithm	CENSUS
number of scales	1
generate confidence values	true
use saliency map	true
past steps used	1
future steps used	0
maximum number of hypotheses V1	5
maximum number of hypotheses MT	5
subsampling factor $V1 \rightarrow MT$	5
maximum detected speed	120 [pixels]
feedback enhancement C	100
saliency preblur factor	1
Hanning blur size Δs	3

VIRTUAL SEQUENCE

B

The virtual sequence was modeled using `blender`. Figure B.1 gives a floor plan of the scene. The agent is traveling on the orange path in clockwise direction. The `.blend` file is published under a Creative Commons license and available on `http://rochus.net`.

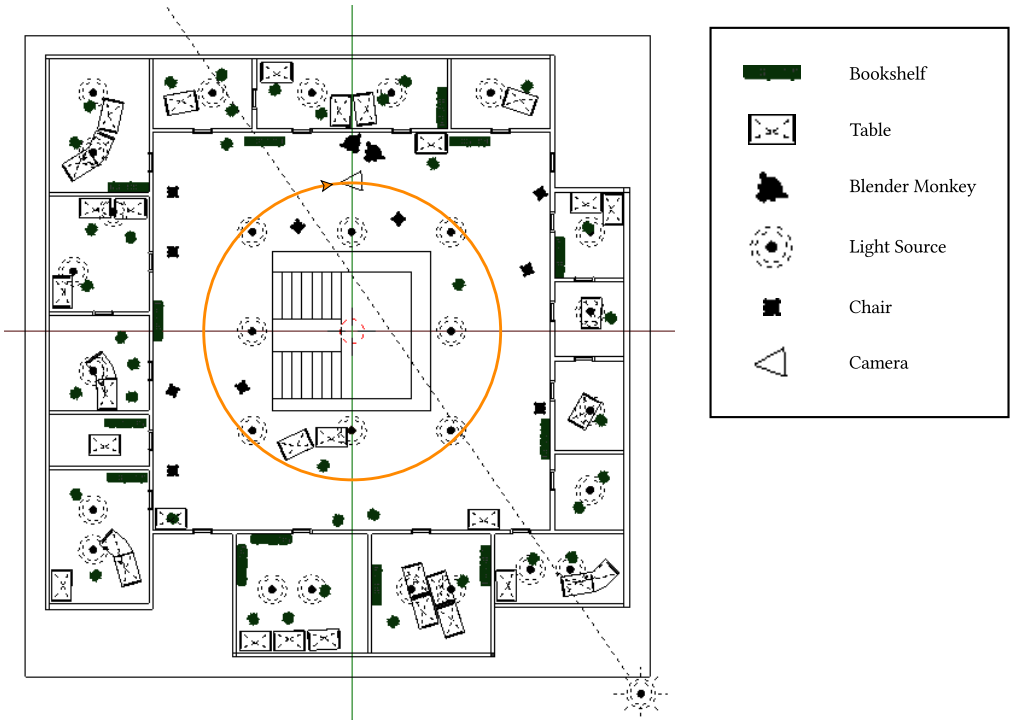


Figure B.1: Virtual sequence. The sequence contains different objects. Each object is textured in a way to yield approximately the same number of tracked features as real world footage of the Ulm University.

SOFTWARE

C

In addition to the proposed model, I wrote some utilities and extensions to already available software products. The goal was to have self-contained tools which can be used not only for the biologically inspired model of visually driven navigation, but for other projects and research as well. While writing the software, I payed attention to be standards compliant to the language which was used in each case. This is especially the case for the CenSurE implementation and the `exrtools` collection which were written in `C` and `C++`, respectively.

This chapter contains only a short overview of the software. Licensing details, man pages and technical documentation for each software ships with the source code.

C.1 CENSURE, LIBCENSURE

At the time of this writing, the implementation of the feature detection and tracking based on CenSurE is available as a command line application of name `censure`. The core of the application, which is written in `ANSI C99` will be stripped of the surrounding parts, which are written in `C++` due to the need to process `OpenEXR` files, and published as `libcensure`. Hence it will be possible to link against it and use it, for example, from within Matlab. Newest versions and updates to the `censure` standalone application as well as `libcensure` will be available on `http://git.rochus.net`.

C.2 EXRTOOLS

The `exrtools` are a collection of tools to process OpenEXR files. They will be available on `http://git.rochus.net` shortly after this writing.

- `exr2pgm` converts OpenEXR files to PGM.
- `exrcvview` uses OpenCV to display a sequence of OpenEXR files as video.
- `exrflow` is the main contribution of the `exrtools` collection. `exrflow` takes a CSV file, a focal length and OpenEXR files as input. The CSV file needs to contain the camera matrix for each frame that is passed as OpenEXR

file. Subsequently, `exrflow` calculates ground truth optical flow for the movement between every consecutive OpenEXR files according to [LP80].

The implementation with respect to the CSV file may change in future releases, as it is possible to store meta-data such as matrices directly to OpenEXR files.

C.3 MATEXR

The `matexr` files contain two extension to Matlab in Matlab's `mex` format. The first extension reads OpenEXR files, converts them to gray scale and passes them to the calling Matlab function. The second extension transforms from OpenEXR's linear RGB to sRGB instead of the gray scale conversion.

C.4 CAMLOCPOS

In order to export the camera matrix with respect to the world coordinate system from `blender`, I had to write a small extension called `camlocpos`. The `blender` extension integrates directly with blender in order to make it easy to use. `camlocpos` provides settings in the Render panel of `blender`'s main UI. `camlocpos` is already available on `http://git.rochus.net`.

BIBLIOGRAPHY

[AKB08] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In David A. Forsyth, Philip H. S. Torr, and Andrew Zisserman, editors, *ECCV (4)*, volume 5305 of *Lecture Notes in Computer Science*, pages 102–115. Springer, 2008. (cited on p. 4, 5, 29, 30, 32)

[Ama90] S.-i. Amari. Mathematical foundations of neurocomputing. *Proceedings of the IEEE*, 78(9):1443 –1463, sep 1990. (cited on p. 38)

[Ama91] Shun-ichi Amari. Mathematical theory of neural learning. *New Generation Computing*, 8:281–294, 1991. 10.1007/BF03037088. (cited on p. 38)

[BETVG08] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110:346–359, June 2008. (cited on p. 29)

[BF09] Yoram Burak and Ila R. Fiete. Accurate path integration in continuous attractor network models of grid cells. *PLoS Comput Biol*, 5(2):e1000291, 02 2009. (cited on p. 38, 67)

[BKS03] Robert Baczyk, Andrzej Kasinski, and Piotr Skrzypczynski. Vision-based mobile robot localization with simple artificial landmarks. In *Artificial Landmarks, Prepr. 7th IFAC Symp. on Robot Control, Wroclaw*, pages 217–222, 2003. (cited on p. 6, 7)

[BN07] Pierre Bayerl and Heiko Neumann. A fast biologically inspired algorithm for recurrent motion estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(2):246–260, 2007. (cited on p. 16, 19)

[BSL+07] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. In *In Proceedings of the IEEE International Conference on Computer Vision*, 2007. (cited on p. 21)

[BT07] Omri Barak and Misha Tsodyks. Persistent activity in neural networks with dynamic synapses. *PLoS Comput Biol*, 3(2):e35, 02 2007. (cited on p. 38)

[DLBA04] Thomas Degris, Loic Lacheze, Christophe Boucheny, and Angelo Arleo. A spiking neuron model of head-direction cells for robot orientation. In S. Schaal, A. Ijspeert, A. Billard, S.Vijayakumar, J. Hallam, and J.A. Meyer, editors, *Simulation of Adaptive Behavior: From Animals to Animats*, pages 255–263, 2004. (cited on p. 2)

[EMC09] Mosalam Ebrahimi and Walterio Mayol-Cuevas. Susure: Speeded up surround extrema feature detector and descriptor for realtime applications. In *"Workshop on Feature Detectors and Descriptors: The State Of The Art and Beyond" as part of IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2009*, June 2009. (cited on p. 4, 5, 29)

[FSG+97] Matthias O. Franz, Bernhard Schölkopf, Philipp Georg, Hanspeter A. Mallot, and Heinrich H. Bülthoff. Learning view graphs for robot navigation. *Autonomous Robots*, 5:111–125, 1997. (cited on p. 4, 5)

[FT06] Mark C. Fuhs and David S. Touretzky. A spin glass model of path integration in rat medial entorhinal cortex. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 26(16):4266–4276, April 2006. (cited on p. 38)

[FWW09] C C Alan Fung, K Y Michael Wong, and Si Wu. Tracking dynamics of two-dimensional continuous attractor neural networks. *Journal of Physics Conference Series*, 197:10, 2009. (cited on p. 38)

[Har92] Richard I Hartley. Estimation of relative camera positions for uncalibrated cameras. *Compute*, 92(1):579–587, 1992. (cited on p. 37)

[Har97] R. Hartley. In Defense of the Eight-Point Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, June 1997. (cited on p. 36)

[HO06] Matthew Harker and Paul O'Leary. First order geometric distance (the myth of sampsonus). In Mike J. Chantler, Robert B. Fisher, and Emanuele Trucco, editors, *BMVC*, pages 87–96. British Machine Vision Association, 2006. (cited on p. 36)

[HS88] Chris Harris and Mike Stephens. *A combined corner and edge detector*, volume 15, pages 147–151. Manchester, UK, 1988. (cited on p. 32)

[HZ00] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000. (cited on p. 34, 35, 36)

[JCM00] Andrew E. Johnson, Yang Cheng, and Larry H. Matthies. Machine vision for autonomous small body navigation. In *In Proceedings of IEEE Aerospace 2000 Conference*, 2000. (cited on p. 6, 7)

[JS98] Taube Jeffrey S. Head direction cells and the neurophysiological basis for a sense of direction. *Progress in Neurobiology*, 55(3):225 – 256, 1998. (cited on p. 2, 13)

[KA08] Kurt Konolige and Motilal Agrawal. Frameslam: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics*, 24(5):1066–1077, October 2008. (cited on p. 4, 5)

[KAS07] Kurt Konolige, Motilal Agrawal, and Joan Solà. Large-scale visual odometry for rough terrain. In Makoto Kaneko and Yoshihiko Nakamura, editors, *ISRR*, volume 66 of *Springer Tracts in Advanced Robotics*, pages 201–212. Springer, 2007. (cited on p. 4, 5, 29)

[KM07] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007. (cited on p. 6, 7)

[KSN+05] Jeffrey Krichmar, Anil Seth, Douglas Nitz, Jason Fleischer, and Gerald Edelman. Spatial navigation and causal analysis in a brain-based device modeling cortical-hippocampal interactions. *Neuroinformatics*, 3:197–221, 2005. 10.1385/NI:3:3:197. (cited on p. 4, 5)

[LBJL07] Thomas Lemaire, Cyrille Berger, Il-Kyun Jung, and Simon Lacroix. Vision-based slam: Stereo and monocular approaches. *International Journal of Computer Vision*, 74:343–364, 2007. 10.1007/s11263-007-0042-3. (cited on p. 4, 5)

[LCB11] Li Ling, Eva Cheng, and Ian S. Burnett. Eight solutions of the essential matrix for continuous camera motion tracking in video augmented reality. In *ICME*, pages 1–6. IEEE, 2011. (cited on p. 37)

[LH87] H. C. Longuet-Higgins. Readings in computer vision: issues, problems, principles, and paradigms. chapter A computer algorithm for reconstructing a scene from two projections, pages 61–62. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987. (cited on p. 34)

[LH09] Taehee Lee and Tobias Höllerer. Multithreaded hybrid feature tracking for markerless augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 15:355–368, May 2009. (cited on p. 6, 7)

[LLB08] Simon Lacroix, Thomas Lemaire, and Cyrille Berger. More vision for slam. In Danica Kragic and Ville Kyrki, editors, *Unifying Perspectives in Computational and Robot Vision*, volume 8 of *Lecture*

Notes in Electrical Engineering, pages 129–147. Springer US, 2008. (cited on p. 6, 7)

[Low99] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society. (cited on p. 29)

[LP80] H. C. Longuet-Higgins and K. Prazdny. The Interpretation of a Moving Retinal Image. *Royal Society of London Proceedings Series B*, 208:385–397, July 1980. (cited on p. 16, 20, 43, 74)

[MBJ+06] Bruce L. McNaughton, Francesco P. Battaglia, Ole Jensen, Edvard I Moser, and May-Britt Moser. Path integration and the neural basis of the 'cognitive map'. *Nat Rev Neurosci*, 7(8):663–678, August 2006. (cited on p. 1, 2)

[MGG10] Himanshu Mhatre, Anatoli Gorchetchnikov, and Stephen Grossberg. Grid cell hexagonal patterns formed by fast self-organized learning within entorhinal cortex. *Hippocampus*, 000:1–17, 2010. (cited on p. 38)

[MHB+08] Tim K. Marks, Andrew Howard, Max Bajracharya, Garrison W. Cottrell, and Larry Matthies. Gamma-slam: Using stereo vision and variance grid maps for slam in unstructured environments. In *IEEE International Conference on Robotics and Automation 2008*, pages 3717–3724, 2008. (cited on p. 4, 5)

[Mil08] Michael Milford. *Robot Navigation from Nature - Simultaneous Localisation, Mapping, and Path Planning based on Hippocampal Models*, volume 41 of *Springer Tracts in Advanced Robotics*. Springer, 2008. (cited on p. 11)

[MK11] S. Meister and D. Kondermann. Real versus realistically rendered scenes for optical flow evaluation. In *Electronic Media Technology (CEMT), 2011 14th ITG Conference on*, pages 1 –6, march 2011. (cited on p. 67)

[MKM08] Edvard I. Moser, Emilio Kropff, and May-Britt B. Moser. Place cells, grid cells, and the brain's spatial representation system. *Annual review of neuroscience*, 31(1):69–89, 2008. (cited on p. 1)

[Moa02] Maher Moakher. Means and averaging in the group of rotations, 2002. (cited on p. 44)

[MRS06] Annalisa Milella, Giulio Reina, and Roland Siegwart. Computer vision methods for improved mobile robot state estimation in challenging terrains. *Journal of Multimedia*, 1(7):49–61, 2006. (cited on p. 6, 7)

[MW10] Michael Milford and Gordon Wyeth. Persistent navigation and mapping using a biologically inspired slam system. *Int. J. Rob. Res.*, 29:1131–1153, August 2010. (cited on p. 4, 5)

[OB05] John O'Keefe and Neil Burgess. Dual phase and rate coding in hippocampal place cells: theoretical significance and relationship to entorhinal grid cells. *Hippocampus*, 15(7):853–866, 2005. (cited on p. 38)

[Per92] John A. Perrone. Model for the computation of self-motion in biological systems. *J. Opt. Soc. Am. A*, 9(2):177–194, Feb 1992. (cited on p. 8, 20, 25, 52)

[PS94] J A Perrone and L S Stone. A model of self-motion estimation within primate extrastriate visual cortex. *Vision Res*, 34(21):2917–38, 1994. (cited on p. 20)

[PS98] John A. Perrone and S. Stone. Emulating the visual receptive-field properties of mst neurons with a template model of heading estimation. *J. Neuroscience*, 18:5958–5975, 1998. (cited on p. 20, 49)

[RD09] Ananth Ranganathan and Frank Dellaert. Bayesian surprise and landmark detection. In *ICRA*, pages 2017–2023. IEEE, 2009. (cited on p. 4, 5)

[RET96] A David Redish, Adam N Elga, and David S Touretzky. A coupled attractor model of the rodent head direction system, 1996. (cited on p. 2)

[Rol99] Edmund T. Rolls. Spatial view cells and the representation of place in the primate hippocampus. *Hippocampus*, 9:467–480, 1999. (cited on p. 1)

[Rol07] Edmund T. Rolls. An attractor network in the hippocampus: theory and neurophysiology. *Learning & memory (Cold Spring Harbor, N.Y.)*, 14(11):714–731, November 2007. (cited on p. 38)

[RPD10] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32:105–119, 2010. (cited on p. 29)

[SCS+09] Denis Sheynikhovich, Ricardo Chavarriaga, Thomas Strösslin, Angelo Arleo, and Wulfram Gerstner. Is there a geometric module for spatial orientation? insights from a rodent navigation model. *Psychol Rev*, 116(3):540–66, 2009. (cited on p. 4, 5)

[Sel96] J. M. Selig. *Geometrical Methods in Robotics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1996. (cited on p. 44)

[SK09] Hee Sohn and Byung Kim. Vecslam: An efficient vector-based slam algorithm for indoor environments. *Journal of Intelligent & Robotic Systems*, 56:301–318, 2009. 10.1007/s10846-009-9313-2. (cited on p. 6, 7)

[SRB10] Deqing Sun, Stefan Roth, and Michael J. Black. Secrets of optical flow estimation and their principles. In *In Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2432–2439, June 2010. (cited on p. 25, 64)

[TS96] C. Tomasi and J. Shi. Image deformations are better than optical flow. *Mathematical and Computer Modeling*, 24:165–175, 1996. (cited on p. 67)

[TTH96] Tina Y. Tian, Carlo Tomasi, and David J. Heeger. Comparison of approaches to egomotion computation. In *In CVPR*, pages 315–320, 1996. (cited on p. 44)

[WA05] Si Wu and Shun-Ichi Amari. Computing with continuous attractors: Stability and online aspects. *Neural Comput.*, 17:2215–2239, October 2005. (cited on p. 38)

[WHA08] Si Wu, Kosuke Hamaguchi, and Shun-Ichi Amari. Dynamics and computation of continuous attractors. *Neural computation*, 20(4):994–1025, April 2008. (cited on p. 38)

[WT00] W Wang and H T Tsui. A svd decomposition of essential matrix with eight solutions for the relative positions of two perspective cameras. *International Conference on Pattern Recognition ICPR 2000*, 1:362–365, 2000. (cited on p. 37)

[XHS02] Xiaohui Xie, Richard H. Hahnloser, and H. Sebastian Seung. Double-ring network model of the head-direction system. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 66(4 Pt 1), October 2002. (cited on p. 38)

[ZHB04] Tao Zhang, Hilary W. Heuer, and Kenneth H. Britten. Parietal area VIP neuronal responses to heading stimuli are encoded in head-centered coordinates. *Neuron*, 42(6):993–1001, June 2004. (cited on p. 65)

[ZY10] Haixian Zhang and Zhang Yi. Dynamic shift mechanism of continuous attractors in a class of recurrent neural networks. In *Cybernetics and Intelligent Systems (CIS), 2010 IEEE Conference on*, pages 339 –343, june 2010. (cited on p. 38)

[ZYT+09] Eric A. Zilli, Motoharu Yoshida, Babak Tahvildari, Lisa M. Giocomo, and Michael E. Hasselmo. Evaluation of the oscillatory interference model of grid cell firing through analysis and measured period variance of some biological oscillators. *PLoS Comput Biol*, 5(11):e1000573, 11 2009. (cited on p. 38)

[ZZL94] Ramin Zabih, , Ramin Zabih, and John Wood Ll. Non-parametric local transforms for computing visual correspondence. pages 151–158. Springer-Verlag, 1994. (cited on p. 19)

DECLARATION

I hereby declare that this thesis was performed and written on my own and that references and resources used within this work have been explicitly indicated.

I am aware that making a false declaration may have serious consequences.

Ulm, 19th March 2012

Nicolai Sebastian Waniek