UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

Universitat de les Illes Balears
Departament de Matemàtiques i Informàtica

Visual Underwater Cable/Pipeline Tracking

Diplomarbeit

zur Erlangung des Grades eines

Diplom-Informatikers
im Studiengang Computervisualistik

vorgelegt von

Stephan Wirth

Betreuer:	Dr. Alberto Ortiz, Departament de Matemàtiques i Informàtica, Universitat de les Illes Balears
Erstgutachter:	Dr. Alberto Ortiz, Departament de Matemàtiques i Informàtica, Universitat de les Illes Balears
Zweitgutachter:	Prof. Dr.-Ing. Dietrich Paulus, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau

Koblenz, im Dezember 2007

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Die Richtlinien der Arbeitsgruppe für Studien- und Diplomarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. ja ☒ nein ☐

Der Veröffentlichung dieser Arbeit im Internet stimme ich zu. ja ☒ nein ☐

Koblenz, den 13. Dezember 2007

Übersicht

Die automatische Detektion der Lage und Ausrichtung von Unterwasser-Kabeln oder -Pipelines in Kamerabildern ermöglicht es, Unterwasserfahrzeuge autonome Kontrollfahrten durchführen zu lassen. Durch Pflanzenwuchs auf und in der Nähe von Kabeln bzw. Pipelines wird deren visuelle Erfassung jedoch erschwert: Die Bestimmug der Lage über die Detektion von Kanten mit anschließender Linien-Extraktion schlägt oft fehl. Probabilistische Ansätze sind hier den deterministischen überlegen. Durch die Modellierung von Wahrscheinlichkeiten kann trotz geringer Anzahl von extrahierten Merkmalen eine Aussage über den Zustand des Systems getroffen werden. Diese Arbeit stellt ein neues auf Partikelfiltern basierendes Tracking-System für die Verfolgung von Kabeln und Pipelines in Bildsequenzen vor. Umfangreiche Experimente auf realistischen Unterwasser-Videos zeigen die Robustheit und Performanz des gewählten Ansatzes sowie Vorteile gegenüber vorangegangenen Arbeiten.

Abstract

The automatic detection of position and orientation of subsea cables and pipelines in camera images enables underwater vehicles to make autonomous inspections. Plants like algae growing on top and nearby cables and pipelines however complicate their visual detection: the determination of the position via border detection followed by line extraction often fails. Probabilistic approaches are here superior to deterministic approaches. Through modeling probabilities it is possible to make assumptions on the state of the system even if the number of extracted features is small. This work introduces a new tracking system for cable/pipeline following in image sequences which is based on particle filters. Extensive experiments on realistic underwater videos show robustness and performance of this approach and demonstrate advantages over previous works.

Contents

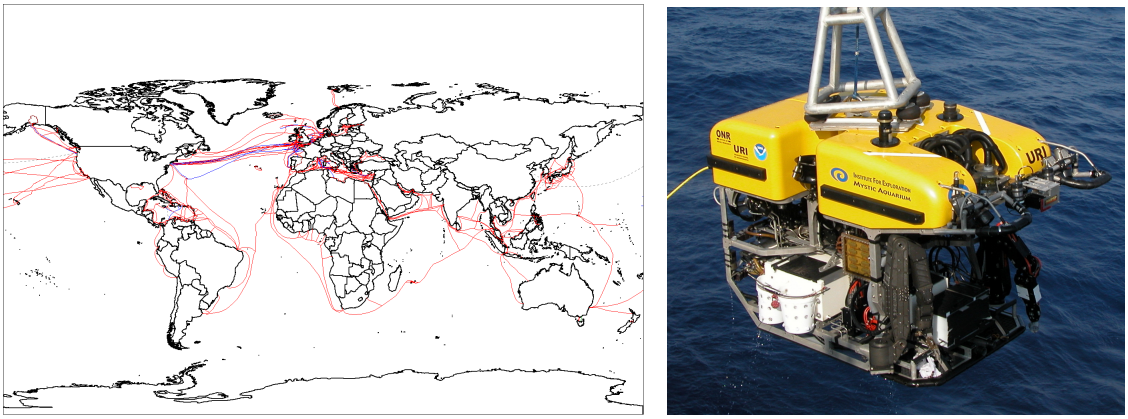
List of Figures

Chapter 1

Introduction

1.1 Motivation

The world gets more and more connected. Conduits for natural resources and energy as well as for information are constructed to build connections between countries and between continents (cf. figure 1.1(a)). Pipelines transport gas or oil and cables transport electric power or information. For connections between islands and mainland and between continents, cables and pipelines have to be layed through the sea. These undersea cables and pipelines are partly exposed to open water where salt, plants, moving sand, fishing activities, anchors and even shark bites can damage the conduits so that they need to be monitored from time to time. Inshore, divers can control the conduits, while offshore – with increasing depth – underwater vehicles have to fulfill this task. To control cables and pipelines with a remote operated vehicle (ROV, figure 1.1(b)) the operator has to steer the vehicle following the conduit by watching the camera images that are sent by the vehicle to the ship. At the same time, the operator – or another person – has to search for defects. Such a manual visual control is a very tedious job and tends to fail if the operator looses concentration. Therefore and because of the high cost of ROVs, various research groups [BTL$^+$97, AO03, AKK$^+$02] are developing autonomous underwater vehicles (AUVs) that are able to fulfill this task without any user interaction (figure 1.2(a)). The sensors used to determine the cable's or pipeline's position and orientation are mainly sonar, magnetome-



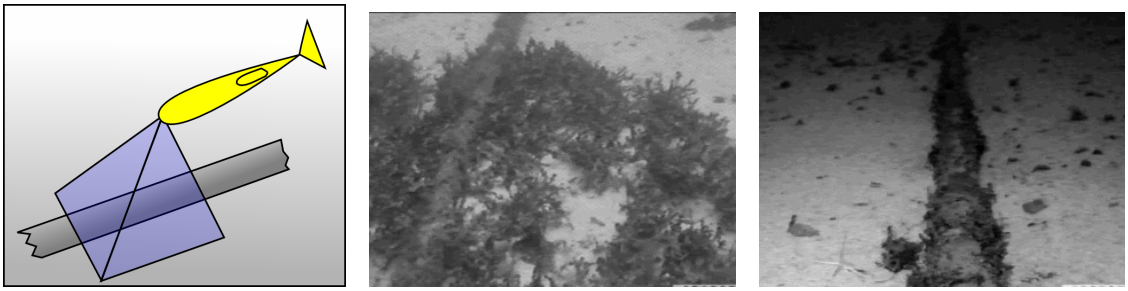
(a) (b)

Figure 1.1: (a) Undersea cables in the world; (b) a typical Remote Operated Vehicle (ROV)

ters and cameras. The latter, as illustrated by Santos-Victor and Sentieiro in [SVS94], are powerful sensors in the field of underwater navigation, especially for object recognition and cable/pipeline following although their usage is rather difficult: recently installed cables and pipelines have clearly defined shape and colour and can therefore be easily detected with cameras, but, when they get older, their visual appearance changes drastically. Marine flora tends to grow on top and in the neighbourhood of cables and pipelines and moving mud and sand can make them hardly visible (see figure 1.2(b)). In the deep sea, robots have to carry their own light sources because of the lack of ambient light, which results in irregular lighting conditions (see figure 1.2(c)). Additionally, light suffers from absorption and dispersion along its propagation in the oceanic medium what gives rise to blurred and low-contrast images.

1.2 The cable/pipeline tracking problem

The detection of clear shaped cables and pipelines with straight, high-contrast borders in camera images can be considered as solved. Balasuriya et. al. demonstrated in [BU99] that in this case, simple border enhancing filters followed by Hough transformation ([HA62]) suffice to determine the cable/pipeline position. This method does not work on noisy



(a) (b) (c)

Figure 1.2: (a) Cable/Pipeline tracking scenario: An autonomous underwater vehicle (AUV) follows a conduit without any user interaction and gives only alerts when it detects defects; (b) image of an undersea cable whose borders are hard to detect; (c) bad lighting conditions in deep sea.

images of old cables (figure 1.2(b)), where borders of rocks and marine growth can give a higher response in the Hough Transform than the cable/pipeline itself.

Ortiz et al. have developed a camera based approach for cable tracking ([OSO02]) that is able to deal with this kind of images of old cables. They use a more complex image processing algorithm which includes segmentation, contour and line extraction, line segment fitting and grouping. Another method developed by Grau et al. ([GCA98]) uses texture descriptors and is also able to handle realistic images. However, none of the above mentioned approaches really focusses on *tracking* cables and pipelines in image sequences in the sense of computer vision but in their *detection*. While Ortiz et al. and Balasuriya et al. use the detection results of one time step to predict a region of interest (ROI) for the next time step, the other groups examine each frame of the sequence separately. The prediction of a region of interest makes it possible to filter out wrong detection results or to limit the search space for faster computation, but the method still remains deterministic because it computes for each frame *one* cable position that is supposed to be exact. The position is not *tracked* frame by frame but determined from scratch for every new (ROI-)image.

For tracking it is necessary to think in probability distributions. If the position of an object in an image is given, the question of tracking is: which is the most *probable* position of this object in the next image? All information that is available like previous computation results or current sensor readings can be used to answer this question.

Figure 1.3: The Submarine Robot RAO-II.

1.3 Project Scope

This work is embedded in the project "Visual Inspection and Maintenance of Underwater Installations" of the Systems, Robotics and Vision Group (SRV), Mathematics & Computer Science Department, University of the Balearic Islands, Spain. The goal of the working group is to construct an AUV with the hull and the propulsion system of a SeaLion ROV (see figure 1.3) for undersea inspection tasks (see [OOB07]).

The present work develops a new visual cable/pipeline tracking algorithm based on particle filters ([IB98, AMGC02]), that can be used to control this AUV, enabling it to navigate along a cable or a pipeline respectively without any user interaction.

The AUV is not fully functional yet, but the SRV group possesses a set of underwater videos, taken from other AUVs and ROVs while following a cable. These videos are used as input data for the proposed approach, following the aim of detecting the cable's position and orientation for every frame of the sequences.

The decision on particle filters is based on their ability to handle and compute with multi-dimensional multi-modal probability distributions. As mentioned earlier, ambiguities may occur when objects which look similar to the conduits appear near the conduits. A particle filter is able to track different possibilities until the ambiguity is resolved. Additionally, particle filters are easy to implement and easy to understand and their utility and perfor-

mance have been proven in many works, not only in the field of computer vision (see for example [NKMvG03, FTBD01]).

1.4 Overview

In chapter 2, a review of cable tracking methods is given. Chapter 3 introduces the idea of Bayesian probabilistic tracking. Furthermore particle filters for tracking are introduced by the explanation of the condensation algorithm. In chapter 4, models for cable tracking[1] are developed, architecture and implementation details are explained. The chapter 5 presents experiments with real image sequences of undersea cables and their results. The last chapter summarizes the overall work and gives hints on possible improvements.

[1] *Note:* In the following the term "Cable tracking" is used instead of "Cable/Pipeline tracking", to ease the reading. This should not be understood as a lack of the proposed approach as cables and pipelines have a very similar appearance in undersea images.

Chapter 2

Cable Tracking Approaches

This chapter gives a review on different cable tracking approaches and points out their benefits and shortcomings.

2.1 Hough Cable Detection

Method

Matsumoto and Ito [MI95] as well as Balasuriya et al. [BTL$^+$97, BU99] use the Hough transform ([HA62]) for lines to extract cable borders out of an edge image. They assume that in an underwater scene the cable's borders are very particular and different from the rest of the scene. For the generation of the edge image, both groups use the Laplacian of Gaussian (LoG) filter (see figure 2.1).

The LoG filter is a convolution of a Gaussian and a Laplacian filter and is defined as

$$\Delta g(x,y) = -\frac{1}{\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \left(1 - \frac{x^2+y^2}{2\sigma^2}\right)$$

where σ defines the standard deviation of the used Gaussian function. The Gaussian component smoothes the image and the Laplacian component detects intensity changes. A

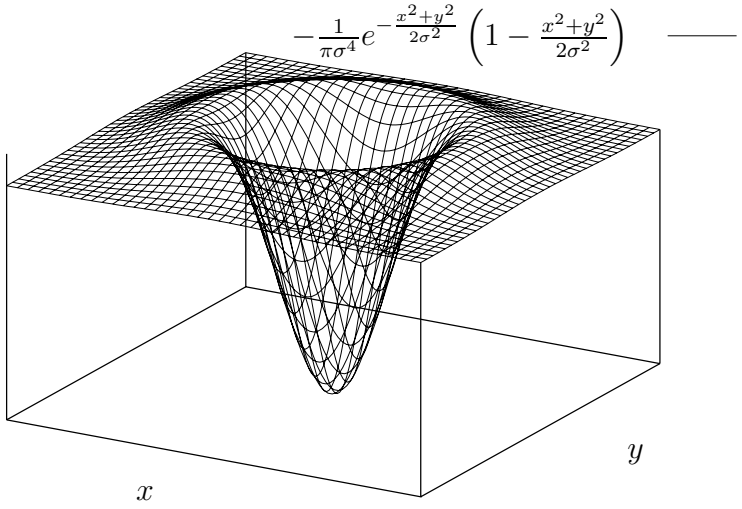


Figure 2.1: Laplacian of Gaussian Filter Kernel.

zero-crossing in the output image indicates a border in the original image. Examples of filtered cable images can be found in figure 2.2.

Before the application of the Hough transform, the edge image is converted into a binary image via thresholding, so that only strong borders survive.

Comments

For clearly shaped cable borders, as in the lower left part of the cable in the second image of figure 2.2, the detected border pixels form a straight line and can therefore be easily found. In the other cases, regression lines have to be found that approximate the main direction of the borders which is a problem that none of the named authors adresses.

Another issue for the application of the Hough tansform is its time consuming complexity. Balasuriya et al. as well as Matsumoto and Ito solve this problem by limiting the parameter space of the Hough transform based on previous computation results. Once the cable

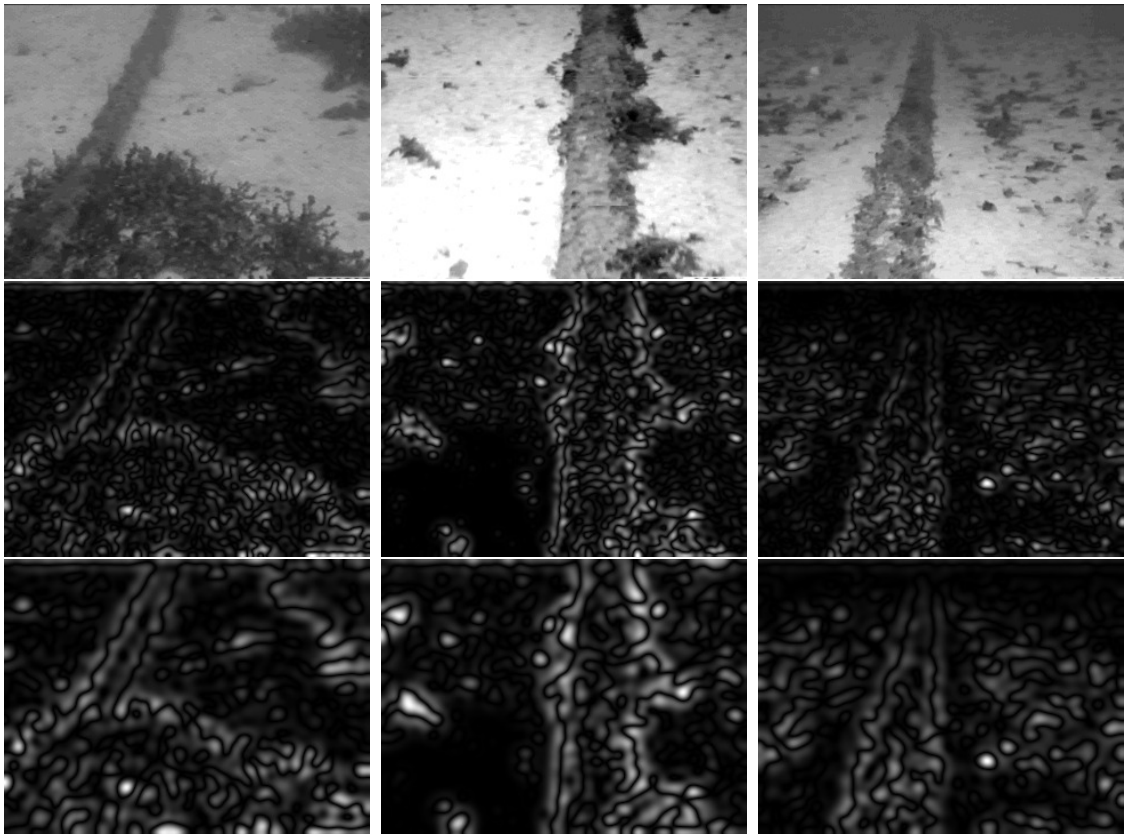


Figure 2.2: LoG-filtered images of underwater cables. The first row shows the original images. The other rows show the filtered images for different σ ($\sigma_1 = 3, \sigma_2 = 5$). Due to the fact that the filtered images have negative values, absolute values are shown. The borders appear as thin black lines in white areas. The brighter the sorrounding field of an edge, the higher is its intensity in the source image.

is successfully detected, the search space for line detection can be limited around the determined parameters, i.e. the Hough transform has to be computed only for a small range of angles and distances. However, the LoG filter still has to be applied on the whole image. If an ambiguous situation occurs, the wrong alternative might be tracked and – due to the limitation of the search space – the correct alternative might be lost.

2.2 Detecting Cable Texture

Method

Grau et al. propose a real-time architecture for cable tracking using texture descriptors in [GCA98]. The texture of an object characterizes its surface. It is assumed that the texture of cables differs from other textures that appear in underwater images. Grau et al. use five texture descriptors which have been heuristically selected based on human perception:

1. **Straightness:** indicates the straight lines density in a region
2. **Abruptness:** measures the sudden changes in the direction of the lines in a region
3. **Discontinuity:** indicates the amount of cut lines in an area
4. **Granularity:** is the degree of isolated pixels in the image
5. **Bluriness:** indicates soft changes in the light intensity in any direction

The first four descriptors are determined using filter masks, the last one by comparing a pixel to the pixels in its neighbourhood.

Grau et al. developed a hardware architecture that is able to perform the computation of the texture descriptors in video rate (50 frames/s). The research group used a set of underwater cable images to learn the texture parameters for different types of regions:

- **cable area** (high straightness and abruptness),
- **sand** (smooth areas),

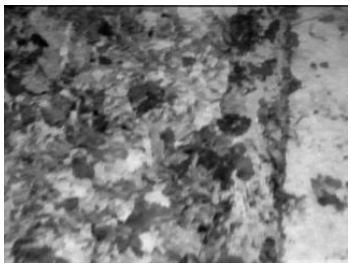


Figure 2.3: A cable image where the textures of the cable and the ground are very similar.

- **stones and vegetation** (high discontinuity and low straightness),
- **undefined regions** (blurry texture).

After the learning phase, the system was able to find cable regions in realistic underwater images.

Comments

Because of marine growth beside and on top of cables, the cable often has a similar texture to its environment (cf. figure 2.3). I assume that in these cases this approach tends to fail. The texture of the cable and the seabed might also change over time which makes the previous learned texture parameters useless. Furthermore the determination of the cable position out of the segmentation results is not adressed here but necessary if the system is to be used for the guidance of an underwater vehicle.

2.3 Using Segmentation

Method

The research group around Ortiz developed a more comprehensive approach for cable detection in underwater images ([AO03, OSO02]) which is based on a particular segmentation method that was originally introduced by Panda and Rosenfeld in [PR78] followed by

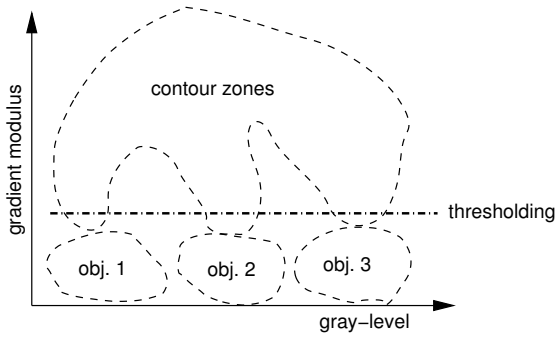


Figure 2.4: Ideal bidimensional histogram for three objects. The dashed lines mark the hills of the histogram.

contour extraction, edge detection, line extraction and line segment grouping. The output of the algorithm is the position and direction of the cable in the input image.

The first step of their algorithm is the division of the input image into a pyramid of cells (for instance, 2×2).

The next step is the segmentation process which is based on the idea that the cable is surrounded by a strong alignment of contour pixels. A bidimensional histogram is computed for each cell where one axis corresponds to the gray-levels and the other axis corresponds to a digital approximation of the gradient modulus (see figure 2.4) which is computed using the Sobel operator. Contour pixels have a high gradient modulus and are therefore situated in the upper part of the histogram whereas pixels from smooth regions fall into the lower part. In the ideal case of objects having different gray-levels, the histogram looks like figure 2.4. Now, the histogram is divided, discarding the pixels with high gradient modulus, which are the contour pixels, and is projected on the plane {gray-level, number of pixels}. The resulting histogram now stores the number of pixels for each grey-level counting only these pixels which are situated in smooth regions. The back projection of the "mountains" of this histogram define the segmentation of the image cell.

Once the image is segmented, the contours of the segments are used for the following steps. As the camera is looking slightly forward, the upper part of the image contains more noise and the contours of the cable are expected to be stronger defined in the lower part of the image. Considering this effect, the search for line segments starts from the

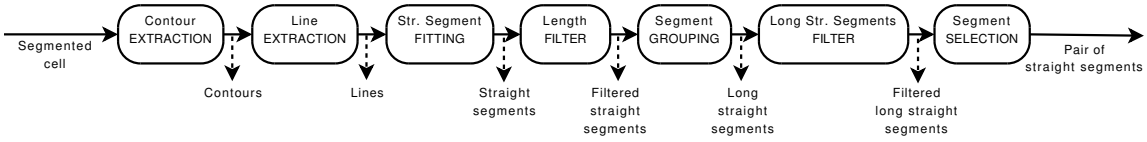


Figure 2.5: Cable Detection Process from [OSO02]

bottom of the image: once a contour pixel of a segmented region is found, adjacent pixels are selected according to a prediction of the cable orientation produced by a Kalman filter applied over the past cable parameters. Subsequently straight line segments are fitted on each set of these connected pixels. Short straight line segments are then discarded. Groups of co-linear line segments which fall into a strip-shaped region are used to form longer line segments. These long straight line segments are filtered again using different error metrics including the average of the gradient modulus of the contour pixels fitted by the straight segments considered. This results in rejecting lines that do not lie on or near an edge.

The expected width of the cable is afterwards used to find (nearly) parallel line pairs that are likely to be the cable borders regarding their degree of parallelism and their length. The most likely line pairs of each cell of the image are fused re-executing the line segment grouping, filtering and selection as previously described. The output of the system contains the pair of lines which has the highest score after the fusion. For an overview of the whole process, refer to figure 2.5.

For tracking, Ortiz et al. propose to use a Kalman filter to estimate the cable orientation and position as well as its width. A region of interest (ROI) is computed using the predicted cable parameters to limit the search region and hereby the computational cost for the next search step.

This approach was tested on realistic underwater images (a total of 1367 images with 320 $\times$ 240 pixels) with good results. The average computation time was 24ms per frame with an average success rate of 89.8% on a Pentium III processor with 800 MHz (see [AO03]).

Comments

The proposed cable detection method works well on realistic noisy submarine images. The complex segmentation and line determination works bottom-up: small features are grouped and filtered until only one pair of parallel lines remain. The tracking strategy enables the limitation of the search space to an ROI which reduces computation time. However, system dynamics are not used for this prediction. Additionally, the system is not able to track multiple hypothesis if ambiguities occur due to the unimodal nature of the Kalman filter.

2.4 Conclusions

All of the three analyzed cable tracking approaches are based mainly on deterministic image processing. Features are extracted and their number or parameters are compared to dynamic or non-dynamic thresholds with a binary result: the cable is successfully detected or not. The proposed tracking methods of 2.1 and 2.3 use former results to predict regions of interest which reduces time complexity but does not change the binary answer.

Probabilistic tracking approaches can provide better non-binary results. Trucco and Plakas come to a similar conclusion in their survey on video tracking ([TP06]), where they examine 28 papers on subsea video tracking. They found out that various techniques developed in the computer vision community have not yet been deployed underwater and expicitly say that one surprising omission is the use of particle filters for tracking.

The basis for particle filters is Bayesian tracking. Both will be introduced in the following chapter.

Chapter 3

Bayesian Tracking with Particle Filters

This chapter introduces Bayesian probabilistic tracking and its realization through particle filters. Particle filters are able to track multi-dimensional multi-modal probability distributions without any limitation according to their shape. The idea of particle filters has been introduced by several research groups independently (see [IB98, GSS93, Kit96]). This work mainly uses the condensation (**con**ditional **dens**ity propag**ation**) algorithm ([IB98, Mac00]) and the tutorial on particle filters from Arulampalam et. al ([AMGC02]) as guidelines.

3.1 Introduction

Bayesian tracking means tracking of probability distributions. The state of a dynamic system is estimated based on all available information. For this estimation, a probability density function (PDF) for the state is constructed. Depending on the application, this PDF can have any number of dimensions, e.g. the tracking of a circle of known size in a 2D-image requires the PDF to have two dimensions: one for the x-coordinate and one for the y-coordinate of the circle's center. If the size of the circle is unknown, it can be added as a third dimension to the PDF.

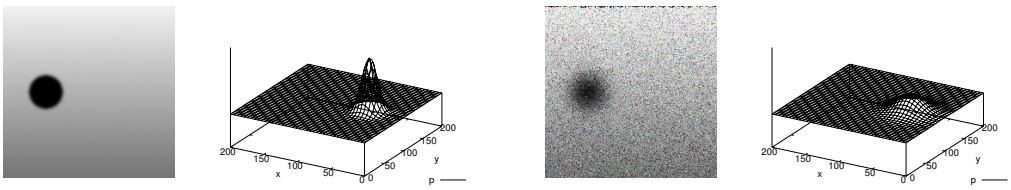


Figure 3.1: Examples for probability distributions that estimate the position of a circle in an image. In (a) the image has a good quality, thus, the probability distribution has a high peak. In (b) the image sensor captured a very noisy image which leads to a less defined peak in the PDF.

The necessity of using probabilistic representations for tracking can be justified by the characteristics of physical sensors: physical sensors always produce noise and therefore their measurements are never exact. For example, the position of an object in a camera image can never be determined exactly. The key to this problem is to determine its most *likely* position. The more exact the sensor is, the clearer is the peak in the PDF which estimates the object's position (cf. figure 3.1).

During time, the PDF is modified by integrating known system dynamics and measurements. The estimation process is therefore split into two stages:

1. the **prediction** integrates all information about the dynamics of the system and
2. the **update** integrates current measurements.

Figure 3.2 shows an example of the Bayesian estimation process.

3.2 Mathematical Model

In Bayesian tracking the state vector $\mathbf{x}$ is recursively estimated in discrete time steps. The transition of the state vector from step $t-1$ to the next step t is given by

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{v}_{t-1}) \tag{3.1}$$

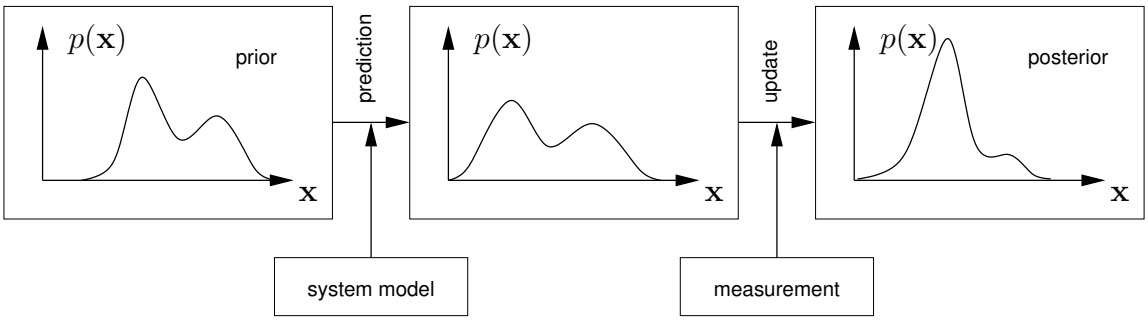


Figure 3.2: Steps of the Bayesian estimation process for a one dimensional PDF.

where $\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \longrightarrow \mathbb{R}^{n_x}$ is the transition function, $\mathbf{v}_{t-1}$ is an independent and identically distributed process noise and n_x and n_v are the dimensions of the state and process noise vectors, respectively. It is assumed that the process noise is independent of past and current states and that its PDF is known.

In discrete time steps, measurements $\mathbf{z}_t$ become available. The relation of measurement and state gives the measurement function $\mathbf{h} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \longrightarrow \mathbb{R}^{n_z}$:

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t, \mathbf{w}_t) \tag{3.2}$$

with $\mathbf{w}_t$ being another independent and identically distributed noise of known PDF. n_z and n_w are the dimensions of the measurement and measurement noise vectors, respectively.

Let $\mathbf{z}_{1:t} = \{\mathbf{z}_i, i = 1, \ldots, t\}$ be the set of all measurements up to time t. From a Bayesian perspective, the objective is now the recursive construction of the PDF of the state $\mathbf{x}_t$ depending on all measurements up to time t:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t})$$

It is assumed that the initial distribution $p(\mathbf{x}_0|\mathbf{z}_0) = p(\mathbf{x}_0)$ – the initial prior – is available. $\mathbf{z}_0$ is here the empty set of measurements.

The recursive construction of the state's PDF follows then the steps

$$p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}) \xrightarrow[\text{prediction}]{} p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) \xrightarrow[\text{update}]{} p(\mathbf{x}_t|\mathbf{z}_{1:t}) \quad ,$$

analog to figure 3.2. The first step involves the state evolution density $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ which can be defined using the transition function of equation 3.1:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \int \delta(\mathbf{x}_t - \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{v}_{t-1}))p(\mathbf{v}_t)d\mathbf{v}_t \tag{3.3}$$

with $\delta(\cdot)$ being the Dirac delta function. Using the state evolution density together with the prior $p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})$, the *prediction step* consists in calculating $p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})d\mathbf{x}_{t-1} \tag{3.4}$$

For the second step – the *update step* – the conditional PDF of the measurement $\mathbf{z}_t$ given the state $\mathbf{x}_t$, $p(\mathbf{z}_t|\mathbf{x}_t)$, has to be obtained, which can be done using the measurement function (equation 3.2):

$$p(\mathbf{z}_t|\mathbf{x}_t) = \int \delta(\mathbf{z}_t - \mathbf{h}(\mathbf{x}_t, \mathbf{w}_t))p(\mathbf{w}_t)d\mathbf{w}_t \tag{3.5}$$

The overall posterior $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ can be expressed by the Bayes formula

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) = k \underbrace{\int p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})p(\mathbf{x}_t|\mathbf{x}_{t-1})d\mathbf{x}_{t-1}}_{=p(\mathbf{x}_t|\mathbf{z}_{1:t-1})} p(\mathbf{z}_t|\mathbf{x}_t) \quad , \tag{3.6}$$

where k is the normalizing constant

$$k = \frac{1}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})} = \left(\int p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})d\mathbf{x}_t \right)^{-1}.$$

Equation 3.6 forms the basis for the optimal Bayesian solution. After the posterior density for time step t is constructed, it is used as prior for the next time step $t + 1$. This recursive propagation of the posterior density in general cannot be determined analytically. There exist solutions for a restrictive set of cases, such as the Kalman filter, which is able to compute the optimal posterior for Gaussian distributed noise and linear transition and measurement functions.

Particle filters provide a more general solution. They only approximate the PDF but do not have any restrictions on their shape. In contrast to the Kalman filter, which is restricted to Gaussian distributions, particle filters can model multi-modal distributions which makes them very useful for tracking. They can track two solutions in an ambiguous situation, discarding the wrong one when the ambiguity is resolved.

3.3 Particles – Approximating Probability Distributions

The idea of particle filters is to approximate probability distributions by a discrete number of weighted samples (Monte Carlo approximation), called *particles*.

Let $\mathbf{S} = \{(\mathbf{s}^{(i)}, \pi^{(i)}), i = 1, \ldots, N\}$ be the set of weighted samples. The weights of all samples – as they describe a probability distribution – sum up to one: $\sum_{i=1}^{N} \pi^{(i)} = 1$. The goal is to achieve a good representation of a PDF p by the sample set, so that

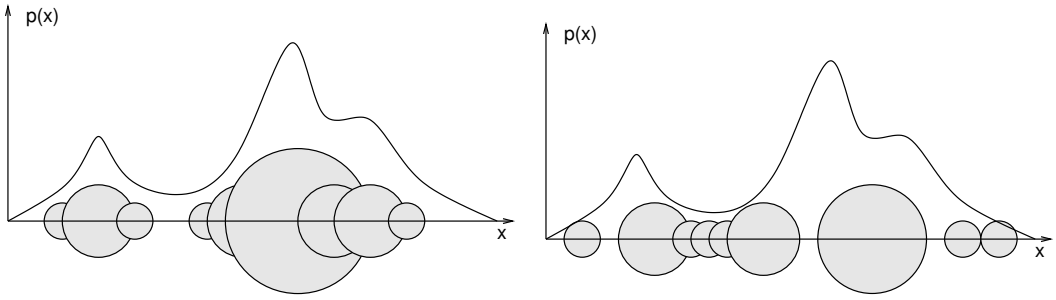
$$\sum_{i=1}^{N} g(\mathbf{s}^{(i)})\pi^{(i)} \approx \int g(\mathbf{x})p(\mathbf{x})d\mathbf{x} \tag{3.7}$$

for typical functions $g(\cdot)$ of the state space. This is an approximation in the sense that the left-hand side converges (in probability) to the right-hand side as $N \rightarrow \infty$.

An example for $g(\cdot)$ is $g(x) = x$, which computes the minimum mean square error (MMSE) estimate.

For tracking, we are interested in the most likely state(s), i.e. in the peaks of the PDF. Other regions, where the state's probability is low, are uninteresting. Hence, to obtain good approximations of PDFs for tracking, the particles have to be distributed around the PDF's modes (cf. figure 3.3). Obviously, the sum in equation 3.7 approximates the integral better, if the samples $\mathbf{s}^{(i)}$ are chosen where $p(\mathbf{x})$ is high.

The size of the particle set is essential for the qualitiy of the approximation. The higher the number of particles, the better they can approximate a PDF (figure 3.4). The number of particles needed for a good approximation depends on the shape and on the dimensions of the PDF. If the PDF has only a small number of modes and the rest of the function is near zero, few particles that cover the modes are sufficient. On the other hand, if the PDF



(a) (b)

Figure 3.3: One-dimensional probability distribution function and two different approximating particle sets. The size of the circles refer to the particles' weights. In (a) the particles are distributed around the "interesting" modes of the PDF, hence, they are a good approximation; in (b) the distribution of the particles is inappropriate.

is rather flat, more evenly distributed particles are needed. Increasing the dimensions of the PDF requires increasing the number of particles, too.

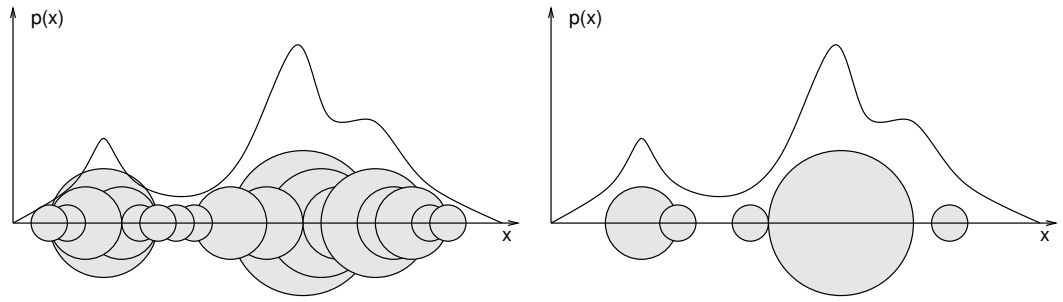
It is therefore neccessary to keep the dimensions of the state, which has to be estimated, as small as possible and to find a good measurement function that clearly marks out the most probable states.

3.4 The Condensation Algorithm

Given the particle representation of a PDF, the Condensation Algorithm defines how the set of particles has to be modified to apply the Bayes update from equation 3.6. This is the **con**ditional **dens**ity propag**ation**. The term "conditional density" is a synonym for the above used "posterior density".

The condensation process can be seen as step-wise computation of the Bayes update formula from equation 3.6 (cf. [Mac00], p. 10):

prior $\longmapsto$ *resampling* $\longmapsto$ $*p(\mathbf{x}_t|\mathbf{x}_{t-1}) \longmapsto \times p(\mathbf{z}_t|\mathbf{x}_t) \longmapsto$ posterior



(a) (b)

Figure 3.4: One-dimensional probability distribution function and two different approximating particle sets. The size of the circles refer to the particles' weights. In (a) the number of particles is high; in (b), only few particles are used, leading to a less precise approximation.

The symbol $*$ here denotes the convolution and $\times$ is the multiplication.

In one time step, the particle set

$$\mathbf{S}_{t-1} = \{(\mathbf{s}_{t-1}^{(i)}, \pi_{t-1}^{(i)}), i = 1, \ldots, N\},$$

that approximates the prior $p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})$ is transformed to the set

$$\mathbf{S}_t = \{(\mathbf{s}_t^{(i)}, \pi_t^{(i)}), i = 1, \ldots, N\},$$

which approximates the posterior $p(\mathbf{x}_t|\mathbf{z}_{1:t})$.

The single steps of the algorithm are described in the following (cf. figure 3.5).

3.4.1 Resampling

prior $\longmapsto$ *resampling* $\longmapsto * p(\mathbf{x}_t|\mathbf{x}_{t-1}) \longmapsto \times p(\mathbf{z}_t|\mathbf{x}_t) \longmapsto$ posterior

In the resampling phase, N new particles from the set $\mathbf{S}_{t-1}$ are drawn, choosing each particular element $\mathbf{s}_{t-1}^{(i)}$ with probability $\pi_{t-1}^{(i)}$. The elements with high weights may be chosen several times, leading to identical copies in the new set. Note that in this way, high peaks in the prior, which are interesting for tracking, are sampled with higher density.

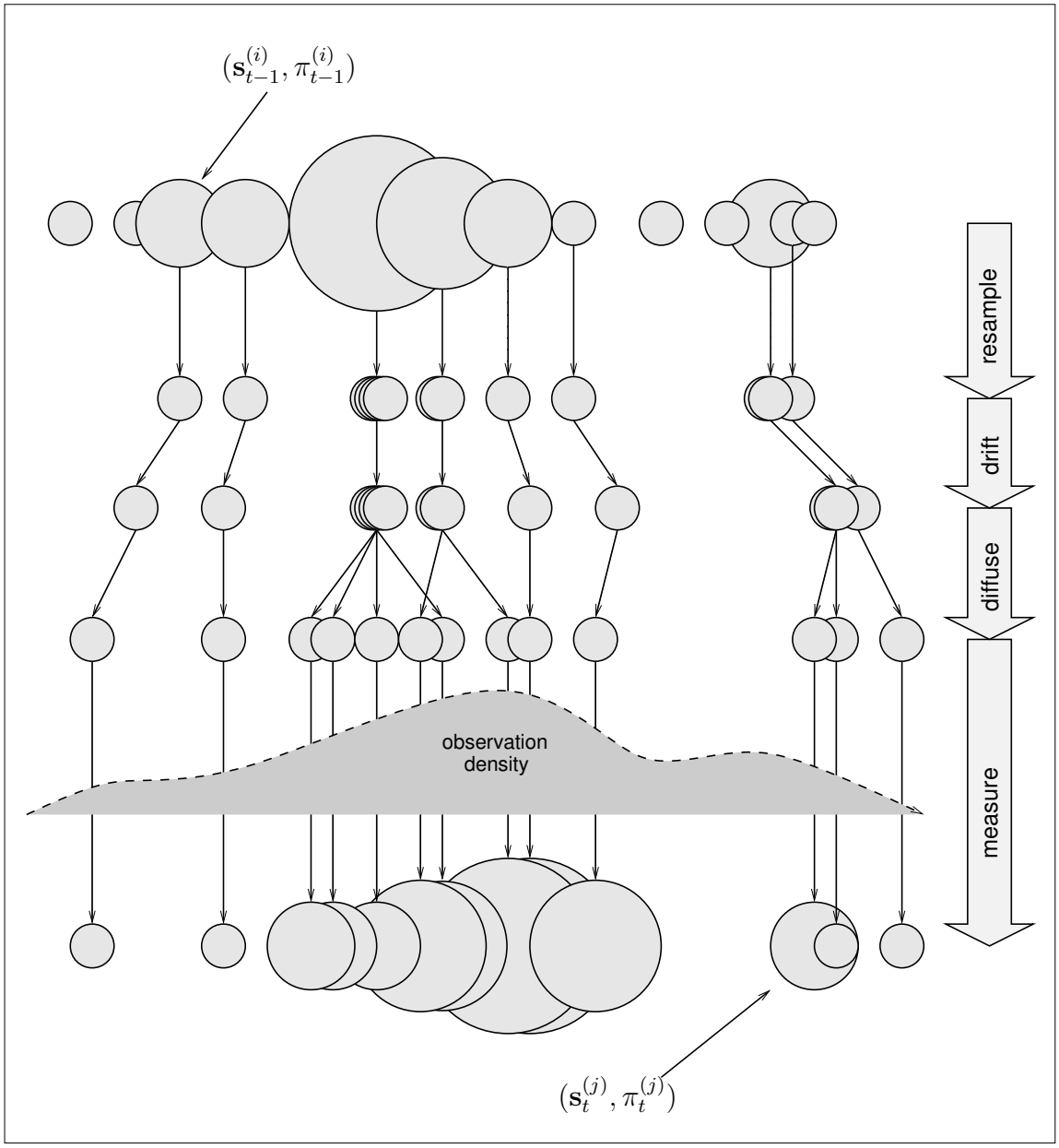


Figure 3.5: One time step of the condensation algorithm. The first row of circles is the particle representation of the prior density. The second row shows the particles after resampling. Note that particles with a high weight have multiple "descendants", where some particles with small weights are discarded. In the third row, the particles are drifted according to the deterministic part of the system model. The random part of the system model is applied in the diffusion step (fourth line). Afterwards, the new measurement is used to weighten the particles. Now, they represent the posterior density (last line). Note that in lines two to four the particle weights are not shown as they are not important for these steps.

Repeat N times (indexed as i):

1. generate a uniformly distributed random number $r \in [0; 1]$
2. find the smallest j for which $c_{t-1}^{(j)} \geq r$
3. set $\mathbf{s}_t'^{(i)} = \mathbf{s}_{t-1}^{(j)}$

Figure 3.6: The Resampling Algorithm.

An efficient way to draw particles for resampling, is to use *cumulative weights* $c_{t-1}^{(i)}$:

$$c_{t-1}^{(0)} = 0 \tag{3.8}$$

$$c_{t-1}^{(i)} = c_{t-1}^{(i-1)} + \pi_{t-1}^{(i)} \quad (i = 1, \cdots, N) \tag{3.9}$$

The resampling algorithm is then as shown in figure 3.6 (cf. also figure 3.7). The search in step two can be sped up by binary subdivision. The set $\{\mathbf{s}_t'^{(i)}, i = 1, \ldots, N\}$ is the new resampled (and yet unweighted) sample set.

As some particles are duplicated several times while others are discarded, too much resampling results in a high concentration of the particles in the PDF's modes. Other parts of the PDF, that may become interesting in the following time steps might then not be covered by particles and thus may not be tracked. This problem is called the *particle depletion problem* which can be tackled by a systematic reduction of the number of resampling steps.

Before each resampling step, the *estimated number of effective particles* $\widehat{N}_{eff}$ is computed as follows (see [Liu96]):

$$\widehat{N}_{eff} = \frac{1}{\sum_{i=1}^{N} (\pi^{(i)})^2} \tag{3.10}$$

As shown by MacCormick in [Mac00], this number can be used as an index on the quality of the approximation. If $\widehat{N}_{eff}$ falls below a certain threshold, e.g. $N/2$, the sample set is resampled. Otherwise, the resampling step is skipped to avoid a collapse of the particles on the PDF's modes.

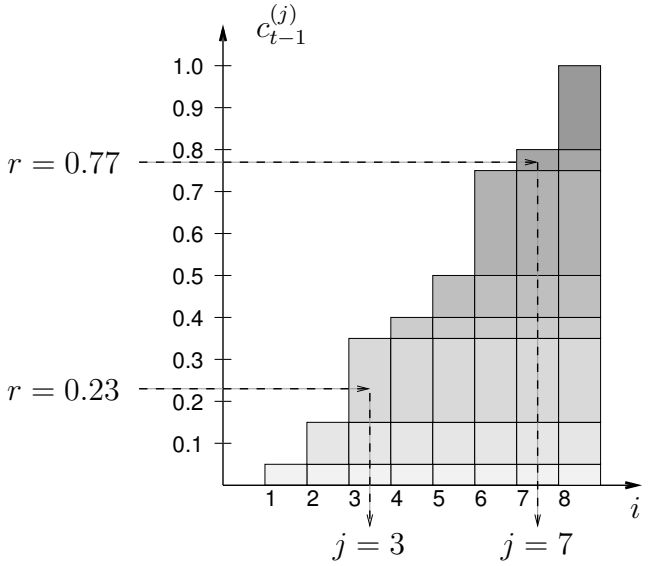
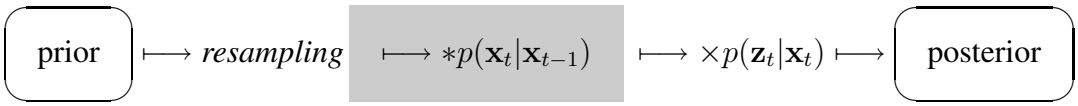


Figure 3.7: Cumulative resampling. The random variable r defines which particle j should be used as copy source. Note that particles with high weights (large boxes) have higher probability to be chosen. You can see this selection as a walk up the "cumulative-weight-stairs" until the value of r is passed.

3.4.2 Drift and Diffuse



This step is the application of the system model. The new states of the particles are predicted by sampling from

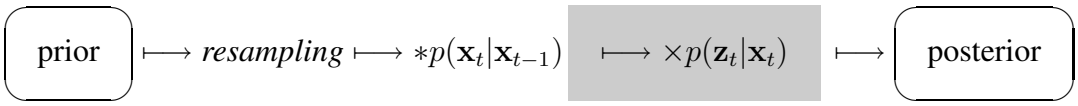
$$p(\mathbf{x}_t|\mathbf{x}_{t-1} = \mathbf{s}'^{(i)}_t).$$

This sampling is nothing else than the application of the system transition function $\mathbf{f}$ (equ. 3.1)

$$\mathbf{s}^{(i)}_t = \mathbf{f}(\mathbf{s}'^{(i)}_t, \mathbf{v}_t),$$

which can be split in two parts. Firstly, the particles undergo a *drift*, which is the deterministic part of the system model. Secondly, the *diffusion* moves the particles randomly with respect to the system noise $\mathbf{v}_t$, splitting now the identical copies that accrued in the resampling step.

3.4.3 Measure



The last step assigns new weights to the particles using the observation density $p(\mathbf{z}_t|\mathbf{x}_t)$ of the new measurement $\mathbf{z}_t$:

$$\pi^{(i)}_t = \frac{p(\mathbf{z}_t|\mathbf{x}_t = \mathbf{s}^{(i)}_t)}{\sum_{j=1}^{N} p(\mathbf{z}_t|\mathbf{x}_t = \mathbf{s}^{(j)}_t)}$$

The division by the sum of all weights assures that they sum up to one. After this normalization, the set

$$\mathbf{S}_t = \{(\mathbf{s}^{(i)}_t, \pi^{(i)}_t), i = 1, \ldots, N\}$$

represents the posterior and can be used to compute, e.g. the minimum mean square error estimate $\hat{\mathbf{x}}_t$ (cf. eq.3.7):

$$\hat{\mathbf{x}}_t = \sum_{n=1}^{N} \pi^{(n)}_t \mathbf{s}^{(n)}_t \tag{3.11}$$

For the next iteration $t + 1$, the sample set $\mathbf{S}_t$ is taken as prior.

Chapter 4

A Particle Filter for Cable Tracking

This chapter describes the application of particle filters for cable tracking and its implementation.

The first step for solving a tracking problem with particle filters is finding out appropriate parameters that describe the system. The set of these parameters form the *state model*, a vector for which the probability density function is to be estimated. Then, the changes of these parameters have to be inspected to define a tracking strategy. Based on collected data and assumptions on the physics of the scenario, a *movement model* has to be found, which describes the transition of the system state from one time step to the next. Lastly, the *observation model* defines how sensor measurements are used to evaluate the quality of predicted state hypotheses.

As a start, let us again have a look at the cable following scenario, outlining some constraints that can be identified.

4.1 Scenario

A submarine robot is to be used to follow a cable, using a camera, which points slightly forward. It is assumed that the cable is visible when the tracking algorithm starts. Search-

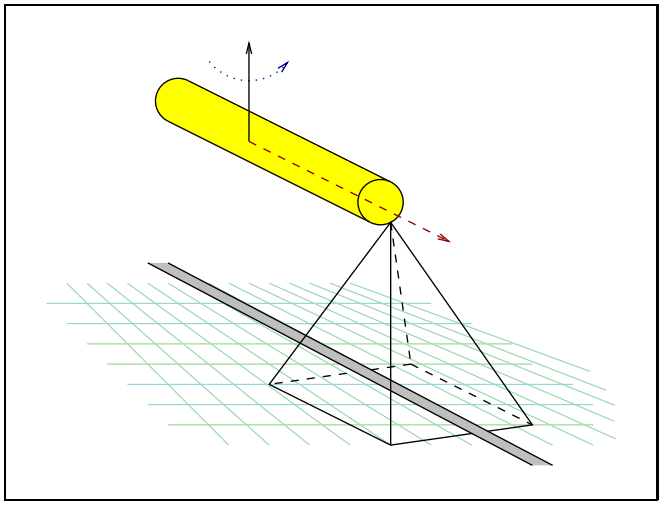


Figure 4.1: The submarine stays parallel to the ground and only performs longitudinal (red/dashed arrow) movements and rotations around its yaw axis (blue/dotted arrow).

ing for a cable (moving the robot around until the cable is visible in the camera image) is another problem that is not adressed here.

Another assumption is, that the robot keeps a constant distance to the seabed with the help of an altitude sensor. Therefore, the vehicle's movements are confined to a 2D plane parallel to the ground. The vehicle only changes its yaw angle and performs longitudinal translations (see figure 4.1). The curvature of the cable is very small so that the cable borders appear as (more or less) straight lines in the image.

The camera's view direction does not change with respect to the robot, i.e. camera and submarine are rigidly coupled.

For the cable following task, the controller of the vehicle, which has already been developed by the SRV group (see [AOO06a, AOO06b]) needs the position and the angle of the cable in the image. More precisely, it needs the parameters of the *cable centerline*. It is then able to steer the vehicle in such a way that the cable is vertically aligned in the center of the image. Figure 4.2 shows some artificial cable images with the corresponding control command that has to be sent to align the cable.

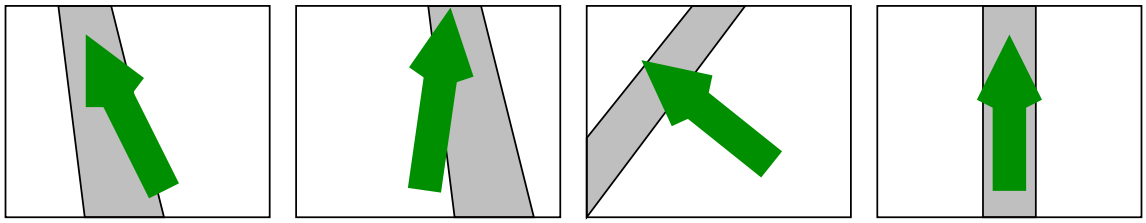


Figure 4.2: Cable images with corresponding control commands. The arrows show into which direction the vehicle has to be steered.

4.2 Decision on Cable Parameters – The State Model

Chapter 2 gives hints on the different possibilities to parameterize a cable in an image. These are

- cable borders (two lines),
- cable texture parameters, or
- line segments from borders of segmented regions.

The texture approach fails if the seabed has a similar texture to the cable and seems only efficient if implemented in hardware. Tracking could solve the problem that texture parameters change over time – because they are tracked. Nevertheless, the problem of computing the parameters of the cable's centerline based on (texture-)segmentation results, is not trivial.

For the grey level-gradient modulus based segmentation approach it is difficult to find the right parameters and the overall procedure is rather complex.

I therefore decided to use the cable borders as features that characterize a cable and can be tracked easily. The use of particle filters limits heavily the computational complexity, because not the whole image has to be processed but only the parts that are covered by the particles. I therefore assume that more complex border extracting filters can be used without time loss compared to the border extraction-Hough transform approach.

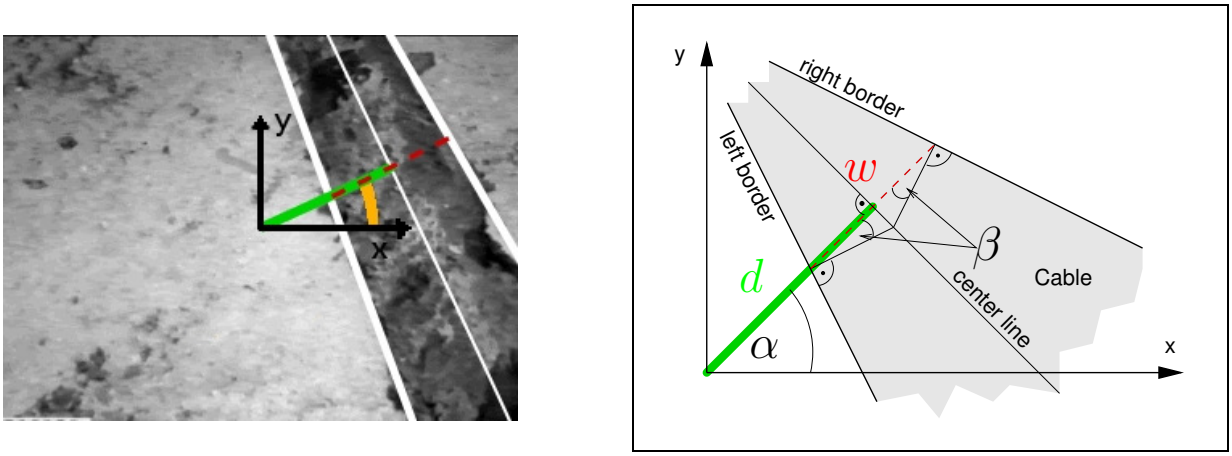


Figure 4.3: Parameters of the cable model.

Parameter	***Description***
d	Distance from image center to cable centerline
α	Angle of cable centerline
w	Cable width
β	Cable "skew", arises from perspective distortion if the camera image plane is not parallel to the cable

Table 4.1: Cable model parameters

Regarding the constraints mentioned in section 4.1, we can now define the appearance of the cable in the image as follows. The cable's borders form nearly parallel lines. The distance between these lines is named the *cable width* w. The centerline between these lines is the cable centerline. Using a coordinate system that has its origin in the image center, its x-axis heading right and its y-axis heading upwards, we can choose two parameters that identify the cable centerline: the *cable angle* α and the *cable distance* d. The angle between a cable border and the cable centerline due to the perspective effect is modeled by the variable β and is called the *cable skew*. Figure 4.3 illustrates all of these parameters which are also listed in table 4.1.

Hence, a sample of the particle filter holds the four-dimensional vector $(d, \alpha, w, \beta)^{\mathrm{T}}$, which represents a hypothetical state.

4.3 Set up the Prior – Initialization

As mentioned in section 3.2, it is assumed that the initial prior $p(\mathbf{x}_0|\mathbf{z}_0) = p(\mathbf{x}_0)$ is available. This distribution is used for sampling the initial particle states. In the cable tracking application, we could use either any available knowledge about the cable state, which might come from marking the cable position in the first image by hand or by any other cable *detection* algorithm, or assume that this distribution is uniform.

For the latter, it is helpful to define some limits that restrict the expansion of the distribution. Let $u_{\min}$ and $u_{\max}$ be the lower and upper bound for typical values of the state parameter u. A random sample $\mathbf{s}$ can then be generated as follows:

$$\mathbf{s} = \begin{pmatrix} r_1(d_{\max} - d_{\min}) + d_{\min} \\ r_2(\alpha_{\max} - \alpha_{\min}) + \alpha_{\min} \\ r_3(w_{\max} - w_{\min}) + w_{\min} \\ r_4(\beta_{\max} - \beta_{\min}) + \beta_{\min} \end{pmatrix} \quad , \tag{4.1}$$

where $r_i, \{i = 1, ..., 4\}$ are random numbers, uniformly distributed over the interval $[0; 1]$.

For both methods, using knowledge or random initialization, experiments are shown in chapter 5.

4.4 Drift and Diffuse – The Movement Model

The movements of the submarine robot result in relative movements of the cable in the camera image. The changes of the above listed parameters from one time step to the next have to be modeled by the movement model. The movement model can then be used in the prediction step (*drift* and *diffuse*) of the particle filter. Again, we can use the constraints from the scenario description to make some assumptions:

- Neither the cable "skew" β nor the cable width w change significantly over time due to the confinement of the submarine's movements to a 2D plane and the rigidly coupled camera.
- The cable angle α changes smoothly when the vehicle rotates around its yaw axis.

- The cable distance d changes when $\alpha \neq 0°$ and the vehicle is moving.

Note that, if the control of the robot works perfect, none of the parameters changes over time, because the robot always stays exactly above the cable, heading towards the cable's direction and keeping always the same distance to the cable.

Perfect control can – of course – never be achieved. Moreover, the input that has been used for testing the proposed tracking algorithm consisted in video sequences which were taken from manually controlled vehicles or AUVs that did not use a visual controlling method.

Examining the video sequences which form the data base of this work, we will probe if the above constraints are reasonable. The plots of figure 4.4 show how the parameters w and β behave in four different underwater cable tracking videos. The data were gathered by hand-labeling each frame of the sequences with the *GroundTruthEditor* (see appendix B). See appendix A for a complete list and description of all videos including screenshots.

It is easy to see that the parameters w and β change slowly. Sequence number two is an exception. The vehicle that carried the camera for this sequence made oscillating pitch moves, looking up and down alternately. This movement can be seen in the plots: every time the camera looks downwards, the cable width increases. The opposite happens when the camera looks up. These kind of movements are unwanted and therefore not modeled here.

Without any further information on camera or vehicle movement, we cannot determine how the parameters cable width and cable skew should be *drifted* in the corresponding filter step. Hence, the propagation of these parameters has to be modeled solely as *diffusion* which is chosen as the addition of Gaussian random noise:

$$\begin{aligned} w_t &= w_{t-1} + G(\sigma_w^2) \\ \beta_t &= \beta_{t-1} + G(\sigma_\beta^2), \end{aligned}$$

where $G(\sigma^2)$ is a function that returns a $\mathcal{N}(0, \sigma^2)$ distributed Gaussian random variable.

The variances σ_w^2 and σ_β^2 are the variances of the *cable width change* and the *cable skew change*, respectively. Their values can be directly obtained from the video sequences.

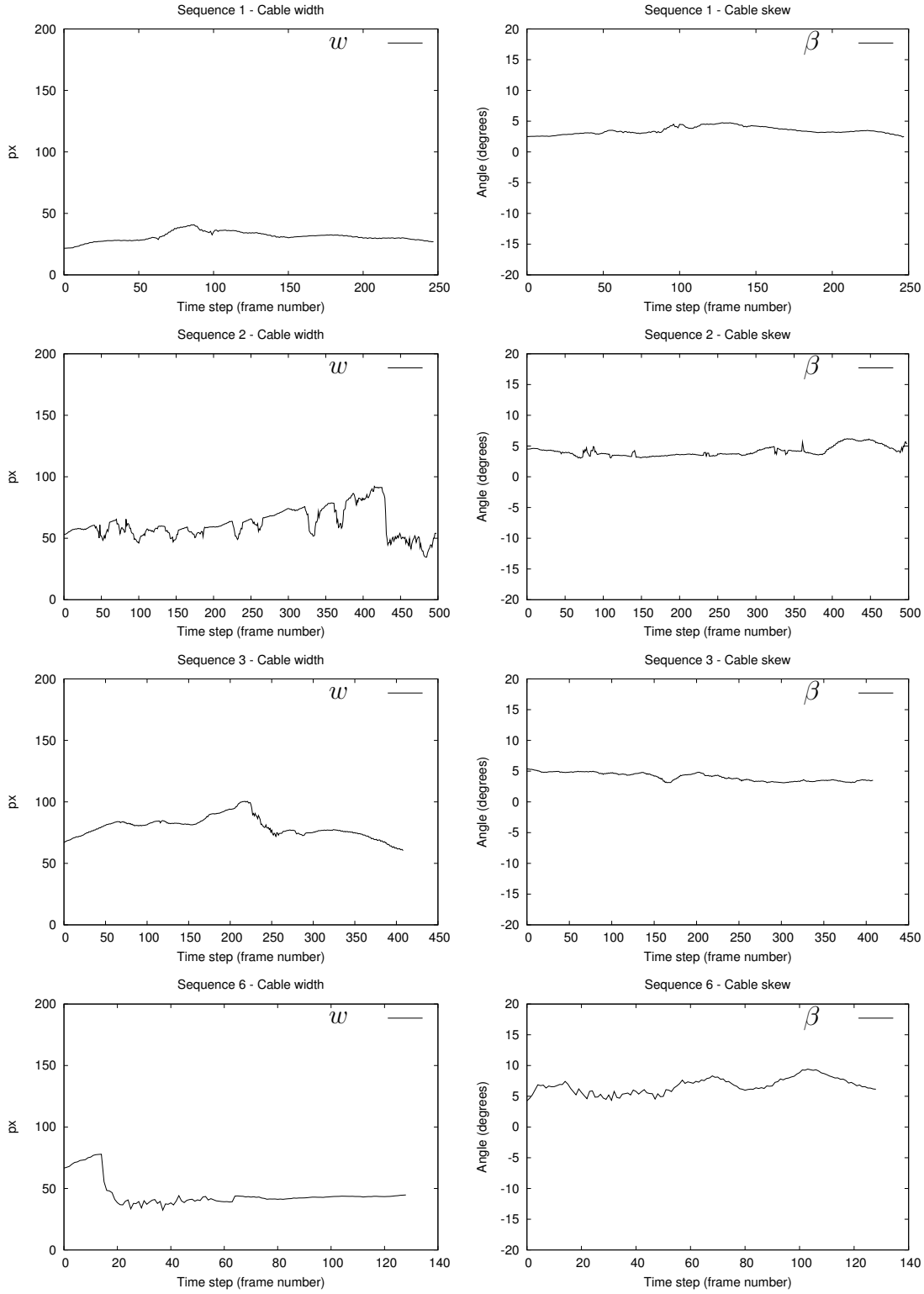


Figure 4.4: These plots show how the parameters *cable width* and *cable skew* change over time in different video sequences.

Sequence	σ_w^2 ***(in px^2)***	σ_β^2 ***(in deg^2)***
S1	0.26	0.008
S2	5.19	0.034
S3	0.51	0.004
S4	8.59	0.009
S5	0.21	0.021
S6	7.50	0.177

Table 4.2: Variances of cable skew and cable width changes.

Let T be the number of observations that are available (number of frames) in one sequence. The variance of the cable width change is then computed as

$$\sigma_w^2 = \frac{1}{T-1}\sum_{t=2}^{T}((w_t - w_{t-1}) - \mu_w)^2 \quad \text{with} \quad \mu_w = \frac{1}{T-1}\sum_{t=2}^{T}(w_t - w_{t-1}) \quad . \tag{4.2}$$

The cable skew change variance is computed analogously. Table 4.2 lists the two variances for the six available video sequences.

Let us now have a look at the two other parameters, cable distance d and cable angle α. These two parameters are the most interesting – as they are used by the controller – and, in addition, the most difficult to track – as they may change rapidly from one time step to the next.

The plots of figure 4.5 show how these two parameters change in three video sequences.

The plots point out, that these parameters vary in a different way as the others. They seem not to be random, which makes sense if we consider the following fact. If the cable is moving quickly in the image from one side to the other (i.e. the cable distance is changing) or rotating in one direction (i.e. the cable angle is changing) it is likely to observe a similar movement in the following frames. We could integrate this information by looking at n past frames, trying to determine the movements in the past and extrapolating these movements to make predictions for the future. I decided to model instant velocities instead, which is an indirect way of using past information.

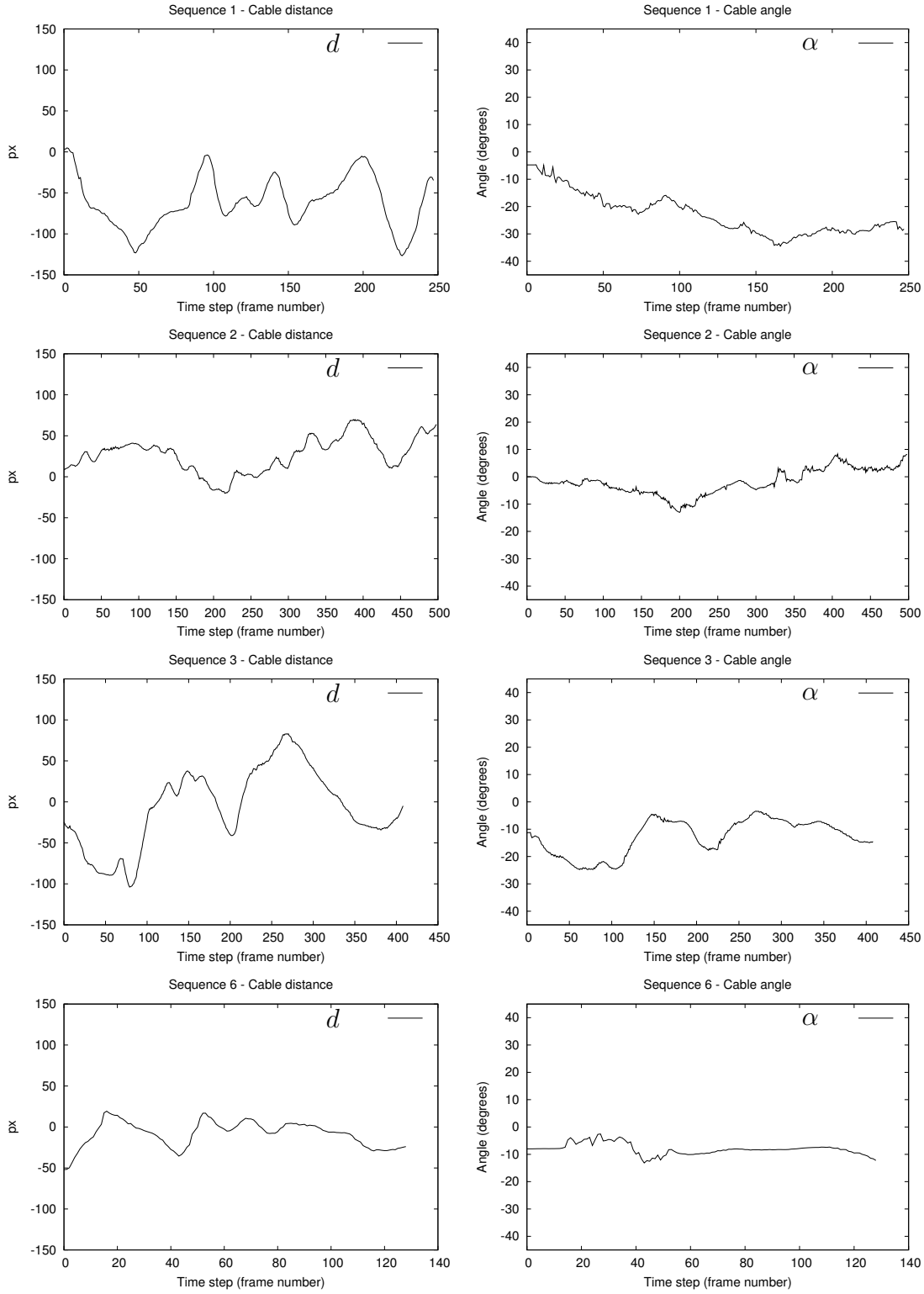


Figure 4.5: These plots show how the parameters *cable distance* and *cable angle* change over time in different video sequences.

Sequence	σ_d^2 ***(in px^2)***	σ_α^2 ***(in deg^2)***
S1	16.92	0.627
S2	2.07	0.286
S3	4.92	0.160
S4	4.93	0.486
S5	10.21	0.199
S6	10.31	0.599

Table 4.3: Variances of cable distance and cable angle changes.

Let v_d be the instant velocity of cable distance changes and v_α the instant velocity of cable angle changes. The *drift* of the parameters cable distance d and cable angle α can then be expressed as

$$d'_t = d_{t-1} + (v_d)_{t-1} \tag{4.3}$$
$$\alpha'_t = \alpha_{t-1} + (v_\alpha)_{t-1} \tag{4.4}$$

The time index for the velocities is necessary, as they change over time. To *track* the velocities, they have to be included as additional parameters into the system state. As a result, the state vector now has six dimensions:

$$\mathbf{x} = (d, \alpha, w, \beta, v_d, v_\alpha)^{\mathrm{T}} \quad .$$

Increasing the system state's dimensions is a critical step which here is justified by the ability of making more precise predictions. If the changes of the parameters d and α are modeled solely as noise, the noise's variance will have to be chosen very high (see table 4.3), which leads to a wider spread particle distribution with the risk of particle depletion. Experiments will show if the inclusion of the velocities into the system state is an advantage or not.

For the *diffusion* of the variables d and α, again, Gaussian random noise is used:

$$d_t = d'_t + G(\sigma_d^2) \tag{4.5}$$
$$\alpha_t = \alpha'_t + G(\sigma_\alpha^2) \tag{4.6}$$

As the instant velocities change over time as well, they are diffused by Gaussian random noise, too:

$$(v_d)_t = (v_d)_{t-1} + G(\sigma^2_{v_d}) \tag{4.7}$$
$$(v_\alpha)_t = (v_\alpha)_{t-1} + G(\sigma^2_{v_\alpha}) \tag{4.8}$$

Note that acceleration – the change of velocity – is modeled by this noise.

The values for the variances σ^2_d and $\sigma^2_{v_d}$ as well as for the variances σ^2_α and $\sigma^2_{v_\alpha}$ are difficult to obtain. It has to be found out, which part of the increment comes from the regular *drift* and which part from the random *diffuse*. The easiest way to determine these values seems to be through experiments.

The overall transition of an hypothetical state $\mathbf{s}^{(i)}$, that is stored in a particle, from time $t-1$ to t is then as follows:

$$\mathbf{s}_t^{(i)} = \mathbf{s}_{t-1}^{(i)} + \boldsymbol{a} + \boldsymbol{b} = \begin{pmatrix} d \\ \alpha \\ w \\ \beta \\ v_d \\ v_\alpha \end{pmatrix}^{(i)}_{t-1} + \begin{pmatrix} v_d \\ v_\alpha \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}^{(i)}_{t-1} + \begin{pmatrix} G(\sigma^2_d) \\ G(\sigma^2_\alpha) \\ G(\sigma^2_w) \\ G(\sigma^2_\beta) \\ G(\sigma^2_{v_d}) \\ G(\sigma^2_{v_\alpha}) \end{pmatrix} \tag{4.9}$$

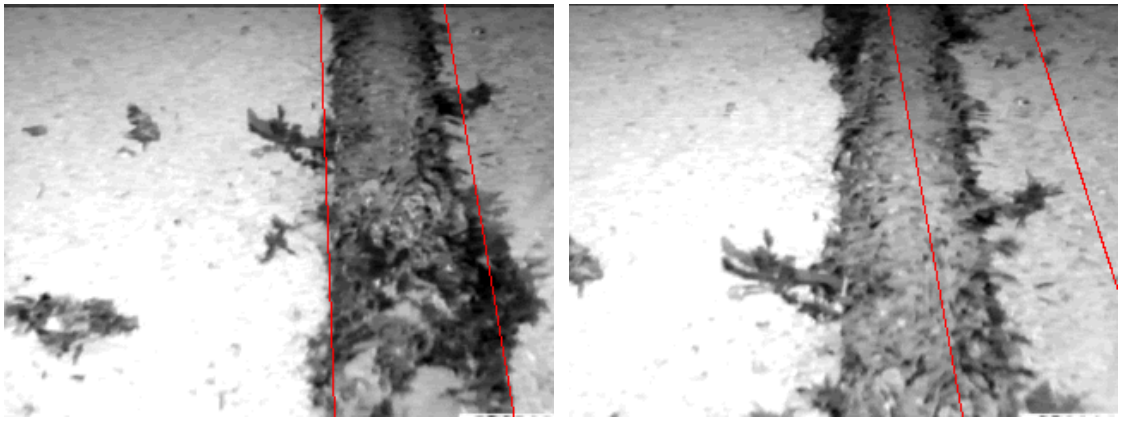
The addition of $\boldsymbol{a}$ is the *drift* and the addition of vector $\boldsymbol{b}$ is the *diffusion*.

The next step of the particle filter is the measurement step, which assigns weights to the predicted hypothetical states.

4.5 Measure – The Observation Model

In every cycle of the particle filter, each particle has to be weighted according to the probability of the most recent observation, given its state. The observations in visual cable tracking are the images that come from the robot's camera.

As already mentioned in the beginning of this chapter, I think that the grey level changes at the cable borders are the best features to identify the cable position. The idea of weighting



(a) (b)

Figure 4.6: Hypothetical cable states, projected on the observed images. (a) shows a good and (b) a bad guess.

a particle lies therefore in examining the grey level changes around the borders of the guessed cable position. Thus, in the first step, the state's parameters are used to project the (hypothetical) cable borders on the image (figure 4.6).

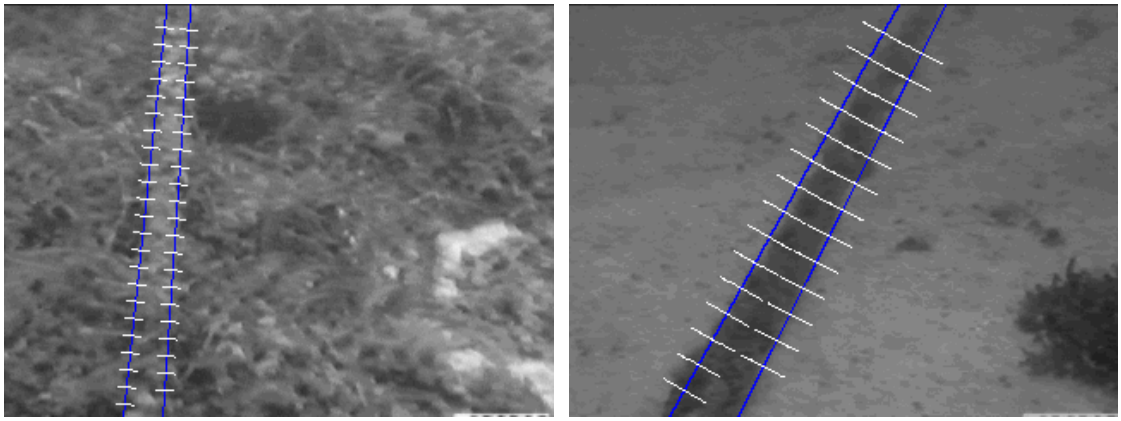
Then, a matching score for each border is computed as follows. Equally spaced along the border, a one-dimensional edge detecting filter is applied, which is oriented perpendicular to the border (figure 4.7). The size of the filter and the distance between the border sampling points are the two parameters of the observation model. We will call them *filter size* and *sample point distance*.

If the filter is applied on a real edge, its response is high, whereas points on falsely predicted borders produce low responses.

The used filter is a derivative of Gaussian filter (cf. figure 4.8). It is defined as

$$f(x) = -\frac{x}{\sigma^3\sqrt{2\pi}}\, e^{-\frac{1}{2\sigma^2}x^2} \quad , \tag{4.10}$$

with σ being the standard deviation. The derivative of Gaussian filter has a smoothing component which makes it detecting solely big intensity changes while skipping noise (cf. figure 4.9). If we have a look at a typical cable image, we can see that this is of great



(a) (b)

Figure 4.7: Hypothetical state with filter windows (short white lines). In (a), the filter size is 11 px and the sample point distance 10 px; in (b), the filter size is 29 px and the sample point distance 17 px.

importance. Marine growth, stones and sand cause many intensity changes we are not interested in, while the cable mostly forms a more "general" intensity change.

To calculate the border matching score, the filter responses from a single border are summed up. As the filter is direction dependent, this matching score can be positive or negative. A high positive matching score indicates a good guess for a border that is dark inside the cable (towards the cable centerline) and bright outside. A negative matching score indicates the opposite. As the cable may be either dark on bright ground (cf. figure 4.7(b)) or bright on dark ground (cf. figure 4.7(a)), the matching score of the two cable borders must have opposite signs.

Let $c_l^{(i)}$ and $c_r^{(i)}$ be the border counts for the two borders that were computed using the parameters from particle i. The (not yet normalized) weight $\pi^{(i)\prime}$ for that particle is then computed as

$$\pi^{(i)\prime} = \begin{cases} c_l^{(i)} - c_r^{(i)}, & \text{if } c_l^{(i)} > 0 \wedge c_r^{(i)} < 0 \\ c_r^{(i)} - c_l^{(i)}, & \text{if } c_r^{(i)} > 0 \wedge c_l^{(i)} < 0 \\ \varepsilon & \text{otherwise.} \end{cases} \tag{4.11}$$

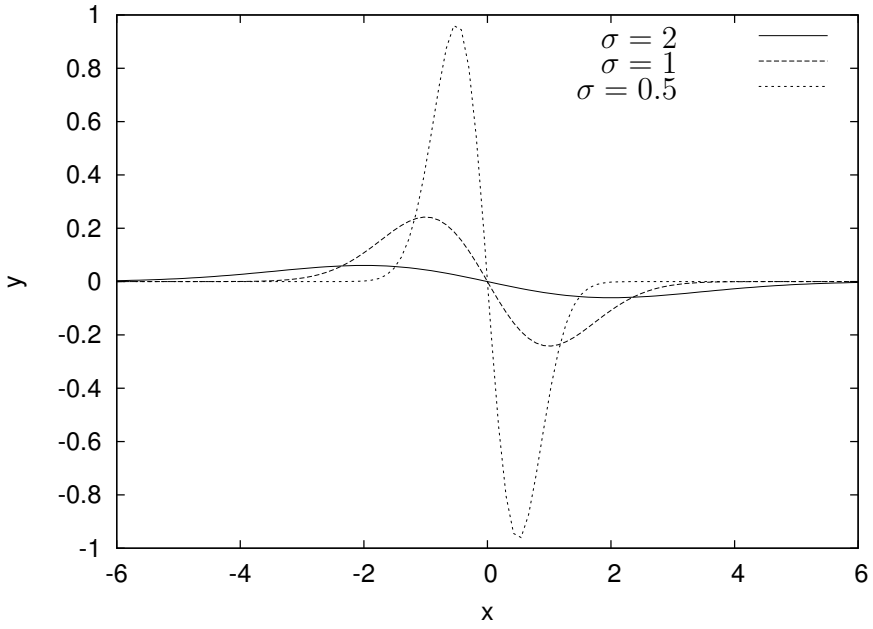


Figure 4.8: Derivative of Gaussian filter for different values of σ.

The bigger the filter size is chosen and the smaller the gaps between the points where the filter is applied, the higher is the number of pixels that have to be accessed to weight a particle. As the approach is planned to work in real-time, this number has to be kept small.

In the last step, the particle's weights are normalized to guarantee that they sum up to one:

$$\pi^{(i)} = \frac{\pi^{(i)\prime}}{\sum_{j=1}^{N} \pi^{(j)\prime}} \tag{4.12}$$

4.6 The Algorithm

The entire cable tracking algorithm consists in the straightforward application of the above described models. In every iteration the input of the algorithm is the particle set of the previous iteration together with a new observation. The output is the resampled, moved, and newly weighted particle set, from which the current cable state can be estimated.

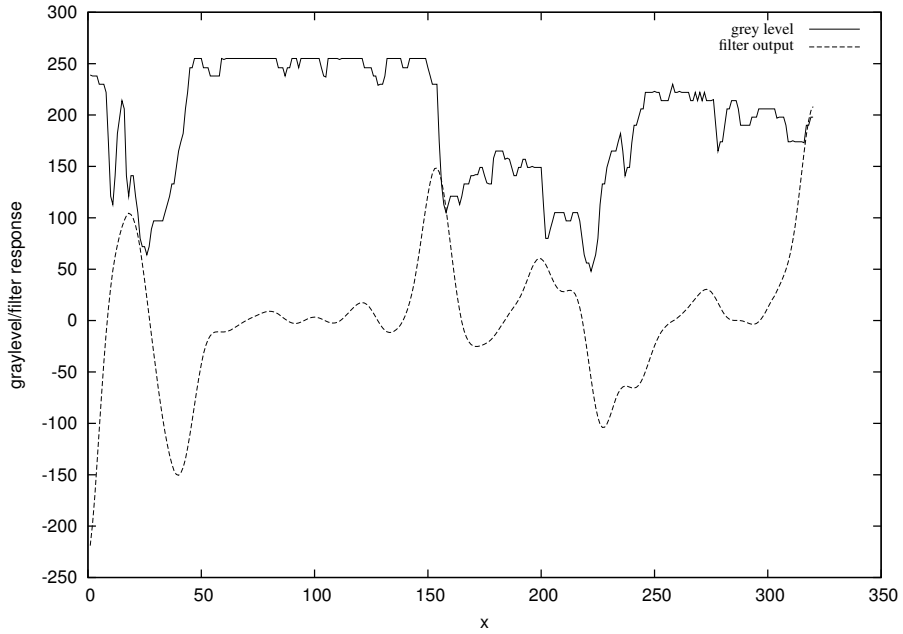


Figure 4.9: Example for a derivative of Gaussian filtered signal. The solid line corresponds to the grey levels of a pixel row of a cable image. The dashed line shows the signal after the application of a derivative of Gaussian filter of size 35 px. Note that big intensity changes (e.g. around $x = 40$ or $x = 160$) give a high response, whereas small changes are filtered out.

Figure 4.10 shows the algorithm, corresponding to the condensation algorithm of figure 3.5 (cf. figure 6 of [IB98]). In figure 4.11 the single steps are visualized.

4.7 Software Architecture

The developed cable tracking particle filter application is divided into three modules. The first module is the particle filter library, which implements all application-independent parts of a particle filter. The second module is the cable tracking library. This library implements the state model, movement model, and observation model for the cable tracking task as described in the previous sections, using the particle filter library's superclasses. The third module is a user interface, written with the help of Trolltech's Qt framework[1], version 4.2. The interface provides easy interaction with the cable tracking library, loading of input data and process monitoring.

For implementation details and source code, please refer to the Doxygen documentation on the enclosed CD (`/program/DoxygenDocumentation/html/index.html`).

4.7.1 Particle Filter Library

The particle filter library contains five classes:

1. `State` is the base class for the system state that has to be tracked. It only contains one abstract function `randomize()` that has to be implemented in subclasses, defining how a state can be initialized randomly (cf. section 4.3).

2. `Particle` is a class that holds a subclass of `State` together with its importance weight. The `ParticleFilter` class holds a set of `Particles` that approximate the tracking probability distribution function.

3. `ParticleFilter` implements the basis operations of the condensation algorithm (cf. figure 3.5) that are state independent.

[1] See `http://trolltech.com/products/qt` for description and download.

Let $\mathbf{S}_t = \{(\mathbf{s}_t^{(i)}, \pi_t^{(i)}), i = 1, \ldots, N\}$ be the set of N weighted samples (particles) at time t.

1. *Initialization:* to create the initial particle set

$$\mathbf{S}_0 = \{(\mathbf{s}_0^{(i)}, \pi_0^{(i)}), i = 1, \ldots, N\}$$

   either use ground truth knowledge or randomize the particles over an adequate space with equal weights $\pi_0^{(i)} = 1/N$. Now the particle set approximates the initial prior $p(\mathbf{x}_0)$.

2. *Iterate:* do the following steps to let the particle set track the state distribution for time t

   (a) *Resampling:* if $\widehat{N}_{eff} > N/2$, resample N new particles from $\mathbf{S}_{t-1}$ to create the new (unweighted) sample set

$$\mathbf{S}'_{t-1} = \{\mathbf{s}_{t-1}^{(i)\prime}, i = 1, \ldots, N\}$$

   using the resampling algorithm from figure 3.6.

   (b) *Drift and Diffuse:* update the particle states according to the cable movement (equation 4.9) to form the (unweighted) sample set $\mathbf{S}'_t$.

   (c) *Measure:* use the currently observed camera image (measurement $\mathbf{z}_t$) to update the observation model and use this model to calculate the new particle weights

$$\pi_t^{(i)} = p(\mathbf{z}_t | \mathbf{x}_t = \mathbf{s}_t^{(i)}) \quad .$$

   Now the particle set $\mathbf{S}_t = \{(\mathbf{s}_t^{(i)}, \pi_t^{(i)}), i = 1, \ldots, N\}$ approximates the posterior $p(\mathbf{x}_t | \mathbf{z}_{1:t})$.

   (d) *Estimate:* use $\mathbf{S}_t$ to estimate the cable state e.g. by computing the minimum mean square error estimate

$$\hat{\mathbf{x}}_t = \sum_{i=1}^{N} \pi_t^{(i)} \mathbf{s}_t^{(i)}$$

   This estimation can now be used as input of the AUV's controller.

Figure 4.10: Cable Tracking Algorithm

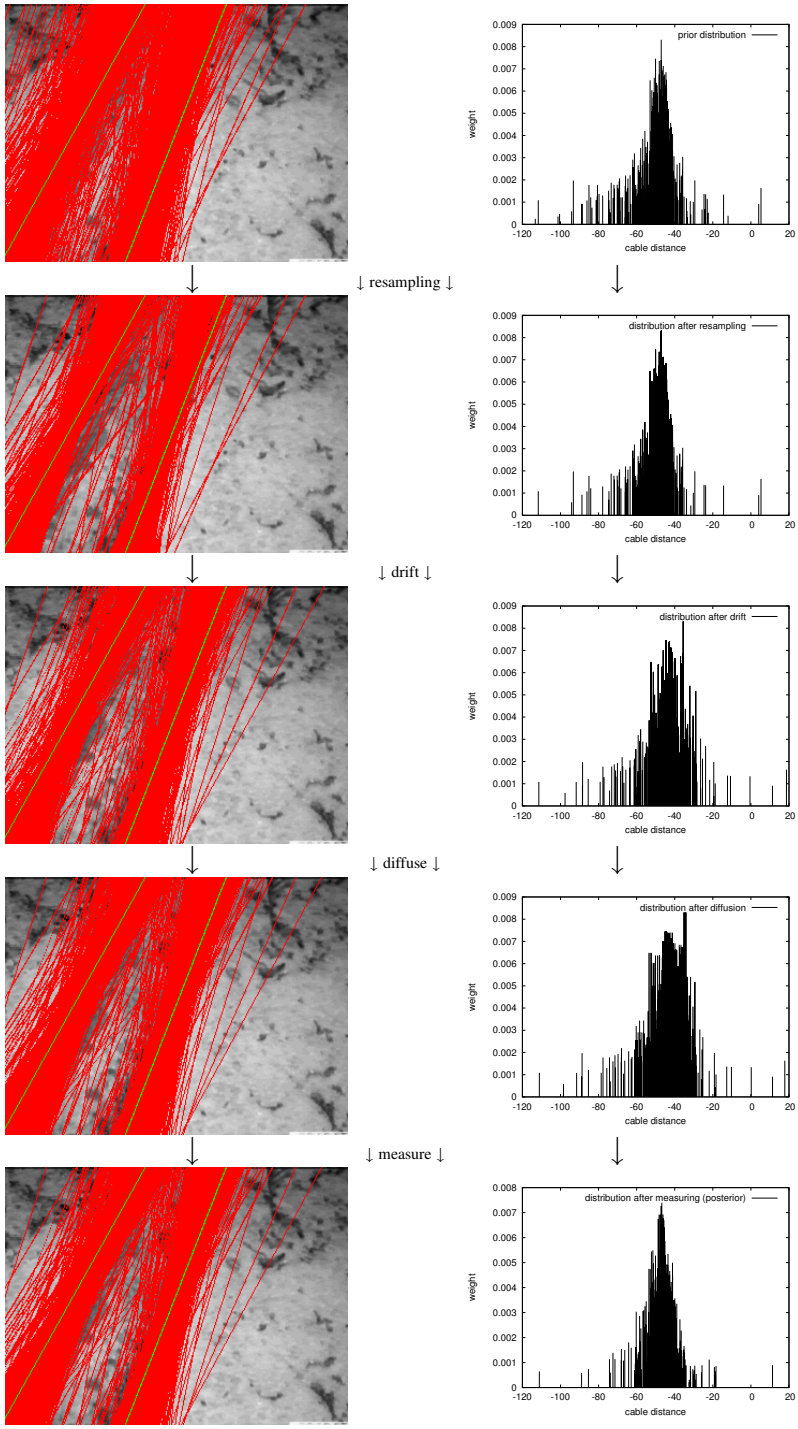


Figure 4.11: Steps of the particle filter algorithm. The left side shows the hypothetical cable poses projected on the current camera image. The green lines represent the ground truth state. The right side shows the distribution of the particles in the dimension cable distance. Each line represents a particle state. The length of the lines corresponds to the particle's weights.

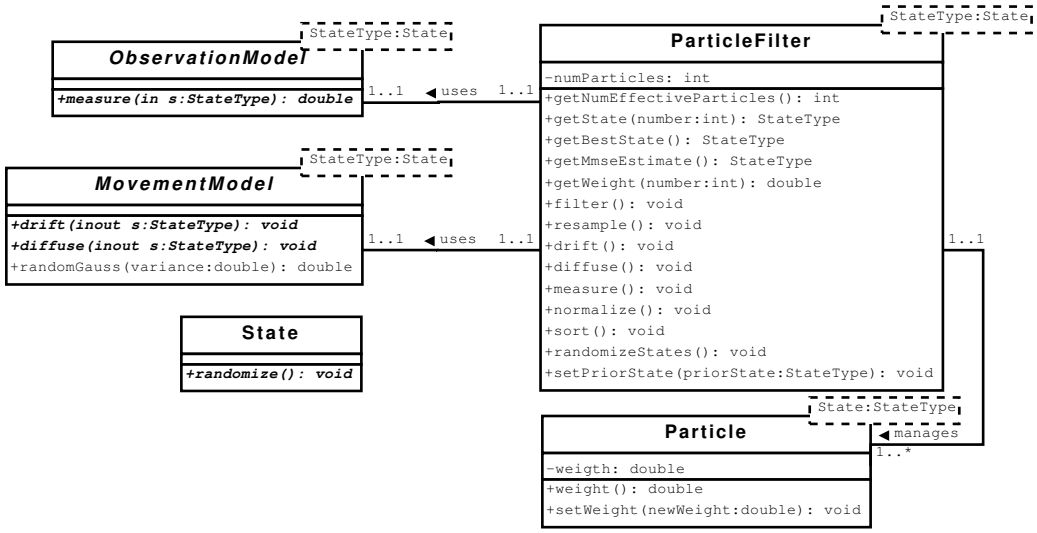


Figure 4.12: Class Diagram of the Particle Filter Library.

4. `MovementModel` is the class that is used in `ParticleFilter` to perform the drift and diffuse steps of the filter. It has to be derived to define how a state has to be modified in these steps. A function that generates Gaussian distributed random numbers is also implemented here, as it is used by most movement models.

5. `ObservationModel` is used in `ParticleFilter` to make the measurement step and has to be derived, too, to fit the application.

The relationships between these classes is depicted in figure 4.12.

To provide a state independent framework, `Particle`, `ParticleFilter`, `Movement-Model`, and `ObservationModel` are implemented as template classes, where a subclass of `State` can be chosen as template parameter.

4.7.2 Cable Tracking Library

The cable tracking library specializes the particle filter library to fit the cable tracking task. It contains six classes:

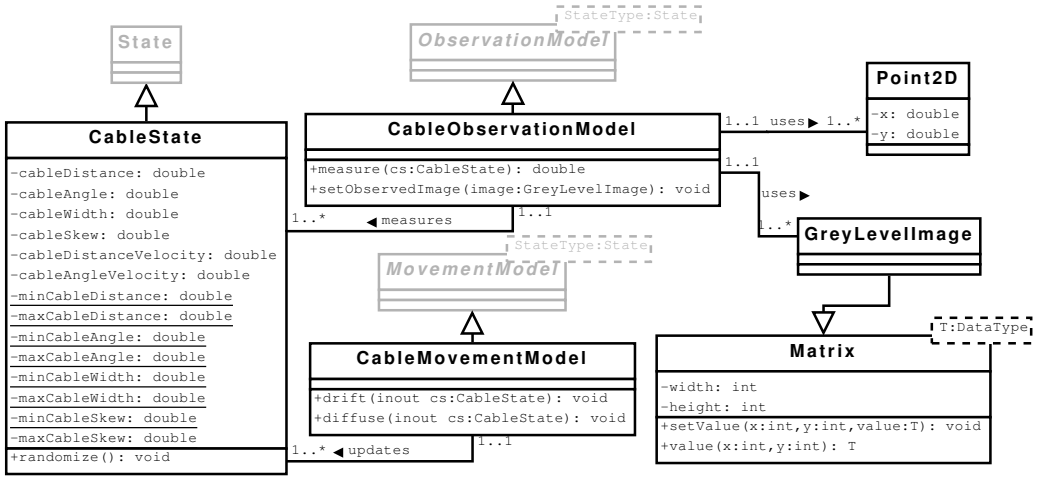


Figure 4.13: Class diagram of the cable tracking library. Note that not all methods are shown for the sake of clarity. The grey classes come from the particle filter library.

1. `CableState` holds the state parameters d, α, w, β, v_d, and v_α. In addition, realistic limits for these parameters are stored as static members for random initialization.

2. `CableMovementModel` implements the movement model from section 4.4. The methods `drift()` and `diffuse()` modify the passed `CableState` according to equation 4.9.

3. `CableObservationModel` implements the observation model from section 4.5. A `GreyLevelImage` has to be passed via the method `setObservedImage()` before a weight for a state can be computed via `measure()`.

4. `GreyLevelImage` handles grey level images as used by `CableObservationModel`.

5. `Matrix` is the base class for all types of matrices (superclass of `GreyLevelImage`).

6. `Point2D` is a class to store and manipulate 2D points. It is used in `CableObservationModel` to compute and store the coordinates of the pixels where the edge detector has to be applied.

Figure 4.13 shows the class diagram for the library.

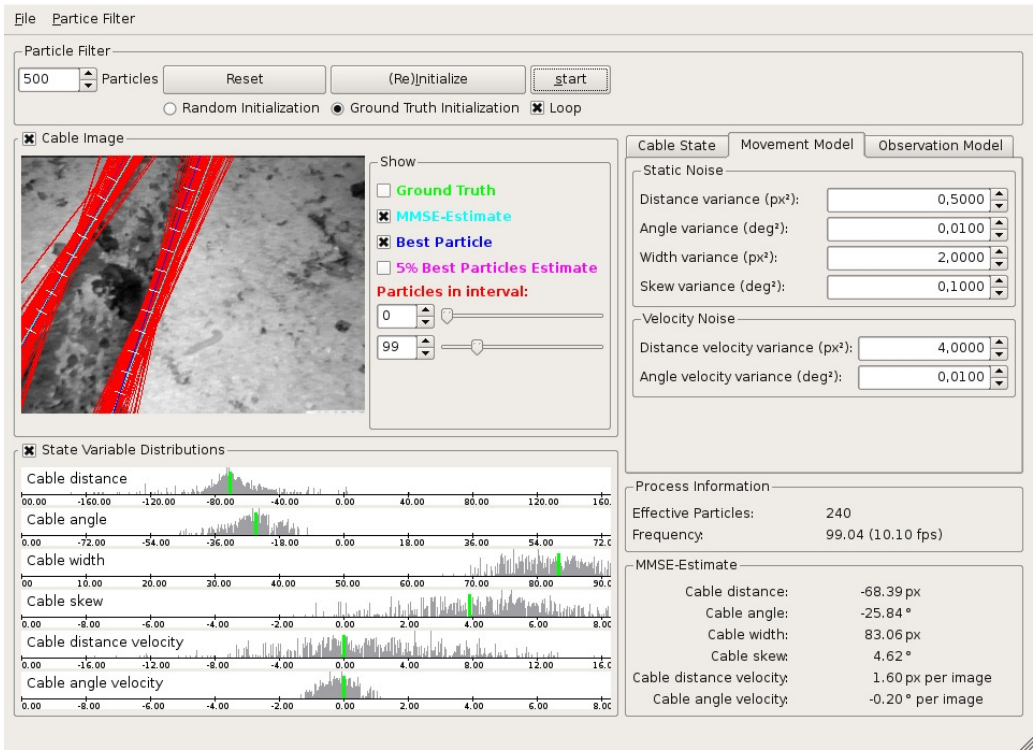


Figure 4.14: Screenshot of the User Interface.

4.7.3 Cable Tracking GUI

For process monitoring and easy parameter manipulation I designed a user interface that can interact with the cable tracking library. Figure 4.14 shows a screenshot of the program.

The tabs on the right side hold three widgets that communicate with the classes `CableState`, `CableMovementModel`, and `CableObservationModel`. They provide direct view and manipulation of the class parameters via spin boxes. The left and the lower part of the window show process information. The image on the left side displays the observation that is currently used to weight the particles. The coloured lines painted on top of the image represent hypothetical cable positions and current state estimations.

The diagrams in the lower left part show the momentary particle distribution. As the cable state has six dimensions each dimension is showed in a separate diagram. For each particle

state, the values of its parameters are drawn as vertical lines, taken the state's probability as line length.

The lower right side shows the numerical values of the minimum mean square estimate, together with process speed information.

To run the filter on an image sequence, an *image sequence file* can be opened via the window menu (*File→Open image sequence file...*). Image sequence files have the following format:

```
# CableDistance CableAngle CableWidth CableSkew # Filename
3.5 -0.082 21.5 0.043 # /home/stephan/data/S1_000.DIB
5.2 -0.078 24.5 0.052 # /home/stephan/data/S1_001.DIB
5.0 -0.078 24.0 0.052 # /home/stephan/data/S1_002.DIB
4.4 -0.078 22.5 0.034 # /home/stephan/data/S1_003.DIB
3.2 -0.069 27.0 0.052 # /home/stephan/data/S1_004.DIB
...
```

The numbers in the first four columns are the "ground truth" for the cable position in an image. The filename of the corresponding image must be given in the last column, separated from the numbers by a hash. The tool *GroundTruthEditor* (see appendix B) can be used to create these files. Cable angle and cable skew are given in radiants, cable distance and cable width are given in pixels. The ground truth is used for knowledge-based initialization and for visual comparison to computation results.

The program can also be run on a set of single images (*File→Open image sequence...*). If done so, the ground truth initialization does not work, as no ground truth is available.

Clicking the start button runs the loop of setting an image as observation, running the filter and displaying the result.

Chapter 5

Experiments

The goal of this work is to design and implement a cable tracking algorithm that is suitable for the control of a cable following AUV. The best way of testing whether this aim is achieved or not is putting the system on an underwater robot and let it follow a cable. Unfortunately the robot RAO-II of the SRV research group is currently still under construction. Therefore I have to test the algorithm on the previously mentioned underwater video sequences. The sequences show typical cable following situations over a high variety of appearances of the seabed and the cable. Figure 5.1 shows one frame of every test sequence.

In the first section of this chapter, I discuss what can be tested using the video sequences and how the results can be evaluated. The subsequent section presents the tests performed and their results. In the last section, the results are evaluated.

5.1 How to Test and Evaluate

From a controlling perspective, the most important aspect of the cable tracking algorithm is a *fast* and *accurate* estimation of the position of the cable centerline. Using the video sequences as input, the process frequency can easily be measured. In teleoperated vehicles, an operator views the camera images and reacts to them by steering the vehicle towards

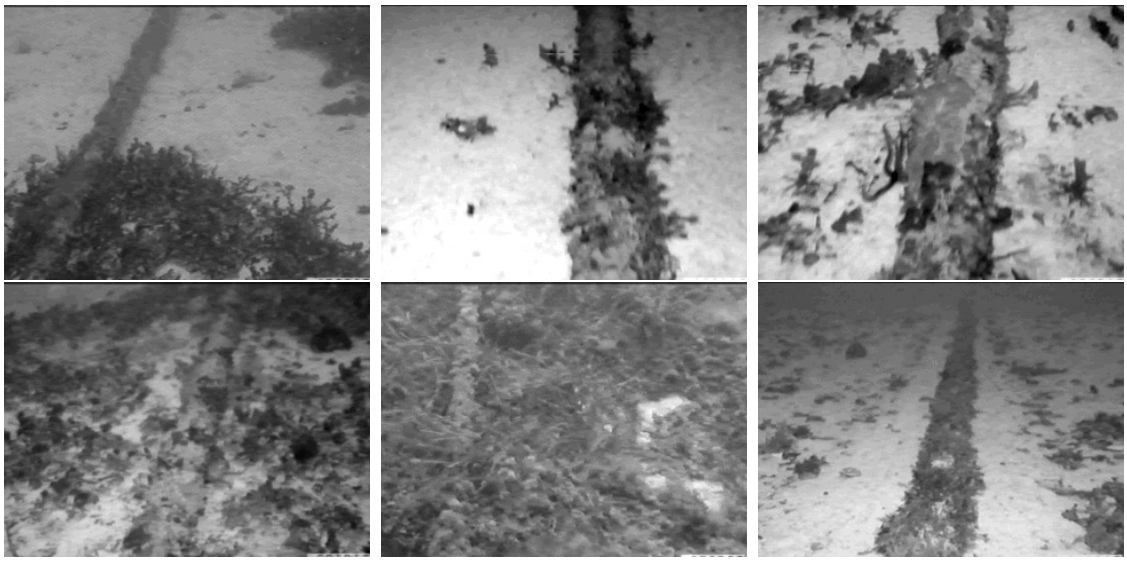


Figure 5.1: Different appearances of cable and seabed in the test sequences.

the cable's direction. Therefore I assume that an algorithm process frequency which lies around the video rate of the used camera (25-30 Hz) is sufficient for control. However, a higher process frequency would allow the robot to increase its velocity, if a faster camera was used.

The only way to measure the accuracy without a working robot is using ground truth-labeled video sequences. The comparison to ground truth enables setting up error metrics and visual comparison. Appendix B explains the tool that was used to obtain this ground truth.

As the sequences were recorded without the usage of a visual controlling method, the vehicle's movements are rather crude and disadvantageous for tracking. If the tracking works good, its results can be used to control the vehicle precisely which in turn is benificial for tracking. Hence, tracking and control highly depend on each other and finally must be tested together. However, when no visual controlling method was used to capture the video sequences, I suppose that if the tracking works good on these sequences, it should even work better if its results are used for control.

From a tracking perspective, it is of interest how and why errors occur. If we control the particle distribution and compare it to ground truth, we can make assumptions on the quality of motion and measurement model. The questions to answer here are the following:

- *Which measurement model parameters have to be chosen to achieve a robust tracking on noisy images and for different appearances of the cable?*
- *Which motion model parameters have to be chosen to keep the particle distribution near the real state?*

These questions can be answered by testing different sets of parameter values on all available image sequences.

5.2 Tests and Results

5.2.1 Test System

All tests have been run on an Intel® Pentium® M processor with 1.50 GHz. The operating system was Ubuntu Linux 7.10, kernel version 2.6.22-14-generic.

The number of particles used was 500. In some image sequences, the cable is easier to track what enables the reduction of the number of particles. However, 300 particles seem to be the minimum number for an accurate approximation of the cable state's PDF.

5.2.2 Interactive Determination of Parameters

The Cable Tracking GUI is a very convenient testing environment as parameters can easily be changed and the result of such a change becomes immediately visible.

By interactive experiments with all video sequences I found parameter values that work well in most situations. Having a close look on the particle distribution window (cf. figure 4.14), it is easy to see if the ground truth state is sufficiently covered by particles. To determine the values for cable distance and cable angle velocities, I slowly increased

Parameter	Value
σ_d^2 in px^2	0.5
σ_α^2 in $(^\circ)^2$	0.01
σ_w^2 in px^2	4.0
σ_β^2 in $(^\circ)^2$	0.1
$\sigma_{v_d}^2$ in px^2	8.0
$\sigma_{v_\alpha}^2$ in $(^\circ)^2$	0.005
filter size in px	11 ($\sigma \approx 1.6$)
point sample distance in px	20

Table 5.1: Experimentally determined parameters. The first six are parameters from the movement model, the last two are observation model parameters.

the corresponding variances until the particles were moved fast enough to follow the correct state. The other variances can be determined by surveying the particle distribution compared to ground truth. If the distribution is too narrow to follow the true state, the corresponding variance has to be increased.

Table 5.1 lists the parameter values that were determined interactively.

5.2.3 Image Sequence Tests

To test the accuracy of the tracking results, let us have a look at the image sequences in detail.

The following plots show ground truth and estimated states for each sequence. Each single plot shows the behaviour of one state parameter. Cable distance velocity and cable angle velocity are not shown, as ground truth for these parameters is not available. The settings for the tests are chosen as listed in table 5.1. The ground truth is always shown as solid line in the plots and the state estimations as dashed line. Appendix A shows the projections of the estimated cable states on the images.

For identification and comments on the tests, please refer to the figure's captions.

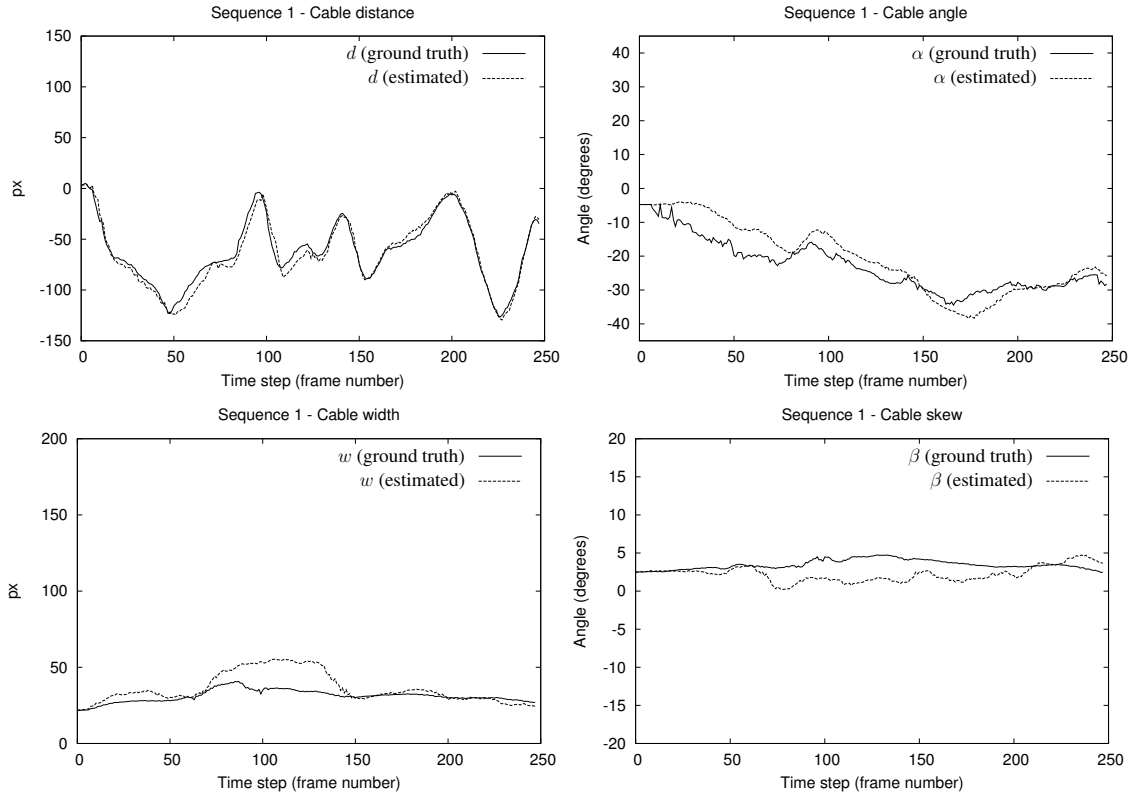


Figure 5.2: Sequence S1: ground truth and estimated parameters. In the first 120 frames of this sequence, the cable is surrounded by very much clutter (stones and algae) which makes the measurement difficult. Additionally, the camera makes wavering tilt and pitch moves which results in very fast changes of the parameter cable distance d. The cable state estimation is temporary bad but recovers the correct state after a couple of frames.

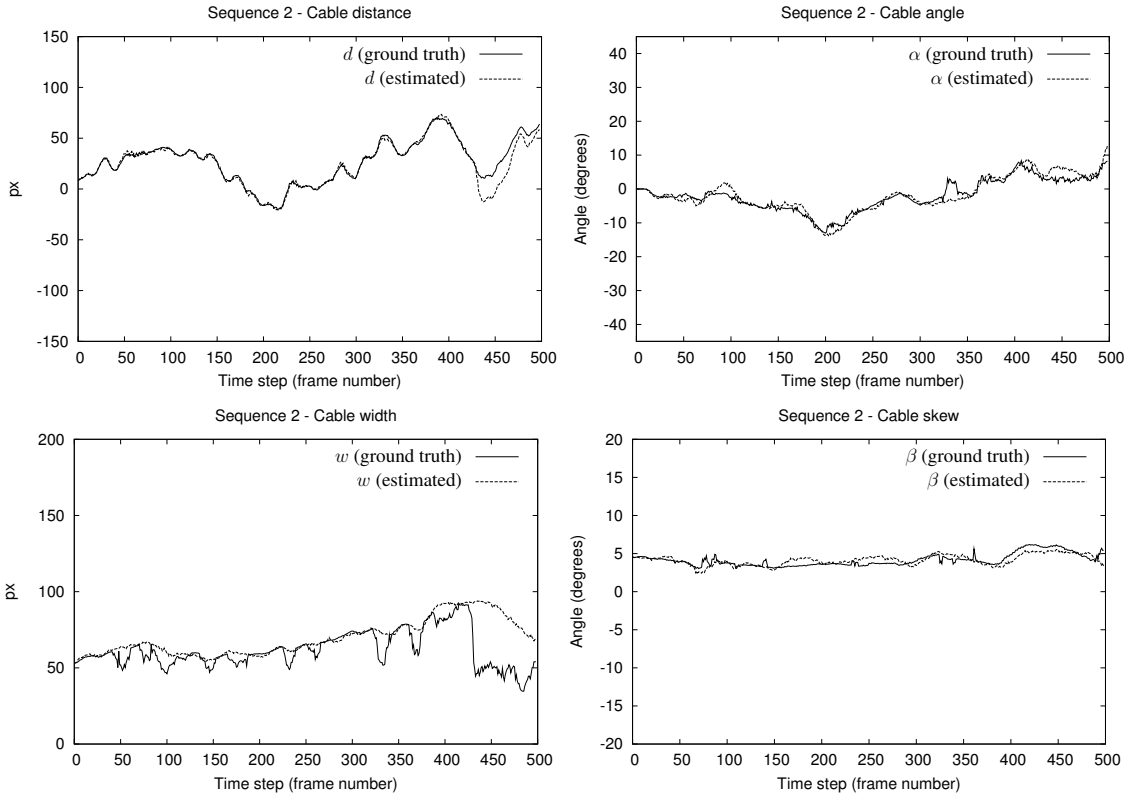


Figure 5.3: Sequence S2: ground truth and estimated parameters. The submarine that captured sequence S2 makes a lot of successive pitch moves and goes up and down which results in rapid changes of the cable width w. Even though the chosen movement model is not capable of compensating such movements, the other parameter estimations are very accurate. A rapid movement around frame 430 reduces the cable width abrubtly. The particles do not follow this change which affects the cable distance estimation (cf. figure A.6).

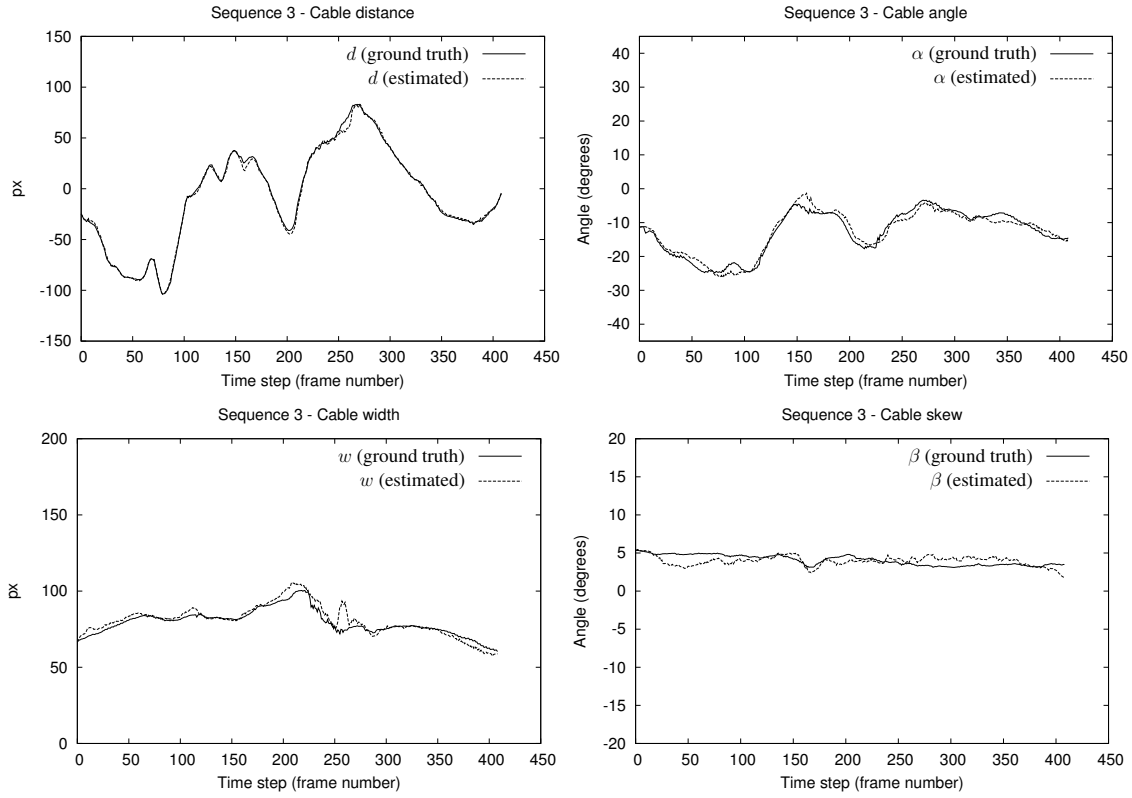


Figure 5.4: Sequence S3: ground truth and estimated parameters. Note the high accurracy of the estimated cable distance and cable angle for this sequence.

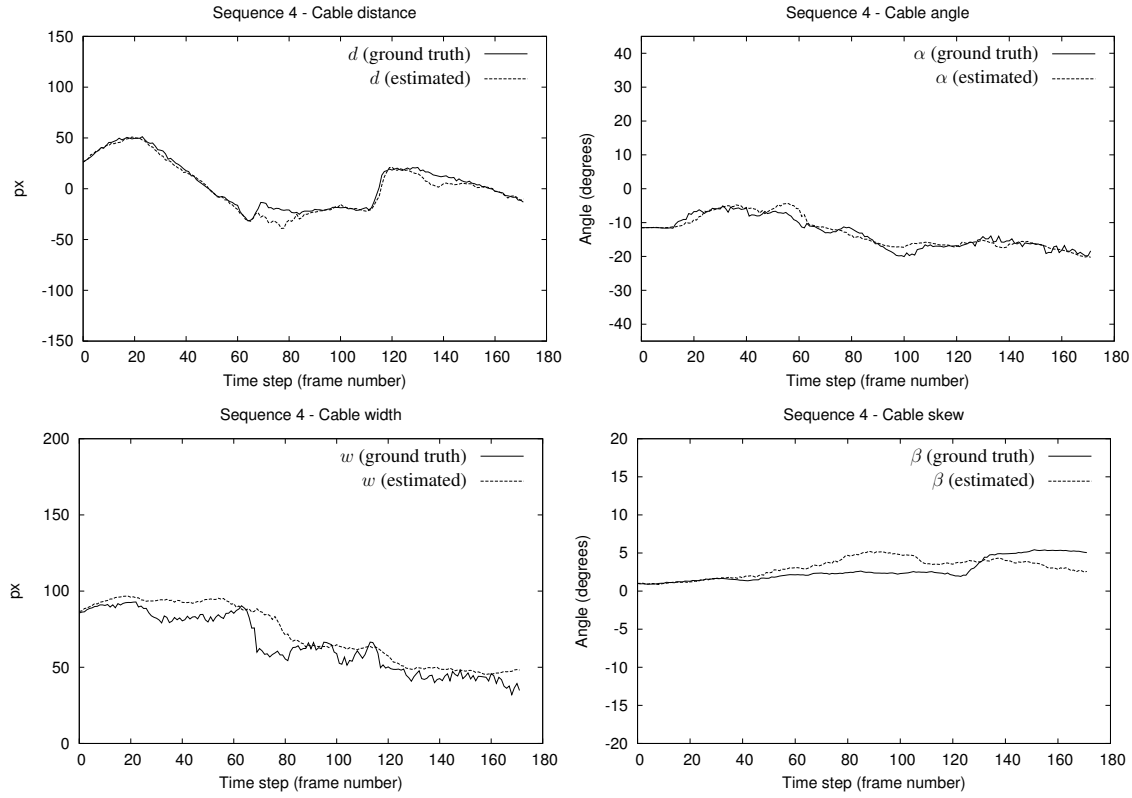


Figure 5.5: Sequence S4: ground truth and estimated parameters. The camera makes pitch moves around frame 70 and 125, therefore the cable width is not tracked correctly.

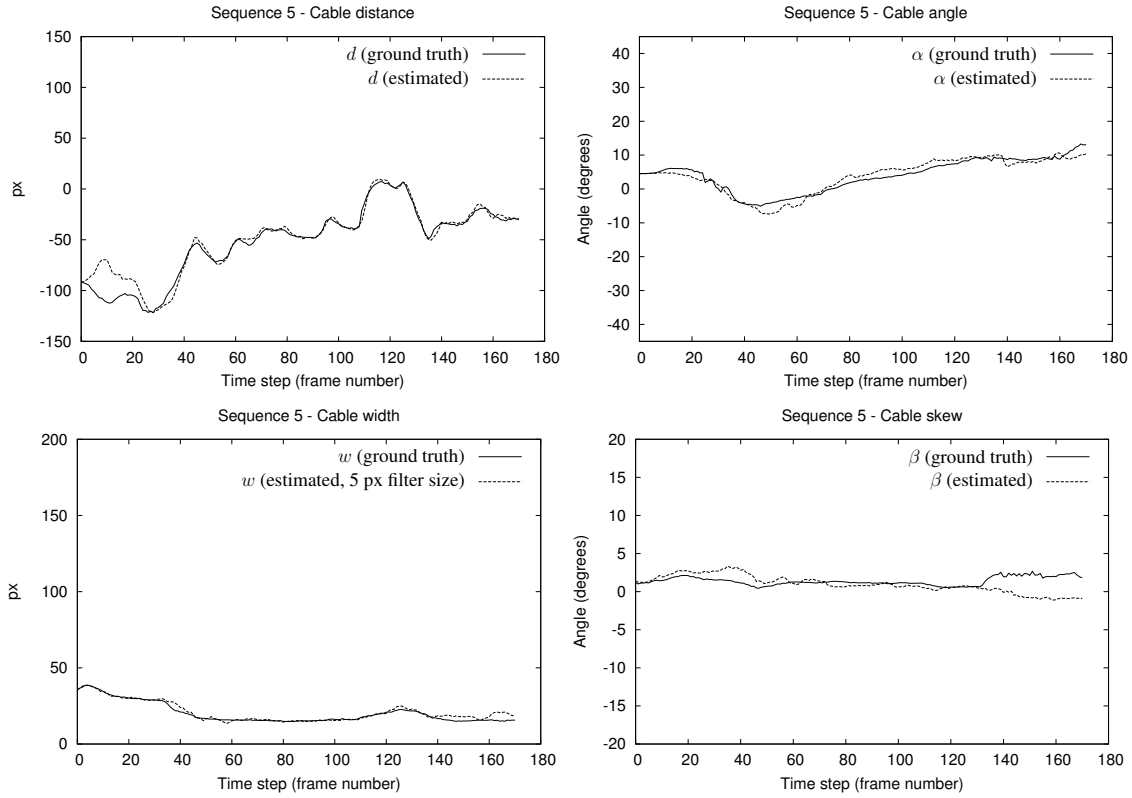


Figure 5.6: Sequence S5: ground truth and estimated parameters. In this sequence, cable and seabed appearances are totally different compared to the other sequences. Nevertheless, tracking works good.

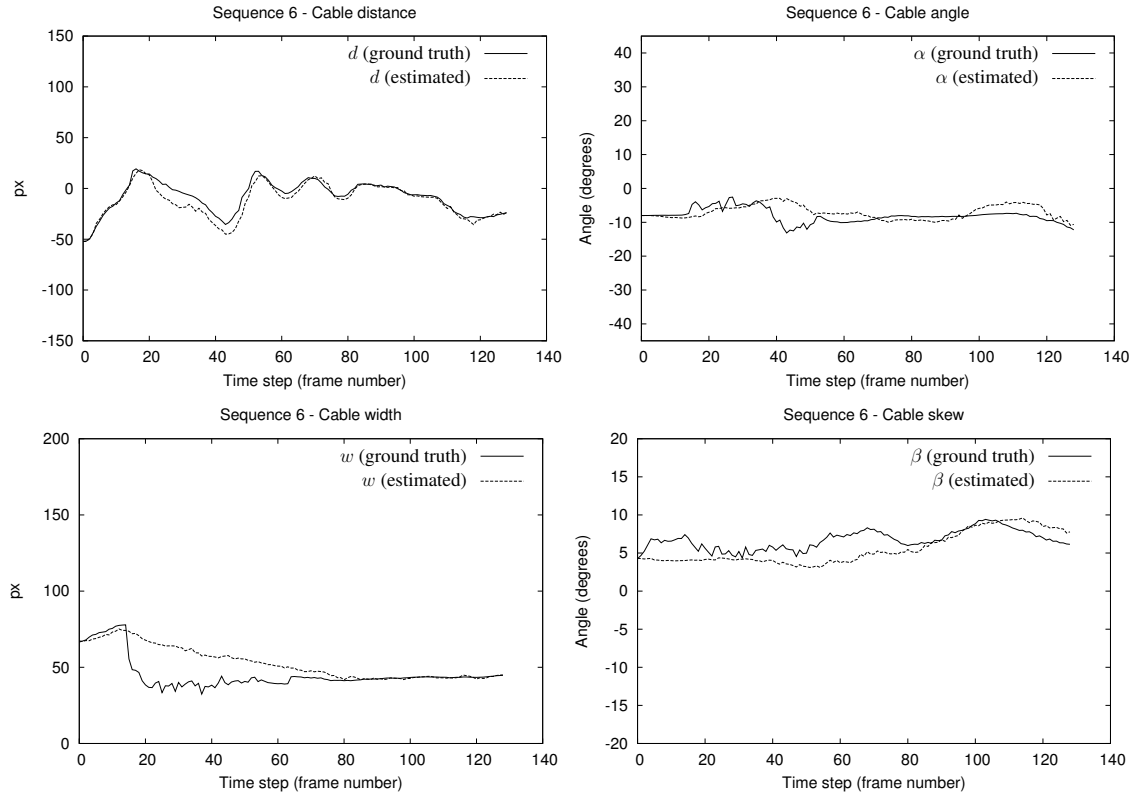


Figure 5.7: Sequence S6: ground truth and estimated parameters. Similar to the end of sequence S2, in this sequence the camera makes a hard move from looking down to looking forward (around frame number 20), which results in a rapid change of cable width w. As the movement model is not designed to follow such movements, the particle distribution needs much time to rearrange itself. Note that anyhow the target is not lost.

Parameter	minimum	maximum
d in px	-150	150
α in °	-45	45
w in px	5	100
β in °	0	10
v_d in px/frame	-1	1
v_α in °/frame	-0.1	0.1

Table 5.2: Limits for random initialization.

5.2.4 Random Initialization

All of the afore made experiments were initialized using ground truth. Before testing random initialization, we have to choose appropriate ranges for each parameter. Table 5.2 shows the ranges used for this experiment.

Figure 5.8 exemplary plots state estimations of sequence S3. For this experiment, the particle distribution was re-initialized randomly every 100 frames. Only in the second phase from frame number 100 to frame number 199 and in the last phase from frame number 300 to 399 the random initialization was successful. In the first phase, no cable was found and from frame 200 to 299, only one cable border was tracked correctly.

Tests on other sequences showed similar results. If the seabed is “clean” (with few grey level changes), the correct cable state is found quickly, otherwise, only one or no cable borders are found.

5.2.5 Performance

With the parameters from table 5.1 the system speed was considerably higher than the camera’s video rate (see table 5.3).

The process speed heavily depends on the parameters filter size, point sample distance, and particle number, because those define how many pixels are touched during the observation model application. For example, if the particle number is reduced to 300 and the filter

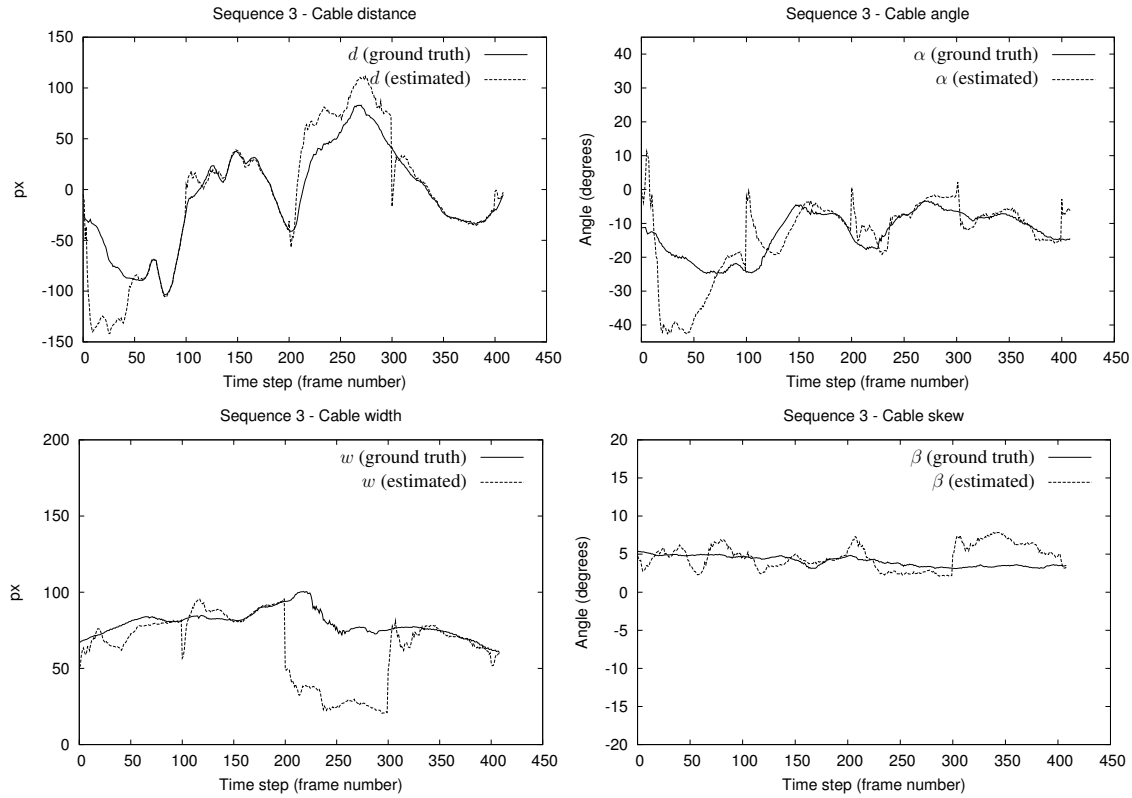


Figure 5.8: Random initialization of the prior distribution every 100 frames.

Sequence	average time per frame (ms)	frames per second
S1	17.29	57.81
S2	16.47	59.73
S3	16.92	59.10
S4	17.51	57.09
S5	16.84	59.38
S6	17.59	56.85

Table 5.3: Performance

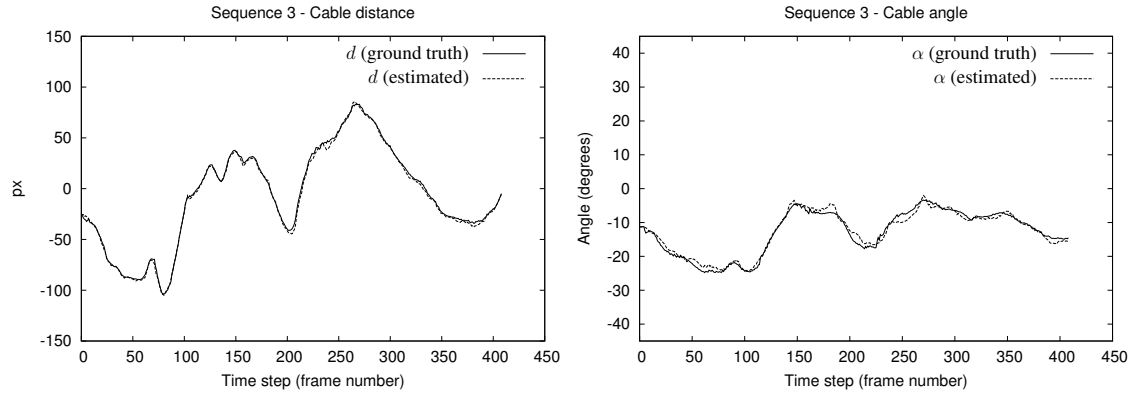


Figure 5.9: High-speed tracking results. The process speed was more than 140 frames per second in this experiment. Note that the state estimation still has good quality.

size is set to 5 px, one process cycle lasts only 7 milliseconds for sequence S3 with good tracking results (see figure 5.9).

5.3 Evaluation

The experiments showed that the proposed algorithm achieves outstanding results both in accuracy and performance. The designed movement model is not capable of tracking fast cable width and cable skew changes. In those situations, the particle distribution needs much time to rearrange itself. Nevertheless, the tracking algorithm did not loose the target. The most important parameters, cable distance and cable angle, are tracked with high accuracy. It can be assumed that the approach works even better if the scenario constraints from section 4.1 are fulfilled and the tracking output is used for the vehicle control.

The measurement model is good enough to keep the tracker focused on the target, once found. The intensity of the grey level changes at the cable's borders are a good measure for the quality of a guess. Even in clutter and in very noisy images, the hypothetical cable states are weighted higher if they are nearer to the correct state. If the borders are well-defined, a small filter size of 5 px is sufficient for a significant measurement.

Chapter 6

Conclusion

6.1 Discussion and Possible Improvements

Initialization: The proposed tracking method needs a good initial particle distribution to start tracking the correct cable state. In some situations, a random initialization is sufficient but this method is not reliable in general. I therefore propose to use some image processing based cable detection in advance to be able to limit the initial distribution, for example the approach presented in section 2.3.

Movement model: Experiments showed, that in some situations, cable movements can not be predicted correctly by the proposed movement model, especially when the cable width or the cable skew change rapidly. This problem can be resolved by adding velocity estimations for these parameters to the system, similar to those for cable angle and cable distance. The cable state vector would then have eight dimensions: $\mathbf{x} = (d, \alpha, w, \beta, v_d, v_\alpha, v_w, v_\beta)^{\mathrm{T}}$.

Observation model: If the cable borders are well-defined, the proposed observation model is sufficient for a good measurement of the quality of predicted states. However, more properties of the appearance of a cable could be introduced to make the observation model more robust. Due to the cylindric shape of the cable and the undersea lighting con-

ditions, the cable's center is mostly brighter than the rest of the cable. A larger filter could be used to test if this shape conditional grey level changes are present.

System model: The presented approach uses a very simple system model. The camera images are seen as two dimensional matrices in which two nearly parallel lines have to be found. The lack of this model is that movements of the cable state, which follow the movements of the submarine robot, are treated to be random. If more knowledge about the imaging process or the movement of the camera is introduced into the system, the cable's movements can be predicted better. Even other sensors, for example a doppler velocity logger, can be used to improve the prediction. But the change from tracking lines in the two-dimensional observation of a camera image to the tracking of the position and orientation of the whole robot with six degrees of freedom increases the dimension of the system state, hence the dimension of the PDF that estimates this state. Further research would have to be done to find out if this change is reasonable.

6.2 Summary

In this work, an approach for underwater cable tracking was developed, implemented and tested. The usage of particle filters for tracking has been motivated by reviews of related work. The concept of Bayesian tracking has been explained, together with its sample-based approach – the condensation algorithm. Models for cable tracking with particle filters have been designed with regard to the scenario of autonomous underwater cable following. Libraries for general particle filtering as well as for cable tracking have been implemented and documented. An intuitive user interface for the cable tracking particle filter has been programmed. Extensive experiments on six real underwater image sequences have been performed and evaluated. The results demonstrated the superb flexibility, performance and accuracy of this approach. Finally, possible improvements have been outlined, which points out that continuing research is possible and necessary.

Acknowledgements

This project was carried out in the Systems, Robotics and Vision Group of the Mathematics & Computer Science Department of the University of the Balearic Islands. It was partly supported by the EU program Socrates/Erasmus which made the interchange between the two Universities of Koblenz-Landau and the Balearic Islands very easy.

I would like to thank my local tutors Gabriel Oliver and Alberto Ortiz for their constant support and feedback. The work in your group was very comfortable and a lot of fun, especially during the first swim and dive tests of the robot RAO-II.

Many thanks go to my professor Dietrich Paulus who was so kind to visit me two times during my stay in Palma de Mallorca, giving me profitable inputs and belaying me in recreative climbing sessions. I want to express my gratitude to him and Johannes Pellenz also for rising my interest in the research fields computer vision and mobile robotics.

Appendix A

Video Sequences and Tracking Results

The following figures show every 10th frame of each test sequence. Refer to the figure's captions for identification and further details. The sequences were used to develop the system models for chapter 4 and to perform the test for chapter 5.

Figure A.1: Sequence S1 (248 frames).

Figure A.2: Tracking results for sequence S1.

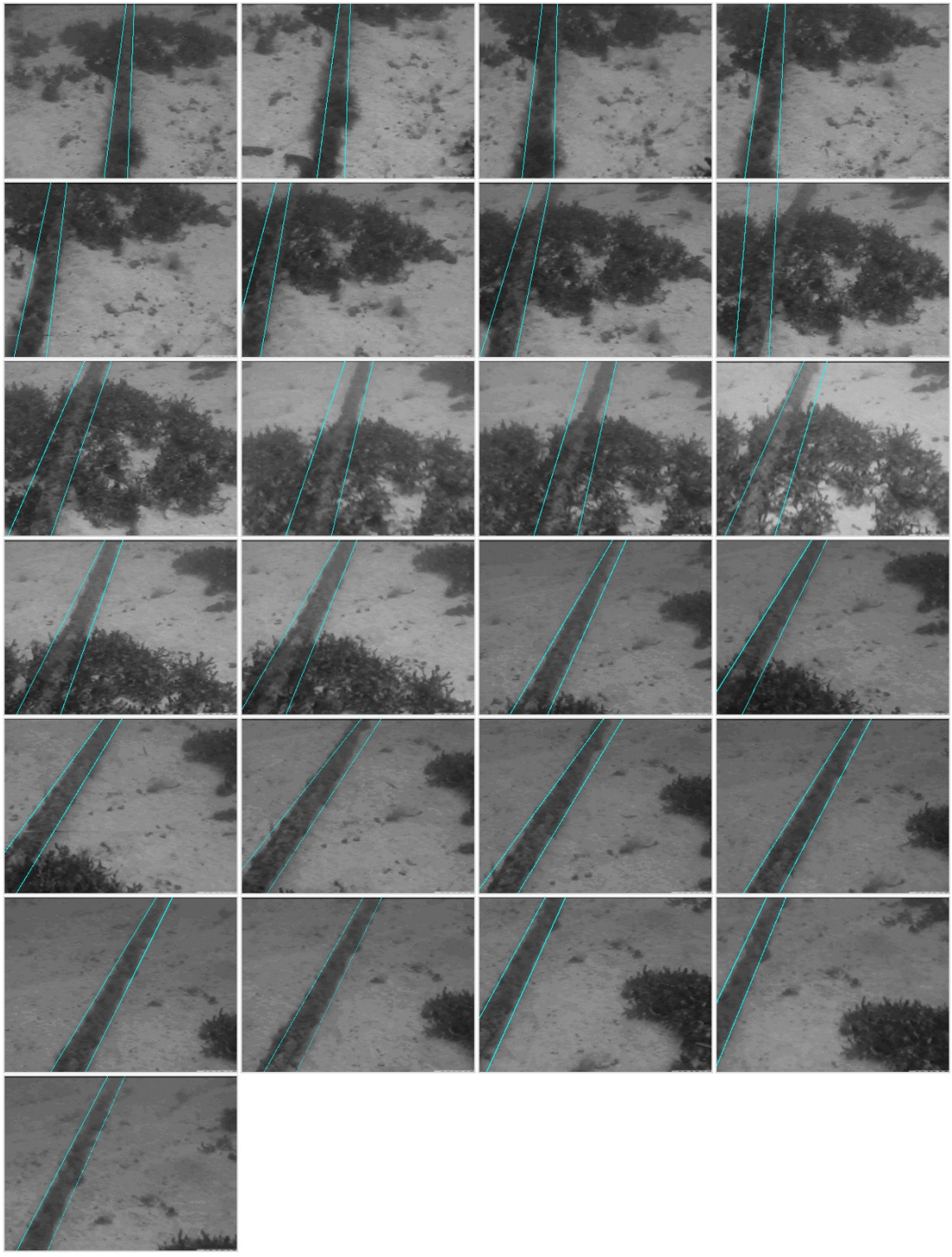


Figure A.3: Sequence S2 (499 frames), part one.

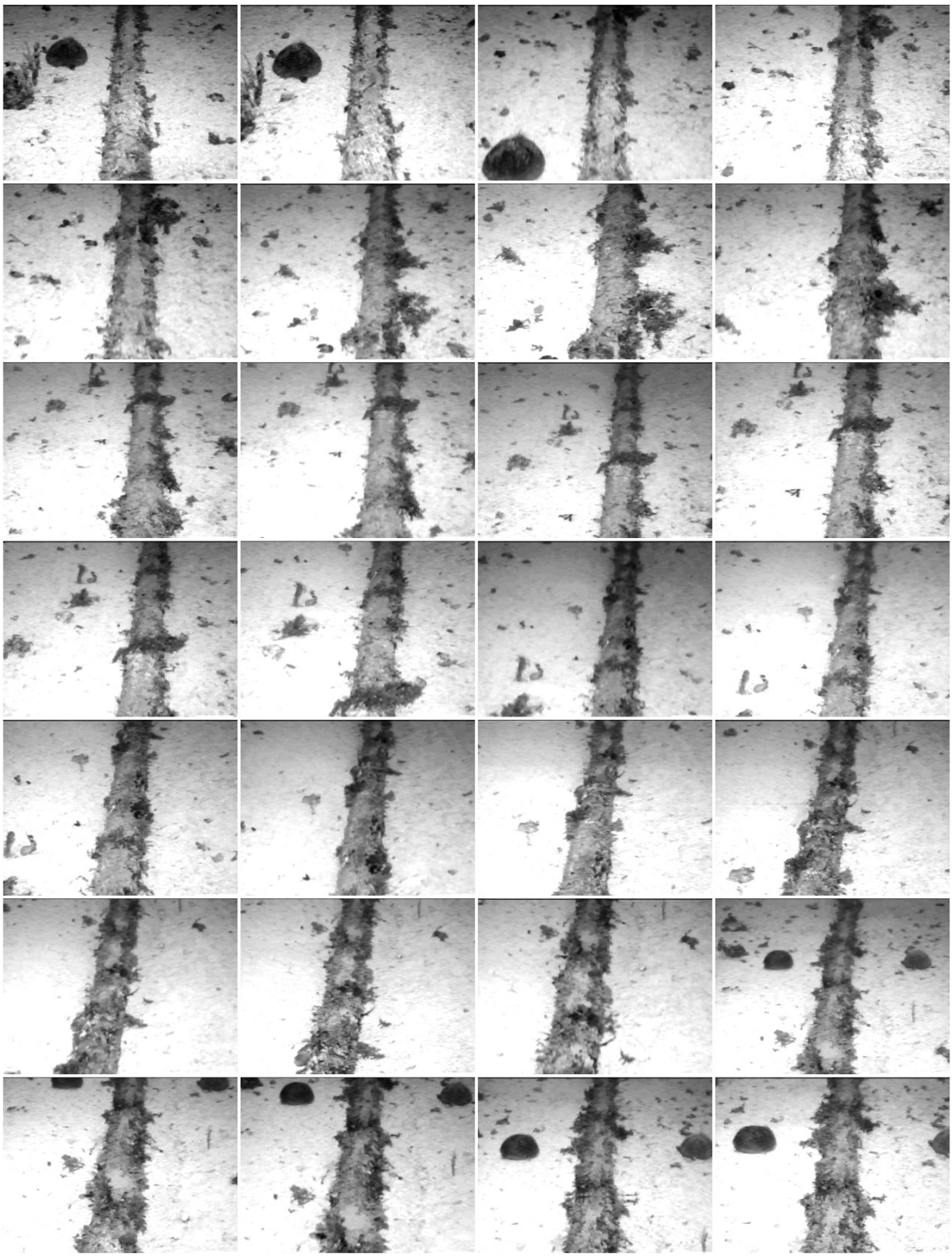


Figure A.4: Tracking results for sequence S2, part one.

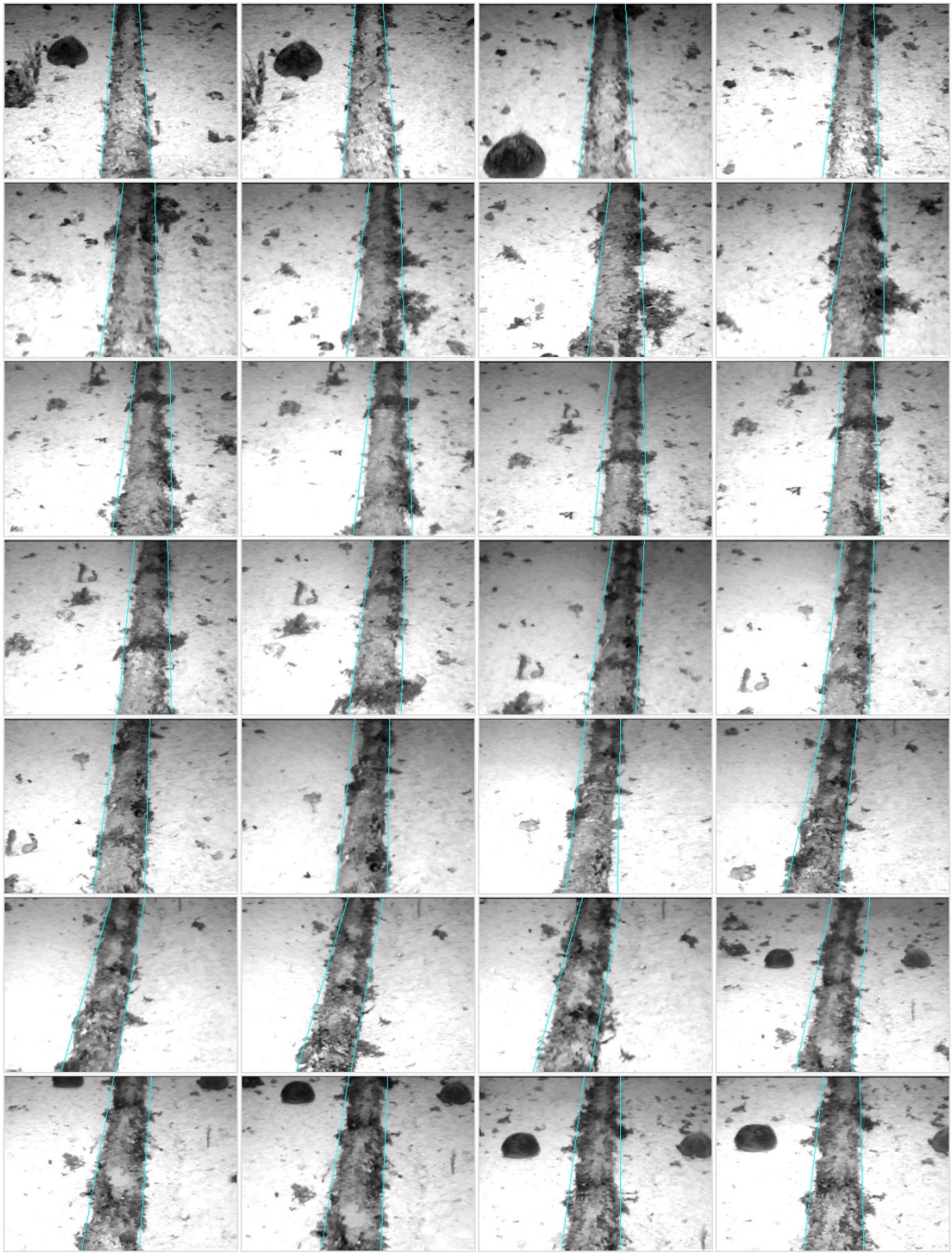


Figure A.5: Sequence S2, part two.

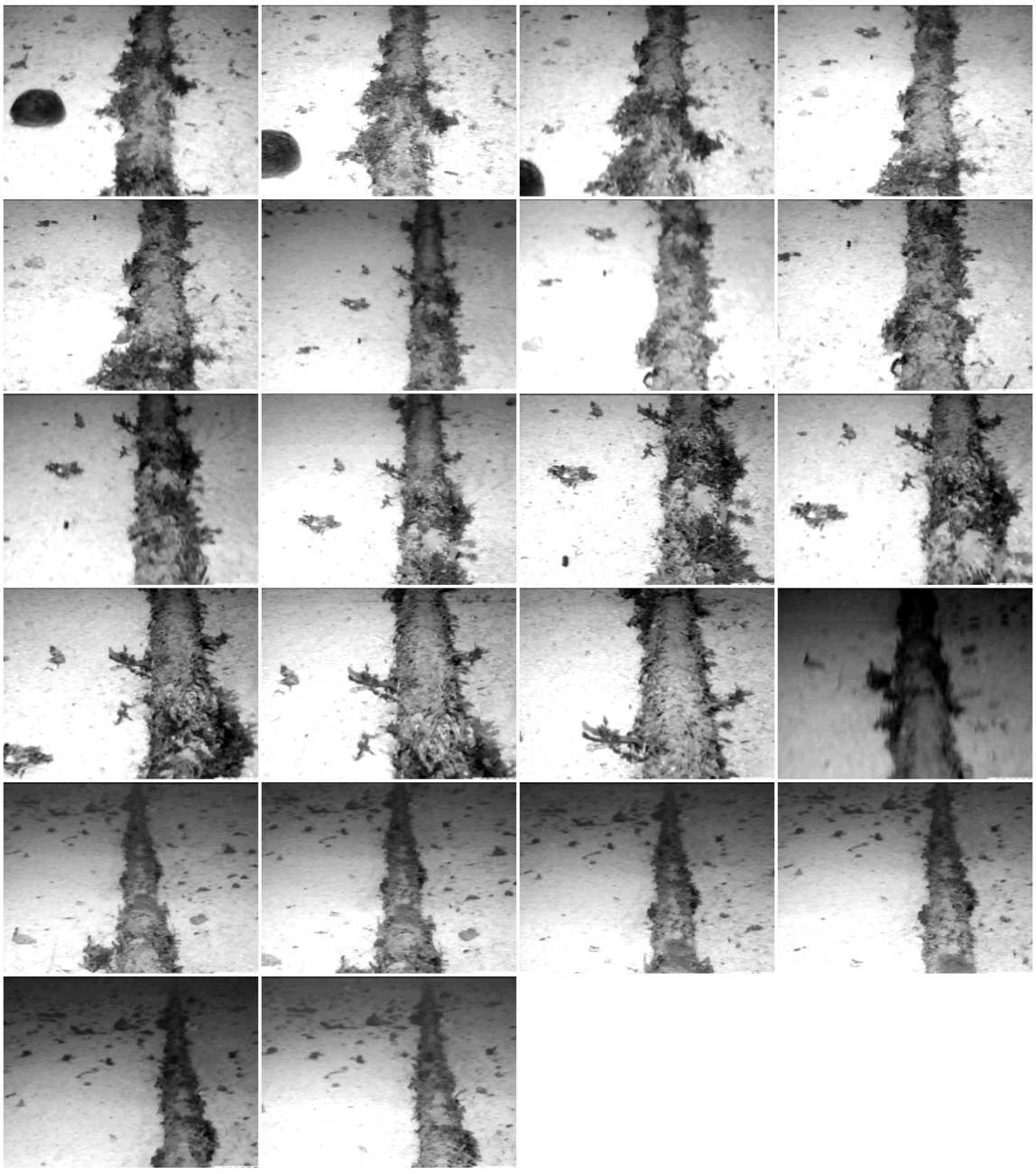


Figure A.6: Tracking results for sequence S2, part two.

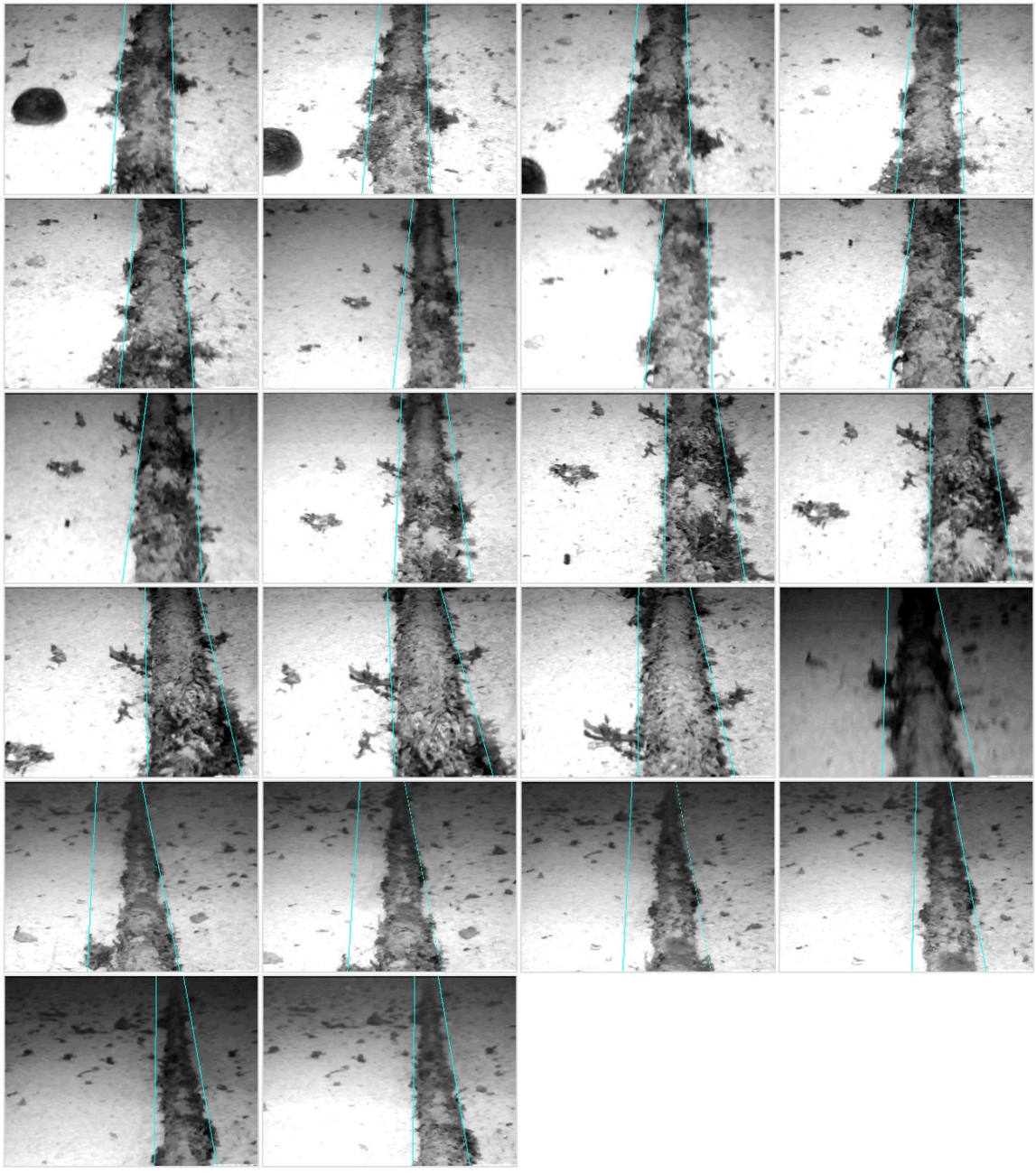


Figure A.7: Sequence S3 (409 frames), part one.

Figure A.8: Tracking results for sequence S3, part one.

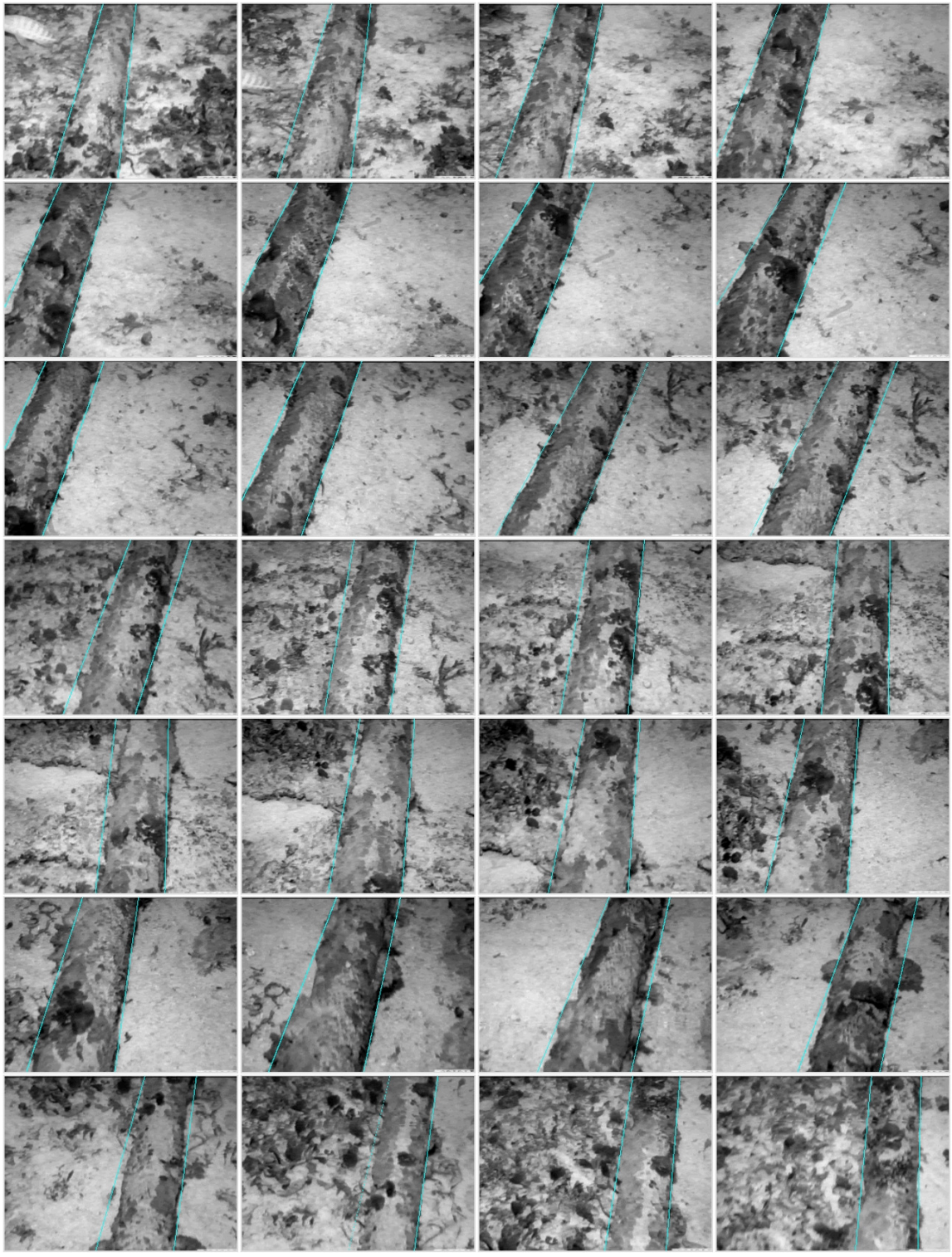


Figure A.9: Sequence S3, part two.

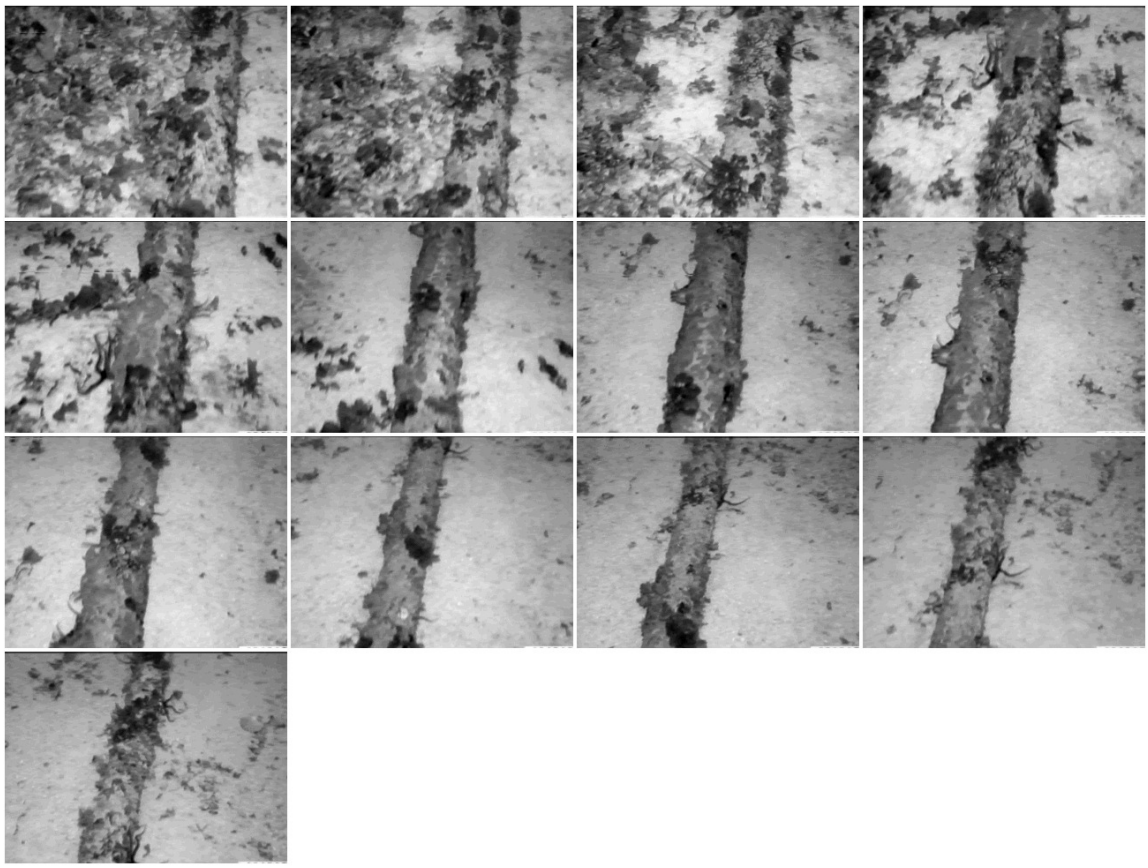


Figure A.10: Tracking results for sequence S3, part two.

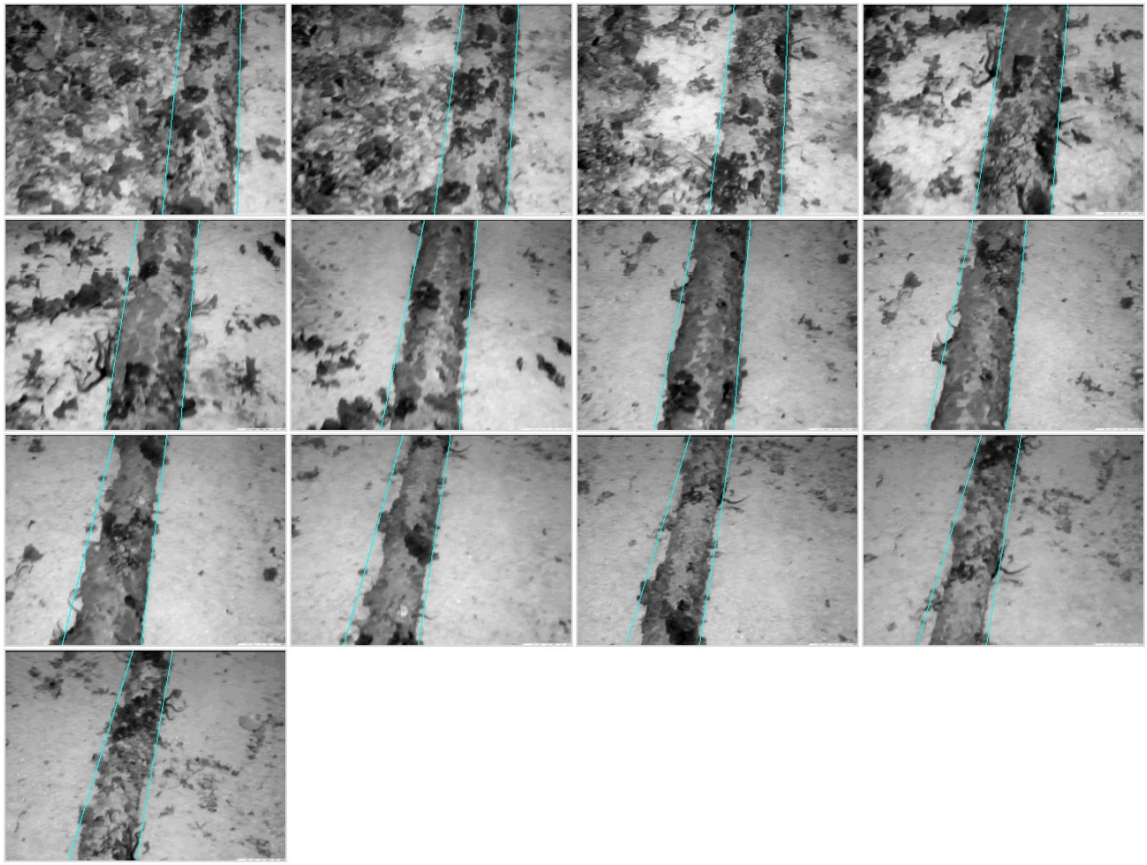


Figure A.11: Sequence S4 (172 frames).

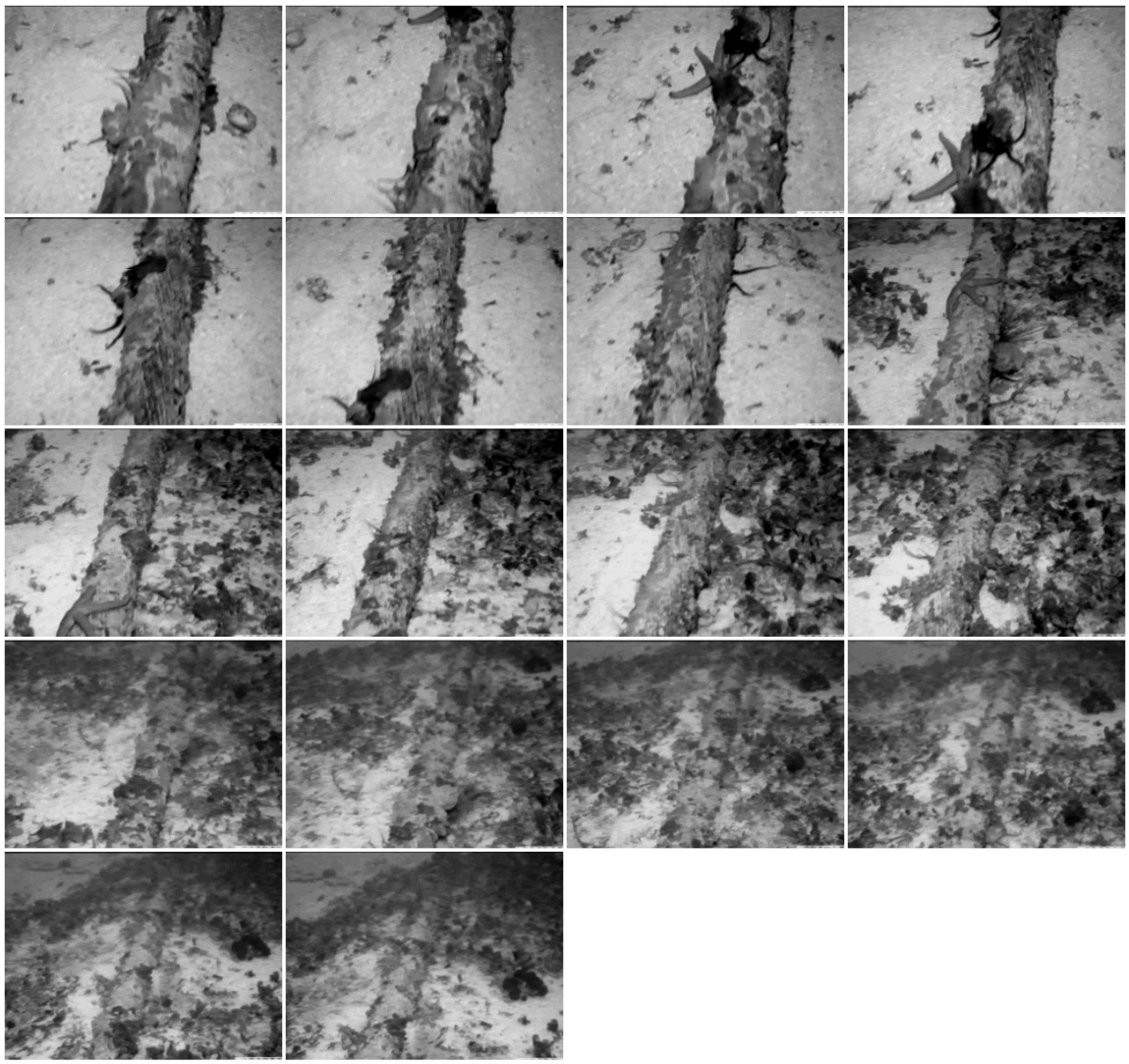


Figure A.12: Tracking results for sequence S4.

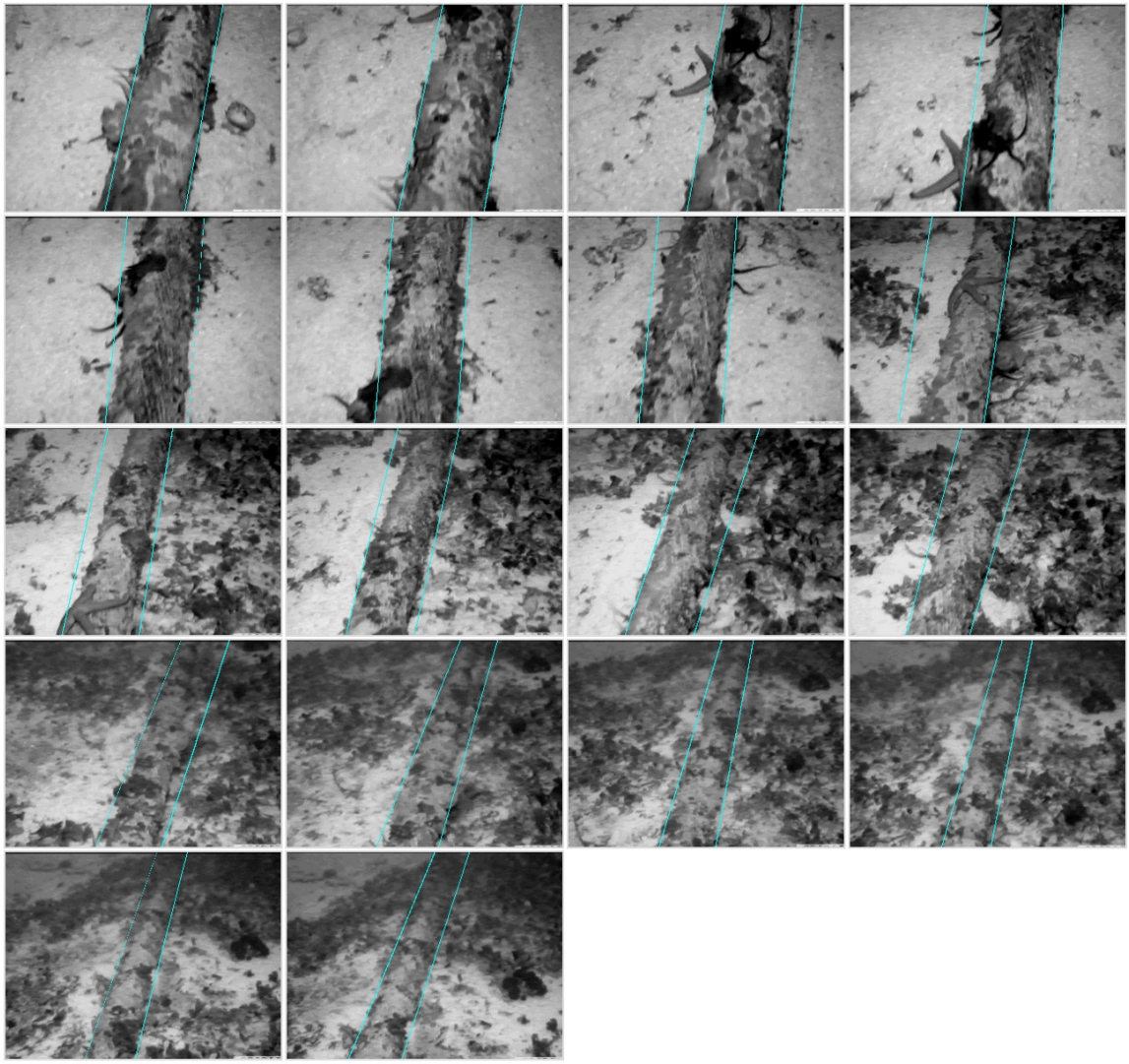


Figure A.13: Sequence S5 (171 frames).

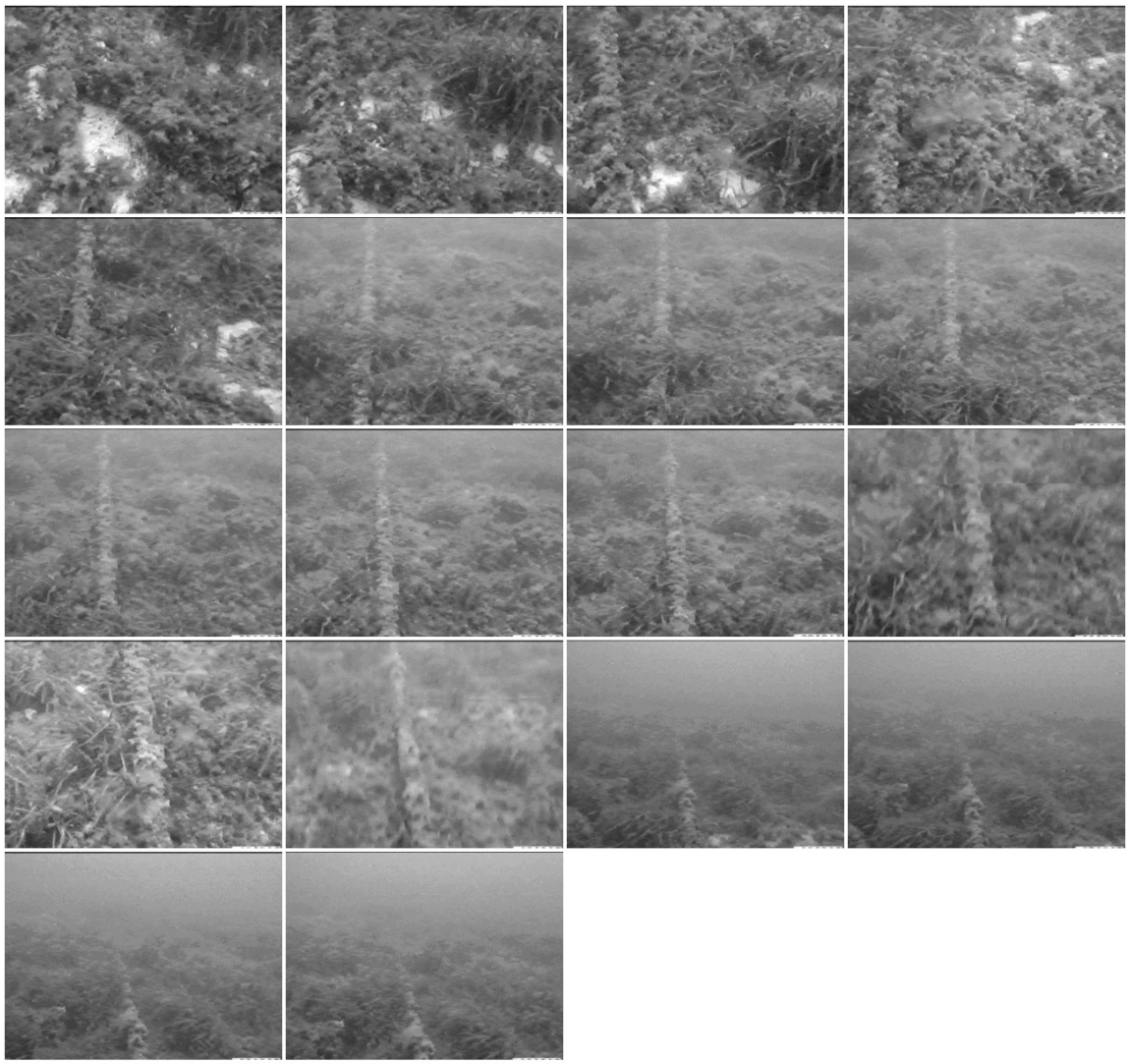


Figure A.14: Tracking results for sequence S5.

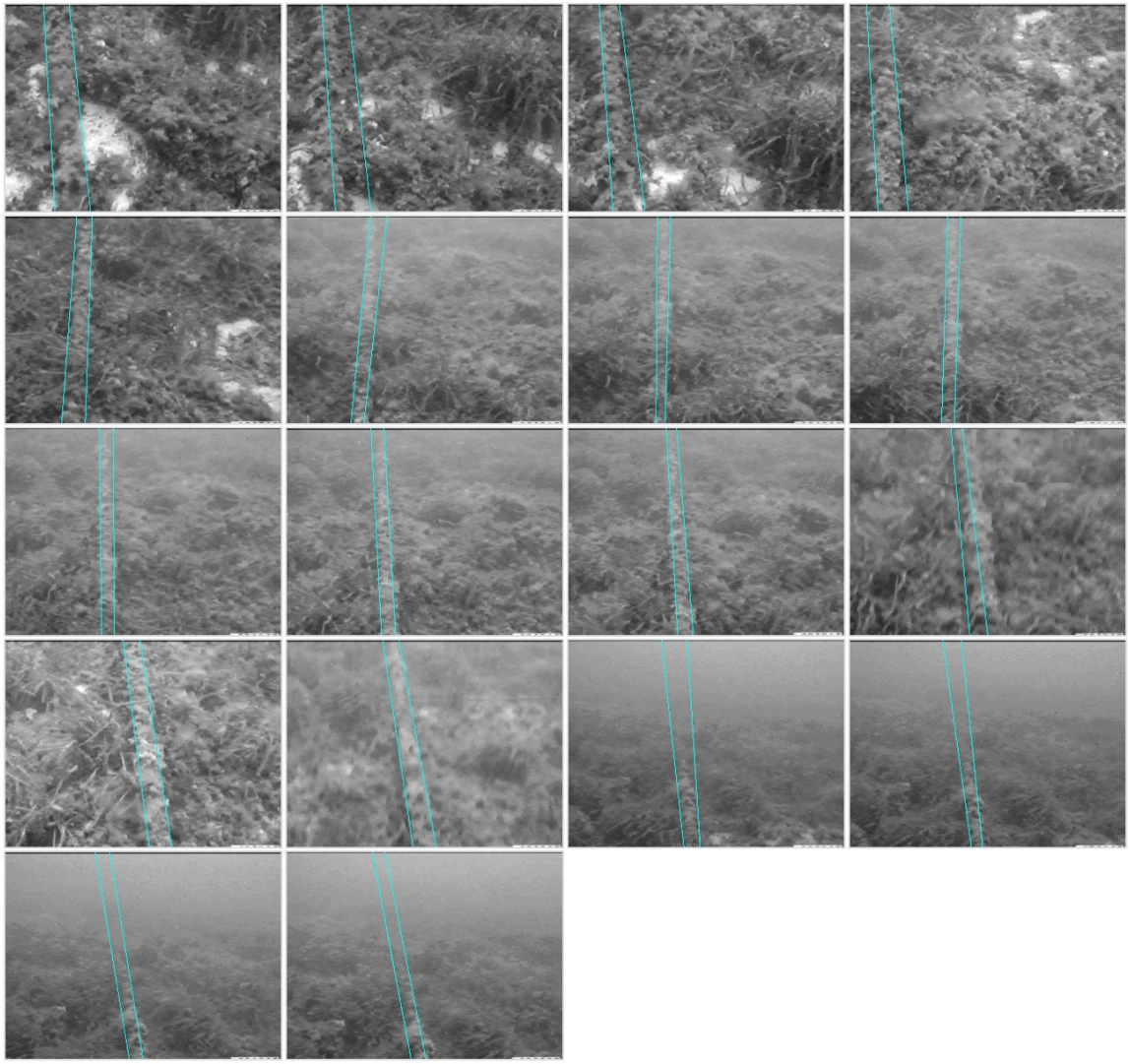


Figure A.15: Sequence S6 (129 frames).

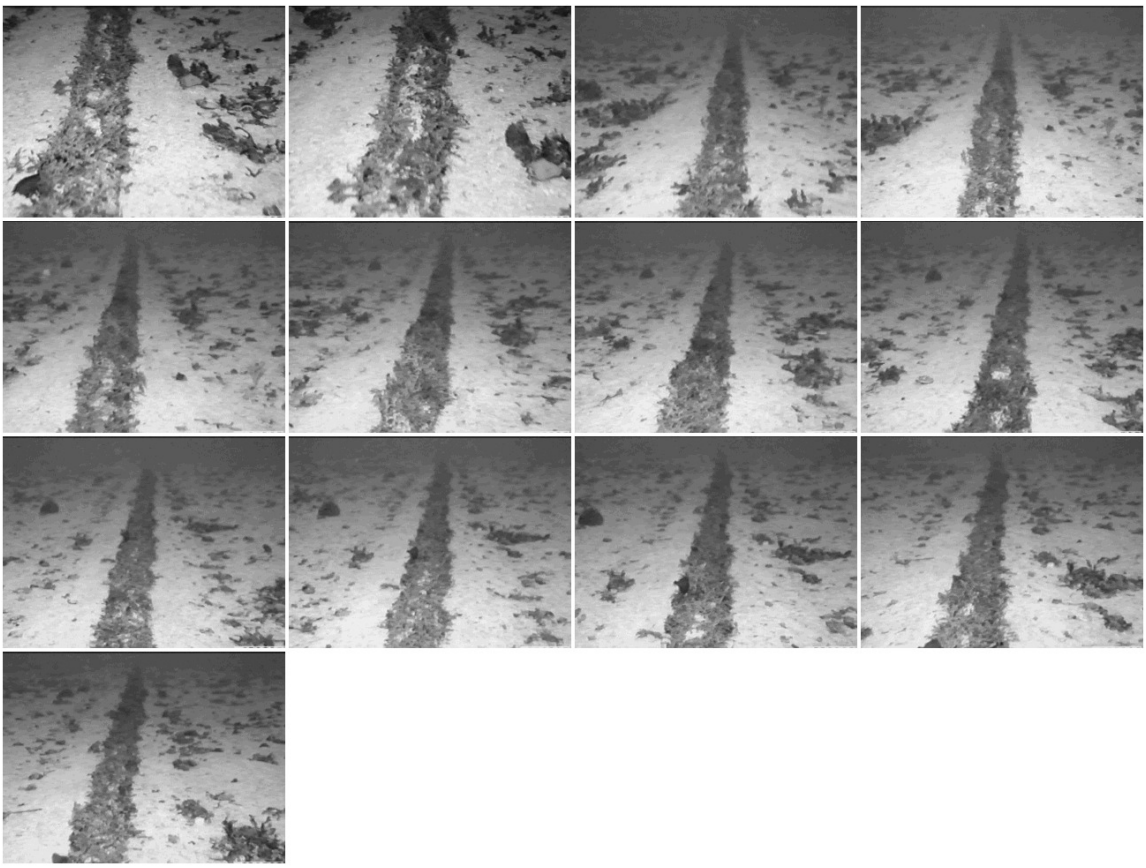
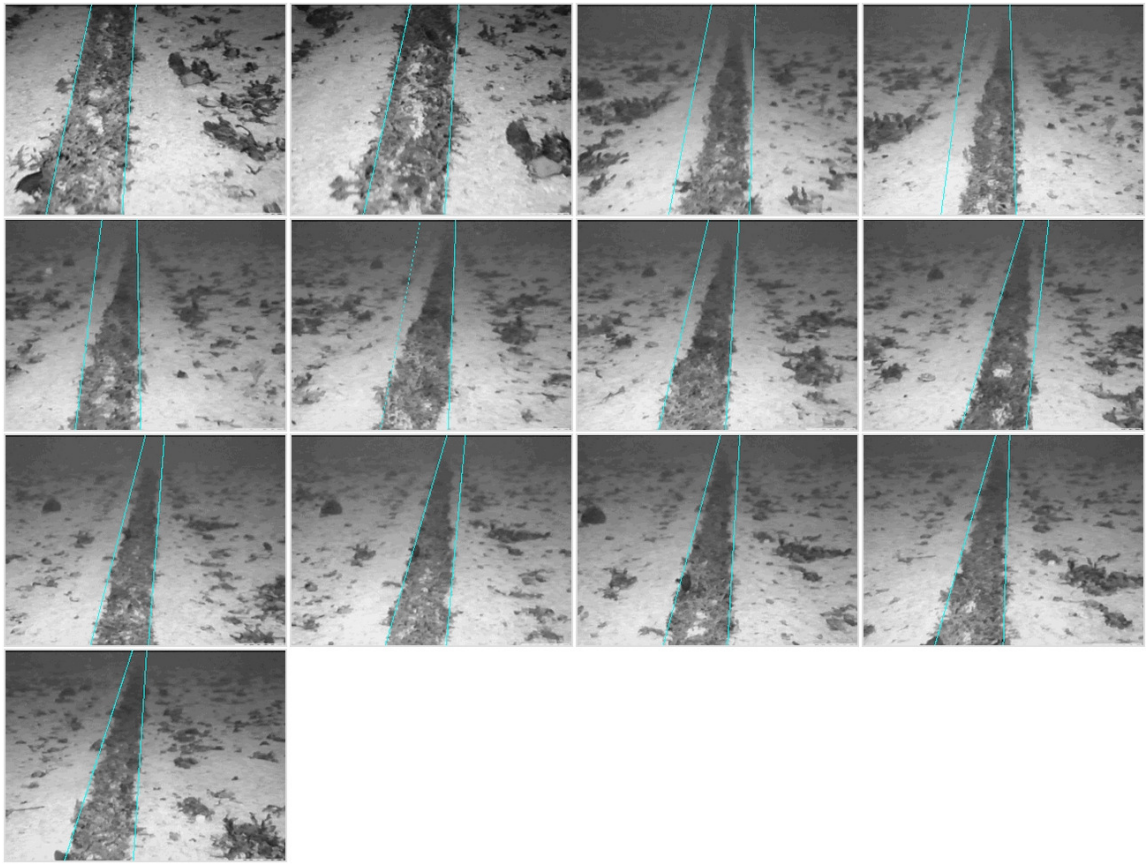


Figure A.16: Tracking results for sequence S6.



Appendix B

GroundTruthEditor

The GroundTruthEditor is a tool for annotating image sequences with ground truth data. It can be found on the enclosed CD in `/program/GroundTruthEditor`. Figure B.1 shows a screenshot of the program. The input of the program is a set of images that represent an image sequence. After labelling every image of the sequence, the data can be stored as *image sequence files*. The format of such a file is described in section 4.7.3. Image sequence files can be read by the Cable Tracking GUI which then takes the images as observations and displays the ground truth for comparison with estimation results.

To start labeling images, choose *File→Import images...* from the menu to import a set of images. Use then the slider above to choose an image and the spin boxes to the right to modify the cable parameters until the green lines are situated right on top of the cable borders. The button *Copy to next frame* copies the current data from the spin boxes to the next frame. Click on the button *Save* or choose *File→Save...* or *File→Save as...* to save the data as image sequence file. To modify an existing image sequence file, open it via *File→Open*.

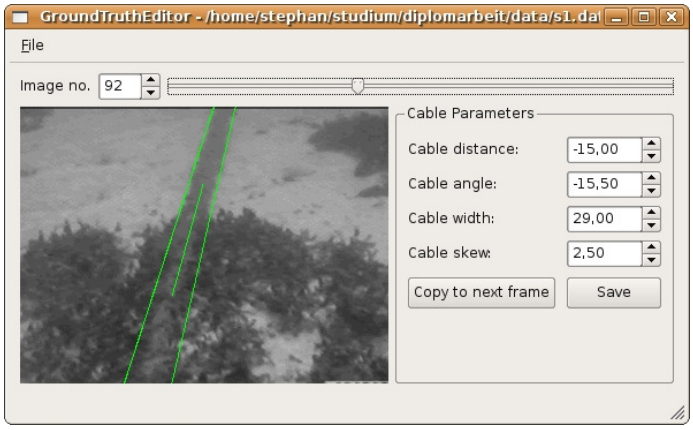


Figure B.1: Screenshot of the GroundTruthEditor

Bibliography

[AKK+02] K. Asakawa, J. Kojima, Y. Kato, S. Matsumoto, N. Kato, T. Asai, and T. Iso. Design concept and experimental results of the autonomous underwater vehicle AQUA EXPLORER 2 for the inspection of underwater cables. *Advanced Robotics*, 16(1):27–42, 2002.

[AMGC02] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-Gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[AO03] J. Antich and A. Ortiz. Underwater cable tracking by visual feedback. In Aurélio C. Campilho, Nicolas Pérez de la Blanca, Alberto Sanfeliu, and Francisco J. Perales, editors, *IbPRIA*, volume 2652 of *Lecture Notes in Computer Science*, pages 53–61. Springer, 2003.

[AOO06a] J. Antich, A. Ortiz, and G. Oliver. A control strategy for fast obstacle avoidance in troublesome scenarios: Application in underwater cable tracking. In *7th IFAC Conference on Manoeuvring and Control of Marine Craft*, Lisbon (Portugal), 2006.

[AOO06b] J. Antich, A. Ortiz, and G. Oliver. Reactive control of a visually guided underwater cable tracker. In *Robotics and Automation in the Maritime Industries*, chapter 6, pages 111–132. Producción Gráfica Multimedia, Madrid (Spain), 1st edition, 2006.

[BTL+97] B.A.A.P. Balasuriya, M. Takai, W.C. Lam, T. Ura, and Y. Kuroda. Vision based autonomous underwater vehicle navigation: Underwater cable track-

ing. In *OCEANS '97 MTS/IEEE Conference Proceedings*, volume 2, pages 1418–1424, Halifax, NS, Canada, 1997.

[BU99] B.A.A.P. Balasuriya and T. Ura. Multi-sensor fusion for autonomous underwater cable tracking. In *OCEANS '99 MTS/IEEE Conference Proceedings*, volume 1, pages 209–215, Seattle, WA, USA, 1999.

[FTBD01] D. Fox, S. Thrun, W. Burgard, and F. Dellaert. Particle filters for mobile robot localization. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, pages 499–516. Springer Verlag, New York, 2001.

[GCA98] A. Grau, J. Climent, and J. Aranda. Real-time architecture for cable tracking using texture descriptors. In *OCEANS '98 Conference Proceedings*, number 3, pages 1496–1500, Nice, France, 1998.

[GSS93] N. Gordon, D. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings Part F: Radar and Signal Processing*, volume 140, pages 107–113. IEEE Computer Society, 1993.

[HA62] P. V. C. Hough and A. Arbor. Method and means for recognizing complex patterns. Technical Report US Patent 3069 654, US Patent, 1962.

[IB98] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[Kit96] G. Kitagawa. Monte carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

[Liu96] J. S. Liu. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and Computing*, (6):113–119, 1996.

[Mac00] J. MacCormick. *Probabilistic modelling and stochastic algorithms for visual localisation and tracking*. PhD thesis, Visual Dynamics Group, Dept. Eng. Science, University of Oxford, 2000.

[MI95] S. Matsumoto and Y. Ito. Real-time vision-based tracking of submarine-cables for AUV/ROV. In *OCEANS '95. MTS/IEEE Conference Proceedings*, volume 3, pages 1997–2002, San Diego, CA, USA, 1995.

[NKMvG03] K. Nummiaro, E. B. Koller-Meier, and L. J. van Gool. An adaptive color-based particle filter. *Image Vision Computing*, 21(1):99–110, 2003.

[OOB07] G. Oliver, A. Ortiz, and F. Bonin. RAO-II: an AUV for underwater inspecion. *Instrumentation Viewpoint*, (6):50 – 51, 2007.

[OSO02] A. Ortiz, M. Simo, and G. Oliver. A vision system for an underwater cable tracker. *Machine Vision and Applications*, 13(3):129–140, 2002.

[PR78] D. Panda and A. Rosenfeld. Image segmentation by pixel classification in (gray level, edge value) space. *IEEE Transactions on Computers*, 27(11):875–879, 1978.

[SVS94] J. Santos-Victor and J. Sentieiro. The role of vision for underwater vehicles. *Proceedings of the 1994 Symposium on Autonomous Underwater Vehicle Technology*, pages 28–35, 1994.

[TP06] E. Trucco and K. Plakas. Video tracking: A concise survey. *IEEE Journal of Oceanic Engeneering*, 31(2):520–529, 2006.