Technischen Universität München
Computational Science and Engineering (Int. Master's Program)

Master's Thesis

Implementation of Contracting Curve Density Algorithm for Applications in Personal Robotics

Shulei Zhu

1st examiner:	Beetz, Michael; Prof.
2nd examiner:	Hans-Joachim, Bungartz; Prof. Dr.
Assistant advisor:	M.Sc. Pangercic, Dejan
Thesis handed in on:	April 1, 2011

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

München, den 23. März 2011 Shulei Zhu

Acknowledgments

If someone contributed to the thesis... might be good to thank them here.

Abstract

An abstracts abstracts the thesis!

Contents

1 Introduction

In the field of computer vison and pattern recognition, determining a model from a finite set of samples or a complex image without prior knowledge is an ill-posed problem. Due to the the ill-posedness, in the sense that a unique model may not exist, models specifying a priori knowledge play an very important role in solving many important computer vision problems, such as object tracking, shape recognition, pose estimation segmentation, edge detection, stereo matching and 3-D reconstruction.

Some parametric curve models, also known as deformable models, deformable templates, snakes, or active contours, have been delevoped to guides the image interpretation process towards particularly likely interpretations(CCD). These parametric curve models, namely prior knowledge have two main outstanding influences in the development of related algorithms. First is the snakes which represented a fundamentally new framework for delineating an object outline in an image. This framework attempts to minimize an energy associated to the current contour as a sum of an internal and external energy(wikipedia). The goal of snakes is to balance the prior knowledge with evidence from an image. The second outstanding influence is in the field of pattern recognition and machine learning where statistical distributions play key roles. There treatment of prior knowledge about shape is put into a probabilistic context. Any shape is regarded as the result of applying some distortion to an ideal prototype shape, and an appropriate distribution governs the nature and extent of the distortion. For both kinds of method, the common aim is to strengthen the visual interpretation of shape via the stabilising influence of prior expectations of the shapes that are likely to be seen.(AC 53) This paper concerns the Contracting Curve Density (CCD) algorithm whose basic step is solving a curve fitting problem in a probabilistic context.

1.1 The CCD algorithm and curve fitting problem

Curve fitting is such a process that makes a practical deformable templates (model) attach to the feature in a image. It in essence involves using algorithms to find the parameter values for a given parametric curve model. A lot of important and challenging computer vision tasks can be formulated as variants of the curve-ftting problem. The nature of curve fitting problem is taken into account from the outset, usually it is quite unnecessary to examine an entire image. Hence, there is no need to organise or group features in the image, and the analysis can be restricted to a relatively narrow ROI (region of interest), so it is usually expected to be computationally cheaper.

The CCD algorithm can be described as follows. Given one or multiple images as input data and a parametric curve model with a prior distribution of the model parameters, through curving fitting process we estimate the model parameters which determine the approximation of the posterior distribution in order to make the curve model best match the image data. Figure 1 depicts a curve-fitting problem and the corresponding solution obtained by the CCD algorithm.

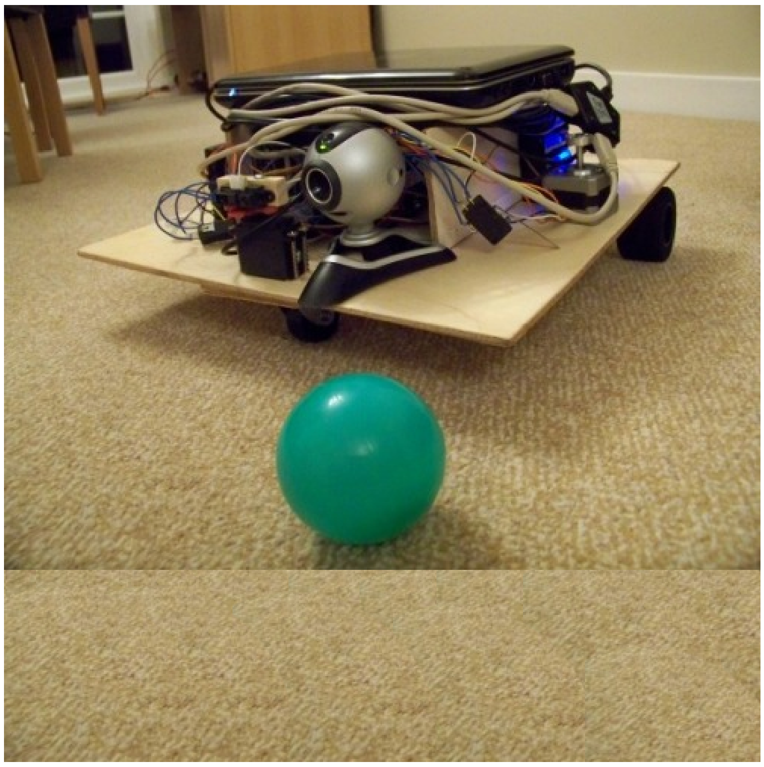

Figure 1.1:

1.2 Motivation of the CCD Approach

Curve fitting problem and its variant have a wide range of applications in the field of robotics, medical processing, user interface, surveillance and biometrics (hanek's paper). In order to be widely applicable to practical computer vision problems, following requirements, such as Robustness, accuracy, efficiency, any-time property and versatility, should be considered when a novel approach is designed and implemented. However, in the computer vision community, solving object segmentation and the related object contour tracking problem are always challenging, especially in natural and unconstrained scenes. Due to clutter, shading, texture, and highlights it is very difficulty to segment an object from an inhomogeneous background. Furthermore, some physical conditions, such as the illumination or surface properties, will influence the efficiency and stability of related approaches. It is necessary and important to develop a method to determine adequate segmentation criteria in advance or even and a single criteria that is applicable for all parts of an object boundary.

Among the currently available methodologies, the CCD algorithm is an interest one. It is developed and presented in as a state-of-the-art improvement over other advanced techniques such as the condensation algorithm (panin 2006). In the CCD approach, curve fitting problem is put in the context of probabilistic framework and converted to a regression one in the field of pattern recognition. It is especially helpful to touch the nature of curving fitting problem. By introducing local image statistics, the algorithm can cope with highly inhomogeneous image regions and provide therefore locally adapted criteria for separating the adjacent image regions. The use of blurred curve models as efficient means for iteratively optimizing the posterior density over possible model parameters. These blurred curve models enable the algorithm to trade-off two conflicting objectives, namely heaving a large area of convergence and achieving high accuracy. The CCD algorithm achieves a high level of robustness and subpixel accuracy even in the presence of severe texture, shading, clutter, partial occlusion, and strong changes of illumination.(modify)

1.3 The Contracting Curve Density (CCD) Approach

1.3.1 Math description

In the field of pattern recognition, the key concept is that of uncertainty. In a image, the uncertainty arises both through noise on measurements, as well as through the nature of objects. Probability theory provides a consistent framework for the quantification and manipulation of uncertainty. In this section, we view curving fitting problem from a probabilistic perspective and talk us towards to Bayesian treatment.

We assume a parametric curve model is governed by a prior distribution over the model parameters Φ (usually a D-dimensional vector). For simplicity, let us consider a Gaussian distribution of the form.

$$p(\mathbf{\Phi}|\mathbf{m_\Phi}, \mathbf{\Sigma_\Phi}) = \mathcal{N}(\mathbf{\Phi}|\mathbf{m_\Phi}, \mathbf{\Sigma_\Phi}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\mathbf{\Sigma_\Phi}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{\Phi} - \mathbf{\Sigma_\Phi})^T \mathbf{\Sigma_\Phi}^{-1} (\mathbf{\Phi} - \mathbf{\Sigma_\Phi})\right\} \tag{1.1}$$

where the D-dimensional vector $\mathbf{m_\Phi}$ is called the mean, the $D \times D$ matrix $\mathbf{\Sigma_\Phi}$ is called the covariance, and $|\mathbf{\Sigma_\Phi}|$ denotes the determinant of $\mathbf{\Sigma_\Phi}$

We now use local image pixels $\mathcal{L}$ as the training data to determine the likelihood function. If the data are assumed to be drawn independently from the distribution, then the likelihood function is given by

$$p(\mathcal{L}|\mathbf{\Phi}) = \prod_{n=1}^{N} \mathcal{N}(l_n|\mathbf{\Phi}) \tag{1.2}$$

where n denotes the number of groups of local discrete image pixels. The goal in the curve fitting problem is to be able to determine the posterior distribution for model parameters $\mathbf{\Phi}$.

In this paper, by pixel value we denote the vector containing the local single-or multichannel image data associated to a pixel. In our experiments, we directly use the sensed RGB values as pixel values. However, other types of local features computed in a preprocessing step may also be used, e.g. texture descriptors or color values in other color spaces

Using Bayes' theorem, the posterior distribution for $\mathbf{\Phi}$ is proportional to the product of the prior distribution and the likelihood function

$$p(\mathbf{\Phi}|\mathcal{L}) \propto p(\mathcal{L}|\mathbf{\Phi})p(\mathbf{\Phi}|\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}}) \tag{1.3}$$

We can now determine $\hat{\mathbf{\Phi}}$ by finding the most probable value of $\mathbf{\Phi}$ given the data, in other words by maximizing the posterior distribution. This technique is called maximum posterior, or simply MAP.

$$\hat{\mathbf{\Phi}} = \arg\max_{\mathbf{\Phi}} p(\mathbf{\Phi}|\mathcal{L}) \tag{1.4}$$

1.3.2 Possible problems

In order to implement the MAP estimation above, we have to address four three questions.

- How to create a parametric curve model as prior knowledge , and at same time determine the distribution? There are a variety of forms active shape models available, such as principally snakes, deformable templates and dynamic contours. For curve fitting problem, deformable template is a good choice to match the image data. Then we have to choose suitable number of parameters to control the strength of prior assumptions. If there are too many parameters in the model, it is easy to control the contour accurately at the expense of complexity and computational cost. Balance among these aspects are non-trivial.
- How can the local statistics in the image evaluated? First, it is necessary to establish the relation between model parameters $\mathbf{\Phi}$ and the image data $\mathcal{L}$. This is a especially challenging in the presence of clutter and strong texture if we consider the factors, such as illumination, surface properties , characteristics of the camera and shadow.
- How can the fit be optimized efficiently? Generally MAP and some other nonlinear optimization algorithms may find multiple local maxima and are not guaranteed to find the largest of these maxima.

For the first question, we will use parametric B-Spline curve as our prior knowledge. It is proved that this kind of curve model is flexible enough and can well represent the contour of object. The parametric B-Spline curve will be described in the chapter 3. The second question is especially challenging. In order to simplify the problem, it is necessary to make some assumptions about the image data. we will describe such assumptions in more detail. For the last question, different optimization methods used for curve-fitting will be discussed in this thesis.

1.3.3 Sketch of the CCD algorithm

In this section, the basic steps of the CCD algorithm will be sketched.

Given an input image as training data, we first choose an initial contour for an object or feature which will be fitted or tracked, and some initial values for the means, covariances. Then we alternatively between the following two updates that we shall call the learning local statistics step and the refining parameters step, for reasons that will become apparent shortly. In the step of learning local statistics, for each pixel ν in the vicinity of the expected curve, two sets of local statistics are computed, one set for each side of the curve. The local statistics are obtained from pixels that are close to pixel nu and most likely lie on the corresponding side of the curve, according to the current estimate of the model parameters. The resulting local statistics represent an expectation of ″what the two sides of the curve look like″.(Hanek's paper)

The CCD algorithm has some interesting similarities to the Expectation-Maximization (EM) algorithm (Dempster et al., 1977), which is often used for clustering-based image segmentation, a subset of region-based image segmentation (Hermes et al., 2002; Belongie et al., 1998). The first step computes local statistics defining an expectation of the pixel values (E-step). The second step maximizes this expectation (M-step). The CCD algorithm differs mainly by: 1.) using local statistics, 2.) exploiting a curve model and optimizing model parameters rather than pixel classifications.

1.4 Overview of the Thesis

The remainder of this paper is organized as follows. In Chapter 2, related work on PR2, Robot Operation System (ROS), active contour, some curve-fitting ,image segmentation and tracking algorithms will be introduced. Details on shape-space models and parametric B-Spline curves are discussed in chapter 3. In chapter 4 , we give a brief introduction of software and hardware infrastructure used in the implementation of the CCD approach. We will concern the OpenCV library, ROS and PR2, and chapter 5 explain the CCD approach in detail. This will be followed by a description of the application of the CCD approach in robotics. In chapter 7, experiments and results are given. In addition, we evaluate the performance of the CCD algorithm and the CCD tracker in terms of robustness, accuracy, and runtime. In chapter 8, the work of this thesis is concluded with the future work for improvement.

2 Related work

3 Software and Hardware Infrastructure

3.1 The Robot Operation System (ROS)

3.2 PR2

4 Shape-space Models and B-Spline Curves

In the CCD algorthms, A parametric curve model is resposible for restricting Region of Interest (ROI) and providing prior knowledge. Throughout this thesis, visual curves are represented in terms of parametric spline curves which is widely applied in computer grapics.

In this chapter, we introduce a deformable 2-D model called shape-space models. In the first section, we start by explaining spline functions and their construction. Details on how parametric curves are constructed from spline functions are discussed in the later section. This forms the basis for the CCD approach.

4.1 B-Spline Function

In the mathematical, every spline function of a given degree, smoothness, and domain partition, can be represented as a linear combination of B-splines of that same degree and smoothness, and over that same partition(wikipedia). In terms of this statement, we can construct a spline curve parametrized by spline functions that are expressed as linear combinations of B-splines.

Given a nondecreasing vector of real values $U = [u_0, \dots, u_{m-1}]$ called knot vector, with u_i is called the knot, a B-spline of degree n is a parametric curve

$$\mathbf{C}(u) = \sum_{i=0}^{m-n-2} P_i B_{i,n}(u)\ ,\ u \in [u_n, u_{m-n-1}] \tag{4.1}$$

where P_i are called control points, $B_{i,n}$ are called basic function. For n=0,1,...,m-2, the m-n-1 basis B-splines of degree n can be defined as

$$\begin{aligned} B_{i,0}(u) &= \begin{cases} 1 & \text{if} \quad u_i \le u < u_{i+1} \\ 0 & \quad \text{otherwise} \end{cases}, \qquad i = 0, \dots, m{-}2 \\ B_{i,n}(u) &= \frac{u - u_i}{u_{i+n} - u_i} B_{i,n-1}(u) + \frac{u_{i+n+1} - u}{u_{i+n+1} - u_{i+1}} B_{i+1,n-1}(u), i = 0, \dots, m{-}n{-}2. \end{aligned} \tag{4.2}$$

Furthermore, we can write the B-spline compactly in matrix notation as

$$\mathbf{C}(u) = \begin{pmatrix} \mathbf{B}(u)^T & 0 \\ 0 & \mathbf{B}(u)^T \end{pmatrix} \mathbf{P} \tag{4.3}$$

where vector $\mathbf{P}$ denotes the axis components of control points

$$\mathbf{P} = \begin{pmatrix} P_x & P_y \end{pmatrix}^T \quad \text{where} \quad P_x = \begin{pmatrix} P_0^x \\ \cdots \\ \cdots \\ P_{N-1}^x \end{pmatrix} \tag{4.4}$$

Where N denotes the number of control points. $\mathbf{B}(u)$ is a vector of basis function

$$\mathbf{B}(u) = (B_0(u), \ldots, B_{N-1}(u))^T \tag{4.5}$$

The B-spline is said to be uniform when the knots are equidistant, otherwise non-uniform. In this thesis, for simplicity, we only consider the uniform case.

As two special examples of the B-Spline, we consider the matrix form and derivative of uniform quadratic and cubic B-Spline respectively.

4.1.1 Uniform quadratic B-Spline

Because the knot-vector is equidistant, the blending function can easily be computed. In each segment, the blending function shares the same form as following

$$\mathbf{b}_{i,2}(u) = \begin{cases} \frac{1}{2}u^2 \\ -u^2 + u + \frac{1}{2} \\ \frac{1}{2}(1-u)^2 \end{cases} \tag{4.6}$$

Its derivative is

$$\mathbf{b}'_{i,2}(u) = \begin{cases} u \\ -2u + 1 \\ u - 1 \end{cases} \tag{4.7}$$

The matrix-form of uniform quadratic B-Spline is given by

$$\mathbf{B}_i(u) = \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} \tag{4.8}$$

4.1.2 Uniform cubic B-Spline

Uniform cubic B-Spline is most commonly used form of B-Spline, The blending function is

$$\mathbf{b}_{i,3}(u) = \begin{cases} \frac{1}{6}(-u^3 + 3u^2 - 3u + 1) \\ \frac{1}{6}(3u^3 - 6u^2 + 4) \\ \frac{1}{6}(-3u^3 + 3u^2 + 3u + 1) \\ \frac{1}{6}u^3 \end{cases} \tag{4.9}$$

Its derivative can be written as

$$\mathbf{b}'_{i,3}(u) = \begin{cases} \frac{1}{2}(-u^2 + 2u - 1) \\ \frac{1}{2}(3u^2 - 4u) \\ \frac{1}{2}(-3u^2 + 2u + 1) \\ \frac{1}{2}u^2 \end{cases} \tag{4.10}$$

Its matrix-form is given by

$$\mathbf{B}_i(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \mathbf{p}_{i+2} \end{bmatrix} \tag{4.11}$$

4.2 Parametric B-spline curve

In a 2-D image, a curve is composed of many discrete points which are identified by its coordinates. For those points on a curve, their axis components x, y are particular functions of knots value u, known as splines. Hence, we can define a B-spline curve as

$$\mathbf{C}(u) = (x(u), y(u))^T = \mathbf{U}(u)\mathbf{P} \tag{4.12}$$

where $\mathbf{U}(u)$ denotes the matrix mapping control point vector P to image curve $\mathbf{C}(u)$

$$\mathbf{U}(u) = \begin{pmatrix} \mathbf{B}(u)^T & 0 \\ 0 & \mathbf{B}(u)^T \end{pmatrix} \tag{4.13}$$

4.2.1 Norm for B-spline curves

Norm and inner product of curves are useful in curve approximation. In an image plan, using the Euclidean distance measure, we can define the norm for curves as

$$||\mathbf{C}||^2 = \mathbf{C}^T \mathcal{U} \mathbf{C} \tag{4.14}$$

where $\mathcal{U}$ represents the metric matrix for curves, it can be defined as

$$\mathcal{U} = \begin{pmatrix} \mathcal{B}^T & 0 \\ 0 & \mathcal{B}^T \end{pmatrix} \tag{4.15}$$

$\mathcal{B}$ denotes the metric matrix for a given class of B-spline function

$$\mathcal{B} = \frac{1}{L} \int_0^L \mathbf{B}(u)\mathbf{B}(u)^T \mathrm{d}u \tag{4.16}$$

4.3 Shape-space models

The shape-space $\mathcal{S}$ is a linear parameterisation of the set of allowed deformations of a base curve(AC). From the perspective of mathematics, A shape-space $\mathcal{S} = \mathcal{S}(\mathbf{A}, \mathbf{P})$ is a linear mapping of a shape-space vector $\mathbf{\Phi} \in \mathbb{R}^{N_{\mathbf{\Phi}}}$ to a spline vector $\mathbf{C} \in \mathbb{R}^{Np}$:

$$\mathbf{C} = \mathbf{AX} + \mathbf{P} \tag{4.17}$$

where T is a $Np \times N_{\mathbf{\Phi}}$ shape-matrix. The constant P_0 is a deformable template curve. (AC), Usually, the dimension $N_{\mathbf{\Phi}}$ of the shape-space vector is clearly smaller than the dimension of the spline-vector Np. This makes the problem simplicity and computationally cheaper.

In this chapter, we consider a important shape space named planar affine. The planar affine has 6 degree of freedom: horizontal translation, vertical translation, rotation, horizontal scale, vertical scale and diagonal scale. The planar affine group can be viewed as the class of all linear transformations that can be applied to a template curve **C**(AC).

$$\mathbf{C}(\mathbf{u}) = \mathbf{T} + \mathbf{M}\mathbf{C}_0(u) \tag{4.18}$$

where $T = (T_1, T_2)^T$ represents the two-dimensional translation vector, and M is 2×2 matrix to control the rotation and scaling, so that T and M together can represent the 6 degree of freedom of planar Affine space. We can write shape-matrix **A** as

$$A = \begin{bmatrix} 1 & 0 & P_0^x & 0 & 0 & P_0^y \\ 0 & 1 & 0 & P_0^y & P_0^x & 0 \end{bmatrix} \tag{4.19}$$

The model parameters can be interpreted as

$$\mathbf{\Phi} = (T_1, T_2, M_{11} - 1, M_{22} - 1, M_{21}, M_{12}) \tag{4.20}$$

The planar affine shape-space discussed above can be extended for three-dimensional affine shape-space, which is more suitable for a non-planar object. The new space-model needs 8 degrees of freedom to make up of the six-parameter planar affine space and a two-parameter extension. The transformation matrix for the three-dimensional case can be written as:

$$A = \begin{bmatrix} 1 & 0 & P_0^x & 0 & 0 & P_0^y & P_0^z & 0 & \\ 0 & 1 & 0 & P_0^y & P_0^x & 0 & 0 & 0 & P_0^z \end{bmatrix} \tag{4.21}$$

the three-dimensional affine shape-space components have the following interpretation:

$$\mathbf{\Phi} = (T_1, T_2, M_{11} - 1, M_{22} - 1, M_{21}, M_{12}, \nu_1, \nu_2) \tag{4.22}$$

where ν_1, ν_2 are the two additional basis elements.

5 The CCD Algorithm

In this chapter, we describe the underlying principles of the Contracting Curve Density (CCD) algorithm, which is shown to over-perform in many challenging problems in the field of computer vison and robotics(paper list). Generally, as mentioned in section 1.3.3, the algorithm includes three steps: initialization, learning local statistics and model parameters refinement. We describe these steps in the following sections respectively.

5.1 Pre-processing and model parametrization

The input of the CCD algorithm is image data (one or multiple images) and parametric curve model. Hence, initialization step comprises pre-processing of input data and model parametrization.

5.1.1 Pre-processing of input data

Before we start to processing step, it is required to improve the quality of the images though the CCD algorithm is proved robust for noise and clutter information in images. Sometimes pre-processing make a problem easier to solve because it greatly reduces the variability , and is helpful to extract features. Furthermore, pre-processing might be performed in order to speed up computation. In our implementation, the idea case is that input data is noiseless, shadowless and have sharp edges or boundaries between objects. There are a group of operations corresponding to this aim.

- **Noise reduction**. Many linear and non-linear algorithms can help to remove noises, such as mean filer, Gaussian blur, Bilinear interpolation and so on.
- **Contrast improvement**. Usually, we can use a window to adjust brightness and contrast by selecting a range of input values which are mapped to the full output range of intensities.
- **Sharpening and detail enhancement**. Scale space is one of methods to highlight and enhance fine details in a image.

OpenCV provides a series of implementation of these operations. Note that for different input data, according to the object properties, illumination and other physical conditions, different operations should be taken.

5.1.2 Contour initialization

After the pre-processing of input data, let's start to discuss the initialization of model contour. It is followed by the process of model parametrization.

In our implementation, we model the contour as a continuous, differentiable and uniform quadratic or cubic B-Spline in $\mathbb{R}^2$, which is discussed in chapter 3. Given a ROI (region of interest), the first step is generating sufficient control points $\mathbf{P} = P_0, \ldots, P_{N_p-1}$ to do justice to the complexity of the object shape (add a image here).

Now there are two major methods to generate control points: manual initialization and intelligent initialization. The former is given as a simple form by hand. It is easy to use and control. However, there are some problems and limitations in case that the cost of manual operation is large, or vision deviation due to similar brightness between object and background. Therefore, a few new intelligent initialization methods are proposed to extract the contour. In this paper, a initial contour estimation method is described and implemented by employing the well-known SIFT algorithm (source) for keypoint detection and matching. Chapter 6.1(?) be discussed this in detail.

5.1.3 Model parametrization

By applying the (uniform quadratic or cubic) B-spline interpolation to the control points, a new curve grouped by a sequence of equidistant distributed points is generated (add image). The B-spline curve is defined in (Formula 4.12) and is composed of a sequence of points $\mathbf{C} = \{C_0, \ldots, C_k\}, k = N_{C-1}$. N_c denotes the number of sample points in the parametric curve. Because the parametric curve is continuous and differentiable, we can compute the normal $\vec{\mathbf{n}} = \{\vec{n}_0, \ldots, \vec{n}_k\}$ and tangent vector $\vec{\mathbf{t}} = \{\vec{t}_0, \ldots, \vec{t}_k\}$ conveniently.

In the planar affine shape-space $\mathcal{S}$. The curve, a hypothetical initial estimate, can be compactly represented using a vector with 6 real elements $\mathbf{\Phi}$, namely model parameters. From the perspective of probability, the hypothetical curve gives a uncertainty, the solution of the curving problem is therefore no longer just a single curve, but a whole family of possible curves. The Gaussian probability density for these possible curves in shape-space $\mathcal{S}$ is:

$$p(\mathbf{\Phi}) \propto \exp\left\{-\frac{1}{2}(\mathbf{\Phi} - \mathbf{\Sigma_\Phi})^T \mathbf{\Sigma}_{\mathbf{\Phi}}^{-1}(\mathbf{\Phi} - \mathbf{\Sigma_\Phi})\right\} \tag{5.1}$$

As claimed in chapter 1, $\mathbf{\Sigma_\Phi}$ is a 6×6 covariance matrix, which measure the variability of how much two group of model parameters change together. The information matrix $\mathbf{\Sigma}_{\mathbf{\Phi}}^{-}1$ can be written as

$$\mathbf{\Sigma}_{\mathbf{\Phi}}^{-1} = \frac{N_{\mathbf{\Phi}}}{\rho_0^2}\mathbf{A}^T \mathcal{U} \mathbf{A} \tag{5.2}$$

where ρ_0^2 denotes the mean-square displacement along the entire curve (AC). $\mathbf{A}$ is the shape-matrix, and $\mathcal{U}$ is metric matrix for curves. $N_{\mathbf{\Phi}}$ represents the number of model parameters, for our case, it is 6. Note ρ_0^2 is a real value and can be computed easily as

$$\rho_0^2 = \mathrm{tr}(\mathbf{\Sigma_\Phi}) \tag{5.3}$$

where $\mathrm{tr}(\cdot)$ denotes the trace of a matrix.

The pre-processed input data and parametric curve model have been prepared. In the following sections, the iterative procedure of the CCD algorithm will be described in detail.

5.2 Local statistics

5.2.1 Collecting local information

As claimed in chapter 1.1, one of advantages of the curve-fitting problem is that the problem can be restricted to a ROI, and is expected to reduce the computational cost. Therefore, we first define the region which contains pixels in the vicinity of expected image curve. Consider complexity and the expense of computing, it is practicable that choose those pixels actually required for interpolation along normals of the curve. This is proved useful and important for image perception and real-time tracking system. In the implementation of this thesis, processing is limited to a segment of each normal within a search region. A fixed distance h along the normal segment are choose according hypothetical uncertainty of parametric curve. In the special case of the norm-squared prior in spline space, a reasonable search segment is

$$h = \sqrt{2}\rho_0 = \sqrt{\mathrm{tr}(\mathbf{\Sigma_\Phi A}^T \mathcal{U} \mathbf{A})} \tag{5.4}$$

Usually, h denotes the size of *window* used for computing local statistics. In the begining of interative procedure, the value is relatively big and only describe the vicinity of the image curve roughly due to the high uncertainty. The uncertainty is reduced after some iteration steps, as a result, the h becomes smaller and smaller. After determining the length of search segment, a set of points located on these segments can be collected and evaluated. Note the parametric model curve is not required to be closed, but whatever case it encloses a limited area. We only plan to analyze the pixels located in the vicinity of the contour. Therefore, we should pay attention to limit the search distance on the *internal* side in order to avoid crossing the opposite boundary to sample pixels from the wrong area (ICVS06). In order to decrease the computational expenses, it is advisable to uniformly sample those pixels on both segments in the vicinity of the contour. In the other hand, we should avoid to only collect a small number of pixels, thus it makes no sense from the perspective of statistics. Let's denote the sample distance using δh, then the an overall number of spaced sample points L ($2L$ for both sides in all) can be given by

$$L = \lfloor \frac{h}{\delta h} \rfloor \tag{5.5}$$

As mentioned before, the goal of the algorithm is assignment each pixel ($\mathrm{v}_{k,l}, k \in [0, \ldots, N_\mathrm{C} - 1], l \in [0, 2L-1]$) to either side of the contour, which is determined by the curve distribution. We first compute the probabilistic assignments $\mathbf{a}_v$ for each pixel $\mathrm{v}_{k,l}$

$$\mathbf{a}_v = (a_{v,1}, a_{v,2})^T \tag{5.6}$$

where $a_{v,1}$ describes to which extent a pixel v is expected to be influenced by side 1 of the curve, and $a_{v,2}$ is equivalent for side 2 given by $a_{v,2} = 1 - a_{v,1}$. For arbitrary curve **C**, it is difficult to give a closed form of a_v. In the following, a efficient approximation of the assignment is derived.

We use $d_{k,l}$ to denote the **signed** distance between pixel $\mathrm{v}_{k,l}$ and a given curve **C**(a image). $d_{k,l}$ can be approximated by

$$d_{k,l} = \vec{n_k} \cdot (\mathbf{v}_{k,l} - C_k) \tag{5.7}$$

where $\mathbf{v}_{k,l} = \binom{x_{k,l}}{y_{k,l}}$ is the axis components of pixel $\mathrm{v}_{k,l}$, and C_k is the equivalent for point C on the curve given by $\binom{x_k}{y_k}$. Now consider that the curve is distorted by a Gaussian distribution $p(\mathbf{\Phi})$, therefore, the displacement $d_{k,l}$ is also a Gaussian distribution, $p(d_{k,l}) \sim \mathcal{N}(d_{k,l}|m_d, \sigma)$, where m_d and σ are mean and covariance of the distribution. Covariance $sigma$ holds

$$\sigma = \vec{n_k} \cdot \mathbf{J}_k \cdot \mathbf{\Sigma}_\Phi \cdot \mathbf{J}_k^T \cdot \vec{n_K}^T \tag{5.8}$$

where $\mathbf{J}_k$ is the Jacobian of curve $\mathbf{C}$. $sigma$ can be take as the uncertainty of the curve along the normal introduced by the covariance $\mathbf{\Sigma_\Phi}$. Therefore, the probability that a point lies on side 1 of the curve can be evaluated by

$$a_{v,1} = \frac{1}{2} erf(\frac{d_{k,l}}{\sqrt{2}\sigma} + 1) \tag{5.9}$$

Where $erf(\cdot)$ is the error function of a distribution. This approximation process is called **fuzzy** (or **smooth**) assignment. The accuracy of this assignment will increase as the uncertainty of curve governed by covariance $\mathbf{\Sigma}_\Phi$ decreases.

With this assignment and following the suggested rule in (hanek's paper), we now start to assign two suitable weighting functions ω_1, ω_2 to the pixels $\mathrm{v}_{k,l}$ along the normal for the two sides of the curve. the weighting functions are defined as

$$\omega_{1/2}(d_{k,l}) = C \left(\frac{a_{1/2}(d_{k,l}) - \gamma_1}{1 - \gamma_2} \right)^4 \left[e^{-d_{k,l}/(2\hat{\sigma}^2)} - e^{-\gamma_2} \right]^+ \tag{5.10}$$

where γ_1 holds 0.5 (disjoint weight assignment) and γ_2 holds 4 for the truncated Gaussian in (5). In addition, the standard deviation is chosen in order to cover the specified distance h, which yields

$$\hat{\sigma} = \max \left[\frac{h}{\sqrt{(2\gamma_2)}}, \gamma_4 \right], \sigma = \frac{1}{\gamma_3} \hat{\sigma} \tag{5.11}$$

with the two additional constants $\gamma_3 = 6$ (linear dependence between σ and $\hat{\sigma}$) and $\sigma_4 = 4$ (minimum weighting window width) introduced in (ICVS06).

In the implementation of this thesis, $2 \cdot L \cdot N_C$ distance ($d_{k,l}$), fuzzy assignment ($a_{v,1}(d_{k,l})$) and weight function ($\omega_1(d_{k,l})$, $\omega_1(d_{k,l})$) are evaluated offline and saved in a big array. Now we have restrict our analysis the region of interest in the limited area, then collect all statistic information required to learn local statistics. In the next part, we will evaluated the local statistics.

5.2.2 Learning local statistics

For simplicity, in current implementation, we only consider raw RGB statistics. With the pixel coordinates, assignment and weight function, local mean vectors $\mathbf{m}_{v,s}$ and local covariance matrices $\mathbf{\Sigma_\Phi}$ will be derived for each side $s \in 1, 2$ of the curve. We first calculate the zero, first and second order weighted moments $M^0_{k,s}(d_{k,l,s})$, $\mathbf{M}^1_{k,s}(d_{k,l,s})$ and $\mathbf{M}^2_{k,s}(d_{k,l,s})$

$$M^{(0)}_{k,s}(d_{k,l,s}) = \sum_{l=0}^{2L-1} \omega_s(d_{k,l}) \tag{5.12}$$

$$\mathbf{M}^{(1)}_{k,s}(d_{k,l,s}) = \sum_{l=0}^{2L-1} \omega_s(d_{k,l})\mathrm{I}_{k,l} \tag{5.13}$$

$$\mathbf{M}^{(2)}_{k,s}(d_{k,l,s}) = \sum_{l=0}^{2L-1} \omega_s(d_{k,l})\mathrm{I}_{k,l}\mathrm{I}^T_{k,l} \tag{5.14}$$

Where $\mathbf{I}$ is just the pixel raw RGB value, its elements' value are between 0 and 255. Then local mean vectors $\mathbf{m}_{v,s}$ and local covariance matrices $\mathbf{\Sigma_\Phi}$ are obtained by

$$\mathbf{m}_{k,s} = \frac{\mathbf{M}^{(1)}_{k,s}}{M^{(0)}_{k,s}} \tag{5.15}$$

$$\mathbf{\Sigma}_{k,s} = \frac{\mathbf{M}^{(2)}_{k,s}}{M^{(0)}_{k,s}} - \mathbf{m}_{k,s}\mathbf{m}^T_{k,s} + \kappa\mathbf{I} \tag{5.16}$$

In function (????) κI means an identity matrix scaled by κ in order to avoid numerical singularity because later it is required to calculate the inverse matrix of $\mathbf{\Sigma}_{k,s}$. In our experiments, we choose κ to be quite small, $\kappa = 0.5$. It is shown that this is efficient to avoid numerical problems in the process of iteration.

With the local mean vectors $\mathbf{m}_{v,s}$ and local covariance matrices $\mathbf{\Sigma_\Phi}$, we can compute the likelihood function $p(\mathbf{I}_{k,l}|\mathbf{m}_{v,1}, \mathbf{m}_{v,2}, \mathbf{\Sigma}_{v,1}, \mathbf{\Sigma}_{v,1})$ for each pixel $\mathrm{v}_{k,l}$. In terms of observation model, the likelihood function describes how probable the observed data set is for different settings of the parameter vector. Hence, we first establish the relation between image data $\mathrm{I}_{\mathrm{k,l}}$ and the model parameter $\mathbf{\Phi}$. Here we model the pixel value $\hat{\mathbf{m}}_{k,l}$ and $mat\hat{h}bf\Sigma_{k,l}$ for all pixels $\mathrm{v}_{\mathrm{k,l}}$ in the vicinity of curve as the linear combination of $\mathbf{m}_{v,1}$ and $\mathbf{m}_{v,2}$

$$\hat{\mathbf{m}}_{k,l} = a_{v,1}(d_{k,l})\mathbf{m}_{v,1} + a_{v,2}(d_{k,l})\mathbf{m}_{v,2} \tag{5.17}$$

If we define $mat\hat{h}bf\Sigma_{k,l}$ using the rule as $mat\hat{h}bfm_{k,l}$ resulting in a function about $d_{k,l}$, the computational cost in the procedure of parameters refinement which is discussed in next section will be dramatically high. Instead, we decide to model the covariance matrix $mat\hat{h}bf\Sigma_{k,l}$ following the rule in (????????), but we do not consider it as the function about $d_{k,l}$

$$\hat{\mathbf{\Sigma}}_{k,l} = a_{v,1}(d_{k,l})\mathbf{\Sigma}_{v,1} + a_{v,2}(d_{k,l})\mathbf{\Sigma}_{v,2} \tag{5.18}$$

Now for each observed pixel $\mathbf{I}_{k,l}$, the likelihood function is given by

$$p(\mathbf{I}_{k,l}|\mathbf{m}_{v,1}, \mathbf{m}_{v,2}, \mathbf{\Sigma}_{v,1}, \mathbf{\Sigma}_{v,1}) = p(\mathbf{I}_{k,l}|\hat{\mathbf{m}}_{k,l}, \hat{\mathbf{\Sigma}}_{k,l}) \tag{5.19}$$

However, we hope to the likelihood for all pixels in the vicinity of the curve. If we consider the coupling or other complex relation among different group of pixels, the problem will become dramatically complicated and the cost of computing will be very expensive. We can avoid these problem if we assume pixels are drawn independently from the same distribution (this is not true), namely independent and identically distributed (i.i.d)(bishop). Thus we can model the likelihood function as

$$p(\mathbf{I}_{\mathcal{V}}|\hat{\mathbf{m}}_{\mathcal{V}}, \hat{\mathbf{\Sigma}}_{\mathcal{V}}) = \prod_{l}\prod_{k} p(\mathbf{I}_{k,l}|\hat{\mathbf{m}}_{k,l}, \hat{\mathbf{\Sigma}}_{k,l}) \tag{5.20}$$

The index $\mathcal{V}$ indicates quantities for all pixels v in $\mathcal{V}$. Note we only take into account those pixels which are in the vicinity $\mathcal{V}$ of the curve. Pixels outside $\mathcal{V}$ are not considered.

We have derived the likelihood function of observed pixels. Together with the input data and prior knowledge, now we can go into parameters refinement stage by applying Bayesian's theorem.

5.3 Refine parameters

With the likelihood function in (5.20) and prior distribution in (5.1), we can estimate the $\hat{\Phi}$ using MAP which is based the Bayesian treatment. Firstly, the posterior distribution holds

$$\begin{aligned} p(\mathbf{\Phi}|\mathbf{I}_{\mathcal{V}}) \propto & p(\mathbf{I}_{\mathcal{V}}|\hat{\mathbf{m}}_{\mathcal{V}}(\Phi), \hat{\mathbf{\Sigma}}_{\mathcal{V}}(\Phi))p(\mathbf{\Phi}|\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}}) \\ = & \frac{1}{(2\pi)^{1/2}}\frac{1}{|\mathbf{\Sigma}_{\mathbf{\Phi}}|}\exp\{-\frac{1}{2}(\phi - m_{\mathbf{\Phi}})^T \mathbf{\Sigma}_{\mathbf{\Phi}}{}^{-1}(\phi - \mathbf{m}_{\mathbf{\Phi}})\}\cdot \\ & \prod_{k=0}^{N_c-1}\prod_{l=0}^{2L-1} \frac{1}{(2\pi)^{1/2}}\frac{1}{|\hat{\mathbf{\Sigma}}_{k,l}|}\exp\{-\frac{1}{2}[I_{k,l} - \hat{\mathbf{m}}_{k,l}(a_{v,1})]^T \hat{\mathbf{\Sigma}}_{k,l}^{-1} [I_{k,l} - \hat{\mathbf{m}}_{k,l}(a_{v,1})]\} \end{aligned} \tag{5.21}$$

Let's using a function $\mathcal{Q}$ to denote the logarithm function of right part in (5.21)

$$\begin{aligned} \mathcal{Q} = & -2\ln\left\{p(\mathbf{I}_{\mathcal{V}}|\hat{\mathbf{m}}_{\mathcal{V}}(\Phi), \hat{\mathbf{\Sigma}}_{\mathcal{V}}(\Phi))p(\mathbf{\Phi}|\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})\right\} \\ = & \ln(2\pi) + 2\ln|\mathbf{\Sigma}_{\mathbf{\Phi}}| + \mathbf{\Phi}^T\mathbf{\Sigma}_{\mathbf{\Phi}}{}^{-1}\mathbf{\Phi} + 2LN_C \cdot \ln 2\pi + \\ & 2\sum_{k=0}^{N_c-1}\sum_{l=0}^{2L-1}\ln|\mathbf{\Sigma}_{k,l}| + \sum_{k=0}^{N_c-1}\sum_{l=0}^{2L-1}\left\{[I_{k,l} - \hat{\mathbf{m}}_{k,l}(a_{v,1})]^T \hat{\mathbf{\Sigma}}_{k,l}^{-1} [I_{k,l} - \hat{\mathbf{m}}_{k,l}(a_{v,1})]\right\} \end{aligned} \tag{5.22}$$

We interpret the estimate $\hat{\mathbf{\Phi}}$ of the model parameters $\mathbf{\Phi}$ as the mean $\mathbf{m}_{\mathbf{\Phi}}$ of a Gaussian approximation of the posterior distribution, and $\hat{\mathbf{\Phi}}$ can be evaluated as

$$\hat{\mathbf{\Phi}} = \mathbf{m}_{\mathbf{\Phi}}\arg\max_{\mathbf{\Phi}} \mathcal{Q} \tag{5.23}$$

For the estimate $\mathbf{m}_{\mathbf{\Phi}}$ optimizing $\mathcal{Q}$, the Gauss-Newton approximation to the Hessian matrix can be adopted, first the partial derivatives of $\mathcal{Q}$ is computed as

$$\nabla_{\mathbf{\Phi}}\{\mathcal{Q}(\mathbf{\Phi})\} = 2\{\mathbf{\Sigma}_{\mathbf{\Phi}}{}^{-1}\}^T\mathbf{\Phi} - \sum_{k=0}^{N_c-1}\sum_{l=0}^{2L-1}\left\{\mathcal{J}_{a_{v,1}}^T \hat{\mathbf{\Sigma}}_{k,l}^{-1} [I_{k,l} - \hat{\mathbf{m}}_{k,l}(a_{v,1})]\right\} \tag{5.24}$$

with

$$\mathcal{J}_{a_{v,1}} = (\mathbf{m}_{k,1} - \mathbf{m}_{k,2})\,(\nabla_{\phi} a_{v,1}(d_{k,l}))^T \tag{5.25}$$

because $d_{k,l}$ is a function with respect to $\mathbf{\Phi}$, there $a_{v,1}(d_{k,l})$ holds

$$\nabla_{\mathbf{\Phi}} a_{v,1}(d_{k,l}) = \frac{\partial a_{v,1}(d_{k,l})}{\partial d_{k,l}} \left(\frac{\partial d_{k,l}}{\partial x_{k,l}} \mathbf{J}_{\mathbf{\Phi}}(\mathrm{v}_{k,l}(x)) + \frac{\partial d_{k,l}}{\partial y_{k,l}} \mathbf{J}_{\mathbf{\Phi}}(\mathrm{v}_{k,l}(y)) \right) \tag{5.26}$$

Where $\mathrm{v}_{k,l}(x)$ and $\mathrm{v}_{k,l}(y)$ are the axis components of pixel $\mathrm{v}_{k,l}$, and $d_{k,l}$ is given by

$$d_{k,l} = (x_{k,l} - x_k)n_k(x) + (y_{k,l} - y_k)n_k(y) \tag{5.27}$$

with $n_k(x)$ and $n_k(y)$ are the components of normal vector of curve point C_k. Moreover, we have

$$\nabla_{\mathbf{\Phi}} a_{v,1}(d_{k,l}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ -\frac{d_{k,l}^2}{2\sigma^2} \right\} (n_k(x)\mathbf{J}_{\mathbf{\Phi}}(\mathrm{v}_{k,l}(x)) + n_k(y)\mathbf{J}_{\mathbf{\Phi}}(\mathrm{v}_{k,l}(y))) \tag{5.28}$$

In terms of the properties of B-spline curve and planar affine model-space. The pixel coordinate of v_{kl} can be written as

$$\begin{aligned}
\mathrm{v}_{k,l}(x) &= \mathbf{U}_k^T \mathbf{A}_x \mathbf{\Phi} + \mathbf{U}_k P_0(y) + \Delta_h n_k(x) && (5.29)\\
&= \Phi_0 \sum_i^n U_{k,i} + (1+\Phi_2) \sum_i^n U_{k,i} * x_{k,i} + \Phi_5 \sum_i^n U_{k,i} y_{k,i} + \Delta_h n_k(x) && (5.30)\\
\mathrm{v}_{k,l}(y) &= \mathbf{U}_k^T \mathbf{A}_x \mathbf{\Phi} + \mathbf{U}_k P_0(y) + \Delta_h n_k(y) && (5.31)\\
&= \Phi_0 \sum_i^n U_{k,i} + (1+\Phi_2) \sum_i^n U_{k,i} * x_{k,i} + \Phi_5 \sum_i^n U_{k,i} y_{k,i} + \Delta_h n_k(x) && (5.32)
\end{aligned}$$

Now we can compute $\mathbf{J}_{\mathbf{\Phi}}$ by the following formula

$$\mathbf{J}_{\mathbf{\Phi}}(\mathrm{v}_{k,l}) = \begin{bmatrix} \frac{\partial \mathrm{v}_{k,l}(x)}{\partial \Phi_0} & \frac{\partial \mathrm{v}_{k,l}(x)}{\partial \Phi_1} & \frac{\partial \mathrm{v}_{k,l}(x)}{\partial \Phi_2} & \frac{\partial \mathrm{v}_{k,l}(x)}{\partial \Phi_3} & \frac{\partial \mathrm{v}_{k,l}(x)}{\partial \Phi_4} & \frac{\partial \mathrm{v}_{k,l}(x)}{\partial \Phi_5} \\ \frac{\partial \mathrm{v}_{k,l}(y)}{\partial \Phi_0} & \frac{\partial \mathrm{v}_{k,l}(y)}{\partial \Phi_1} & \frac{\partial \mathrm{v}_{k,l}(y)}{\partial \Phi_2} & \frac{\partial \mathrm{v}_{k,l}(y)}{\partial \Phi_3} & \frac{\partial \mathrm{v}_{k,l}(y)}{\partial \Phi_4} & \frac{\partial \mathrm{v}_{k,l}(y)}{\partial \Phi_5} \end{bmatrix} \tag{5.33}$$

Afterwords, the Gauss-Newton approximation to the Hessian matrix is given by

$$\mathcal{H}_{\mathbf{\Phi}} \mathcal{Q} = \mathbf{\Sigma}_{\mathbf{\Phi}}^{-1} + \sum_{k=0}^{N_c-1} \sum_{l=0}^{2L-1} \left\{ \mathcal{J}_{a_{v,1}}^T \hat{\mathbf{\Sigma}}_{k,l}^{-1} \mathcal{J}_{a_{v,1}} \right\} \tag{5.34}$$

The overall gradient and Hessian matrices for the optimization are obtained by adding the prior cost function derivatives, and the Newton optimization step can finally be performed as

$$\begin{aligned}
\mathbf{m}_{\mathbf{\Phi}}^{new} &= \mathbf{m}_{\mathbf{\Phi}} - (\mathcal{H}_{\mathbf{\Phi}} \mathcal{Q})^{-1} \nabla_{\mathbf{\Phi}} \mathcal{Q}\\
\mathbf{\Sigma}_{\mathbf{\Phi}}^{new} &= c\mathbf{\Sigma}_{\mathbf{\Phi}} - (1-c)(\mathcal{H}_{\mathbf{\Phi}} \mathcal{Q})^{-1}
\end{aligned} \tag{5.35}$$

with $c = \frac{1}{4}$ in our implementation, note the covariance matrix is updated as well by an exponential decay rule.

5.3.1 Validation gate

5.4 Efficiency and complexity

6 Applications in Robotics

6.1 Perception

6.2 Tracking

6.3 Refinement of 3D percepts (Point Cloud)

7 Experiments and Results

8 Conclusion and Future Work