Contracting Curve Density Algorithm for Applications in Personal Robotics

Shulei Zhu and Dejan Pangercic and Michael Beetz
Intelligent Autonomous Systems Group, TU Munich
Email: {shulei.zhu, pangercic, beetz}@cs.tum.edu

Abstract—This paper investigates an extended and optimized implementation of the state-of-the-art local curve fitting algorithm named Contracting Curve Density (CCD) algorithm, originally developed by Hanek et al. In particular, we investigate its application in the field of personal robotics for the tasks such as the mobile manipulation which requires a segmentation of objects in clutter and the tracking of them. The developed system mainly consists of the two functional parts, the CCD algorithm to fit the model curve in still images and the CCD tracker to track the model in the videos. We demonstrate algorithm’s working in various scenes using handheld camera and the cameras from the Personal Robot 2 (PR2). Achieved results show that the CCD algorithm achieves robustness and sub-pixel accuracy even in the presence of clutter, partial occlusion, and changes of illumination.

I. INTRODUCTION

The CCD algorithm can be best described as follows. Given one or multiple images as input data and a parametric curve model with a priori distribution of model parameters, through curve-fitting process, we estimate the model parameters which determine the approximation of the posterior distribution in order to make the curve models best matching the image data [1].

The curve-fitting problem and its variants have a wide range of applications in the field of robotics, medical processing, user interface, surveillance and biometrics [2]. In order to be widely applicable to practical personal robotics problems (such as perception of mobile manipulation), robustness, accuracy, efficiency and versatility should be considered when a novel approach is designed and implemented. However, solving object segmentation and the related object contour tracking problems are always challenging, especially in natural and unconstrained scenes. Due to clutter, shading, texture, and specular reflections it is very difficult to segment an object from an inhomogeneous background. Furthermore, physical conditions, such as the illumination or surface properties, will influence the efficiency and stability of related approaches. We thus aim to introduce the substantial improvements to the original version of CCD algorithm which adaptively learns background and foreground models and leverages from other techniques such as SIFT features matching or back-projection of point clouds (Fig. 1) onto the test image to robustly segment the target objects.

A. An Alternative View of the CCD Algorithm

In the field of pattern recognition, the key concept is that of uncertainty. In image data, the uncertainty arises both through

Fig. 1: Personal Robot 2 (PR2 [3]) robot segmenting the book using CCD algorithm. The initial contour of the book was generated through a 3D-based cluster segmentation and back-projection of the bounding box onto the test image. **Bottom-right:** Segmentation of a plate in a rather challenging scene.

noise from measurements, as well as through the nature of the objects (e.g. cardiac motion and deformation). Probability theory provides a consistent framework for the quantification and manipulation of uncertainty. In this section, we consider curve-fitting problem from a probabilistic perspective and turn it into a classification problem.

In the CCD algorithm, the aim is to find the contour of observed object and thus segment it from the background. Therefore, a hypothetical contour divides the image into two parts (Fig. 3), inside and outside. For probabilistic model, we can represent this using binary representation (e.g. $\{0, 1\}$). The goal of the CCD algorithm then becomes to accurately assign a class label to each pixel in the vicinity of the contour and the curve-fitting problem thus becomes a classification one. A powerful approach to solve this problem involves modeling of a conditional probability distribution in an inference stage, and then subsequently using this distribution to make optimal decisions. In order to derive the conditional probability, prior distribution and likelihood function should be given.

We assume that a parametric curve model is governed by a prior distribution over the model parameters (usually a multi-dimensional vector). There exists a range of probability

distributions which can be used to model the distribution of shapes. In this paper we consider the Gaussian distribution.

Defining the prior distribution is only a step of the problem. According to the Bayesian theorem, the conditional distribution is proportional to the product of prior distribution and the likelihood function. Hence, the next step is to define the likelihood function.

In the implementation of the CCD algorithm local image pixel values are used as the training data to determine the likelihood function. If the data is assumed to be drawn independently from the distribution, then the likelihood function is given by the accumulation of all the components. By pixel value we denote the vector containing the local single- or multichannel image data associated to a pixel (e.g. RGB values). However, other types of local features computed in a pre-processing step may also be used, e.g. texture descriptors or color values in other color spaces. The likelihood function obtained from the local statistics does not have a closed-form solution. In addition, the prior distribution is just an approximate to the true distribution. Since the maximization likelihood method was proven [4] not to work here, we have to use an alternative approach known as Iterative Reweighted Least Squares (IRLS) to find the solution. Here the IRLS process is used to find the Maximum a Posterior (MAP) estimate.

In the CCD algorithm, because we just need a parameter vector determining the shape of the specified contour, we do not plan to calculate the predictive distribution. Therefore, the MAP solution mentioned above becomes our final objective function.

B. Key Contributions

In order to improve the stability, accuracy and robustness over the original implementation we introduce the following novel improvements. Firstly, we use the logistic sigmoid function instead of a Gaussian error function which renders a curve-fitting problem as a Gaussian logistic problem¹ known in the field of pattern recognition. Secondly, a quadratic or a cubic B-spline curve is used to model the parametric curve to avoid the Runge phenomenon [5] without increasing the degree of the B-spline. Thirdly, the system supports both planar affine (6-DOF) and three-dimensional affine (8-DOF) shape-space. The latter affine space can avoid curve mismatching caused by major viewpoint changes. Lastly, in order to avoid manual intervention by the user, the developed system also supports robust global initial curve initialization modules based on both keypoint feature matching and projection of concave contours a priori designed mesh models. The herein presented work is implemented as part of ROS and freely available on www.ros.org².

In the remainder of this paper we first introduce the related approaches, in Section III we then shortly revisit an original

implementation of CCD algorithm and continue with a presentation of our improvements over the original idea (Section IV). In Section V we present the set of possible applications for CCD and evaluation thereof and conclude with Section VI.

II. RELATED WORK

We will split this section based on the following two criteria:

- related work on two-dimensional and three-dimensional deformable models;
- related work on applying statistical knowledge to the models.

A. Two-dimensional & Three-dimensional Models

Many traditional segmentation methods are effected by the assumption that the images studied in computer vision are usually self-contained, namely, the information needed for a successful segmentation can be extracted from the images. In 1980s, a paradigm named *Active Vision* [6] escaped this bind and pushed the vision process in a more goal-directed fashion. After that the *Snakes* algorithm was proposed in a seminar work conducted by Kass [7]. The original paper, spawned many variations and extensions including the use of Fourier parametrization [8], and incorporation of the topologically adaptable models [9]. A realization of the Snakes using B-splines was developed in [10]. Gradient Vector Flow (GVF) [11] is an extension developed based on a new type of external field. In [12] authors are concerned with the convergence properties of deformable models. In three dimensions, a good deal of research work has been conducted on matching three-dimensional models, both on rigid [13] and deformable [14] shapes.

B. Statistical Models

Analyzing the model fitting in a probabilistic context has two great advantages. Firstly, the ranges of shapes are defined by a probability density function and secondly, we can make use of the vast number of tools that are available in e.g. pattern recognition field.

In [15] and [16], an elegant use of statistical models for the segmentation of medical images is presented. The resulting segmentation system consists of building statistical models and automatic segmentation of new image data sets by restricting elastic deformation of models. The works in [17] and [18] also exploit the prior knowledge in that the statistical shape models enforce the prior probabilities on objects by approximating the latter with an energy function. In this paper, we assume that the shapes' priors have a Gaussian form in shape-space. In the case of a norm-squared density over quadratic spline space, the prior is a Gaussian Markov Random Field (MRF) [19], which is used widely for modeling prior distributions for curves [20].

Defining a prior distribution for shape is only a part of the problem as prior knowledge only controls the feature interpretation in the image and thus just approximates the contour of an observed object. In order to snap to the exact shape of the object a likelihood function is required. In some special cases, we can get a solution by maximizing the

¹For the two-class classification problem, the posterior probability of class C can be written as a **continuous** logistic sigmoid acting on a linear function of x (equivalent to *fuzzy assignment*).

²http://www.ros.org/wiki/contracting_curve_density_algorithm

likelihood, but usually it is intractable because there is no closed-form solution available. An indirect approach known as Iterative Reweighted Least Squares [4] is used to find the parameters of the model.

III. SYSTEM ARCHITECTURE

In this section we will briefly laid out the basic steps of the initial CCD algorithm as presented by Hanek et al [1]. The

Fig. 2: The flowchart of the basic steps of the CCD algorithm.

algorithm consists of four major steps which are also sketched in Fig. 2:

- 1) **Initialization:** Given an input image $\mathbf{I}$ as training data, we first choose an initial contour for an observed object, then model the contour using a parameter vector Φ . In addition, for most practical applications a pre-processing stage (e.g. smoothing) gets applied as well.
- 2) **Learning of local statistics:** In the step of learning of local statistics for each pixel 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 the pixel on the contour 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” [1], also known as likelihood: $p(\mathbf{I}|\Phi)$.
- 3) **Refinement of model parameters:** The conditional distribution, namely the product of a prior distribution and the likelihood, is evaluated as the cost function. Then MAP estimate is executed to optimize the parameters and as a result a MAP value of model parameter vector and covariance will be generated in this step.

- 4) **Convergence:** Check for the convergence of the log posterior probability function $\ln p(\Phi)p(\mathbf{I}|\Phi)$. If the convergence criterion is not satisfied return to step 2.

IV. CONTRACTING CURVE DENSITY ALGORITHM FOR PERSONAL ROBOTICS

In this section, we describe all important underlying principles of the CCD algorithm and its extensions that make it suitable for the use in personal robotics field. After a thorough evaluation of the original implementation we spotted the following three issues which needed to be substantially improved in order to assure the robust performance and automatic bootstrapping: i) automatic initialization, ii) curve parametrization models and iii) learning of local statistics.

A. Automatic Initialization Modes

In our implementation, we model the contour as a continuous, differentiable and uniform quadratic or cubic B-Spline in $\mathbb{R}^2$. Given a region of interest (ROI), the first step is to generate enough control points $\mathbf{P} = \{P_0, \dots, P_{N_p-1}\}$ to account for the complexity of the object shape (Fig. 3).

Fig. 3: A contour with sample points used for learning of local statistics. The blue contour divides the image into two parts: outside and inside. As depicted in right-bottom image, pixels inside the contour are labeled “0”, and those outside the contour are labeled “1”. For each point on the contour (the light-blue point), we sample points (red points) along both positive and negative direction of the normal of the contour point.

There are two ways to generate control points, manually and automatically. Since the former is rather tedious, the two new intelligent initialization methods are proposed to extract the contour. Firstly, an initial contour estimation method is described and implemented by employing the well-known Scale-Invariant Feature Transform (SIFT) algorithm [21] for keypoint detection and matching. Secondly, we make use of the a priori modeled meshes of deformable objects (such as

e.g. T-shirts from Fig. 9), extract their concave contours and use projections of those as an initial guess. Both methods are discussed below.

1) *SIFT-based Initialization*: We first extract SIFT features in the test image and match them against the template image using Nearest Neighbor Search. For filtering out of false positive matches and estimation of the homography, we use Random Sample Consensus (RANSAC) [22], which is an iterative process that randomly selects matches in the test image, back-projects them onto the template image and verifies the relative normalized difference in order to select inliers.

In order to speed-up the homography computation we use the Normalized Direct Linear Transform (normalized DLT [23]). The Normalized DLT algorithm computes a homography for a projective transformation by using at least 4 point correspondences and then minimizing the corresponding norm.

After obtaining the homography between the template image and the test image, the bounding box around the feature inliers of the object can be easily projected into the test image to obtain the estimate of the contour position. This is illustrated in Fig. 8.

2) *Mesh Model-based Initialization*: In this mode we use publicly available mesh models, such as the ones from the Google 3D Warehouse (see a mesh of the T-shirt in top row of Fig. 9). In order to generate the initial contour we first sample vertex points and convert the meshes into point clouds. Next we compute the 3D concave hull of the resulting point cloud using a delaunay triangulation algorithm and thus obtain an approximate object shape identifier. In order to account for multi-scale of images we build the pyramid of different hull sizes normalized between the $[1/2 \dots 1.0]$ of the image width. Finally we project all contour sizes onto the test image and check the convergence criterion. Fig. 9 depict a successful segmentation of a T-shirt which enables the robot to detect the article and e.g. folds it.

B. Curve Parametrization

Given the control points, by applying the (uniform quadratic or cubic) B-spline interpolation, a new curve grouped by a sequence of equidistant distributed points is generated. The B-spline curve is composed of a sequence of points $\mathbf{C} = \{C_0, \dots, C_k\}$, $k = N_C - 1$. N_C denotes the number of sample points in the parametric curve. N_C is significant for the performance of the CCD algorithm, because its value is directly proportional to the circumference of the observed object. For a high resolution image, more sample points should be taken into account. Hence, there is a trade-off between the computational expense and the accuracy. Sampling more points near spinodals and corners is thus crucial for the successful operation of the algorithm.

Since the resulting parametric curve is continuous and differentiable, we can easily compute the normal vector $\mathbf{n} = \{n_0, \dots, n_k\}$ and the tangent vector $\mathbf{t} = \{t_0, \dots, t_k\}$ for every contour points (see Fig. 3).

In the planar affine shape-space $\mathcal{S}$, the contour can be compactly represented using a vector Φ with 6 real elements, namely a model parameter vector. In order to model the whole family of possible curves (Fig. 4) let us first introduce a Gaussian Probability Density Function that is used for modeling of these possible curves in shape-space $\mathcal{S}$:

$$p(\Phi) \propto \exp \left\{ -\frac{1}{2} (\Phi - \mathbf{m}_\Phi)^T \Sigma_\Phi^{-1} (\Phi - \mathbf{m}_\Phi) \right\} \quad . \quad (1)$$

Fig. 4: The top-row figure represents the mean shape curve, the bottom-left figure represents the euclidean similarities and the bottom-right figure some samples in affine space. All these are governed by a Gaussian distribution in shape-space with root-mean-square displacement of 0.3 length units.

In two dimensions, the current mean model parameter $\mathbf{m}_\Phi$ is a vector with 6 elements, and current covariance matrix Σ_Φ is a 6×6 matrix, which measures the variability of how much two groups of model parameters change together. The information matrix Σ_Φ^{-1} can be written as:

$$\Sigma_\Phi^{-1} = \frac{N_\Phi}{\rho_0^2} \mathbf{A}^T \mathcal{U} \mathbf{A} \quad , \quad (2)$$

where ρ_0^2 denotes the mean-square displacement along the entire curve ([19]). $\mathbf{A}$ is the shape-matrix, and $\mathcal{U}$ is a metric matrix for curves. N_Φ represents the number of model parameters. Note that ρ_0^2 is a real value and can be computed as:

$$\rho_0^2 = \text{tr}(\Sigma_\Phi) \quad , \quad (3)$$

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

The parametric curve model using uniform quadratic or cubic B-spline has thus been set up. In the following subsection the logistic sigmoid function for the purpose of the local statistics learning will be discussed.

C. Learning of Local Statistics

In order to learn local statistics (raw RGB pixel values), we first have to define the search interval h on both sides of the curve:

$$h = \sqrt{2}\rho_0 = \sqrt{\text{tr}(\Sigma_{\Phi} \mathbf{A}^T \mathbf{U} \mathbf{A})} \quad . \quad (4)$$

h thus denotes the size of a *window* which is used for computing the local statistics. In the beginning of the iterative procedure, the value is relatively big and only roughly approximates the vicinity of the image curve. The uncertainty is reduced after further iteration steps and as a result, the h becomes smaller and smaller. After determining the length of the search segment, a set of points located on these segments can be collected and evaluated. Note that the parametric model curve is not required to be closed, but it shall always encompass a limited area. We only analyze the pixels located in the vicinity of the contour (red sample pixels in Fig. 3). Therefore, we should limit the search distance on the *inner* side in order to avoid crossing the opposite boundary to sample pixels from the outer area [24]. On the other hand we should avoid collecting statistics from too few pixels, because it is statistically invalid and we can not capture all features (e.g. spinodals and corners) of the contour. Let us denote the sample distance using Δh , then the overall number of spaced sample points L ($2L$ for both sides in all) can be given by:

$$L = \lfloor \frac{h}{\Delta h} \rfloor \quad . \quad (5)$$

Note that the goal of the algorithm is to assign each pixel ($v_{k,l}$, $k \in [0, \dots, N_C - 1]$, $l \in [0, 2L - 1]$) on either side of the contour. By doing so we thus run into a classification problem. Although each pixel should be assigned to one and only one class so that the target variable is discrete, we can model the posterior probabilities that lie in $(0, 1)$ interval, which converts the classification into the regression problem. To achieve the latter, we model the probabilistic assignments $\mathbf{a}_v$ for each pixel $v_{k,l}$ as following:

$$\mathbf{a}_v = (a_{v,1}, a_{v,2})^T \quad , \quad (6)$$

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 $a_{v,2} = 1 - a_{v,1}$.

In order to apply the probabilistic generative models to the above classification problem, we need to transform the linear function of Φ using a non-linear activation function $f(\cdot)$ [4]:

$$a_{v,1} = f\left(\frac{d_{k,l}}{\sigma}\right) \quad , \quad (7)$$

where σ can be taken as the uncertainty of the curve along the normal introduced by the covariance Σ_{Φ} . $d_{k,l}$ is the distance between pixel $v_{k,l}$ and a given curve. Unlike in the original CCD implementation we use a non-linear activation functions

Fig. 5: **Left:** Logistic sigmoid function, **Right:** Probit function from original CCD.

Fig. 6: Classification problem: In the the presence of outliers, the probit activation function (violet line) has a higher misclassification rate.

called *logistic sigmoid* function given by (and depicted in Fig. 5):

$$a_{v,1} = \frac{1}{1 + \exp(\frac{d_{k,l}}{\sqrt{2}\sigma})} \quad . \quad (8)$$

The term *sigmoid* stands for S-shaped [4].

The logistic sigmoid function can be used to transform a broad range of class-conditional distributions, described by the exponential family, to a non-linear posterior class probabilities. The edge that the logistic sigmoid function gives over the probit function is in that it is less sensitive to outliers because the logistic sigmoid decays asymptotically like $\exp(-x)$ for $x \rightarrow \infty$, whereas for the probit activation function the decay is like $\exp(-x^2)$ (Fig. 6).

With this assignment and following the suggested rule in [1], we now start to assign two suitable weighting functions ω_1 , ω_2 to the pixels $v_{k,l}$ along the normal for the two sides of the curve. The weighting functions are defined as:

$$\omega_{1/2}(d_{k,l}) = \mathcal{C} \left(\frac{a_{1/2}(d_{k,l}) - \gamma_1}{1 - \gamma_2} \right)^4 \left[e^{-d_{k,l}/(2\hat{\sigma}^2)} - e^{-\gamma_2} \right]^+ \quad , \quad (9)$$

where $\mathcal{C}$ denotes the normalization constant, γ_1 equals to 0.5 (disjoint weight assignment) and γ_2 equals to 4 for the truncated Gaussian [1],

In addition, the standard deviation is chosen in order to cover the specified distance h , which yields

$$\hat{\sigma} = \max \left[\frac{h}{\sqrt{2\gamma_2}}, \gamma_4 \right], \sigma = \frac{1}{\gamma_3} \hat{\sigma} \quad , \quad (10)$$

with the two additional constants $\gamma_3 = 6$ (linear dependence

between σ and $\hat{\sigma}$) and $\gamma_4 = 4$ (minimum weighting window width) as discussed in [25].

In the implementation of this work, there are $2 \cdot L \cdot N_C$ distances, fuzzy assignments and weight functions which leads to as many weight functions being evaluated offline and stored in an array. Now we have restricted our analysis region of interest in the limited area, and collected all statistical information required to learn local statistics. In the next part, we will evaluate the local statistics.

Given the pixel coordinates and their RGB values, assignment and weight function, local mean vectors $\mathbf{m}_{v,s}$ and local covariance matrix Σ_{Φ} will be derived for each side $s \in \{1, 2\}$ of the curve. We first calculate the zero, first and second order weighted moments:

$$M_{k,s}^{(0)}(d_{k,l,s}) = \sum_{l=0}^{2L-1} \omega_s(d_{k,l}) \quad (11)$$

$$\mathbf{M}_{k,s}^{(1)}(d_{k,l,s}) = \sum_{l=0}^{2L-1} \omega_s(d_{k,l}) \mathbf{I}_{k,l} \quad (12)$$

$$\mathbf{M}_{k,s}^{(2)}(d_{k,l,s}) = \sum_{l=0}^{2L-1} \omega_s(d_{k,l}) \mathbf{I}_{k,l} \mathbf{I}_{k,l}^T, \quad (13)$$

where $\mathbf{I}_{k,l}$ is just the raw RGB value of a pixel, for a 3-channel image and the values of three components are between 0 and 255. The local mean vectors $\mathbf{m}_{v,s}$ and the local covariance matrices Σ_{Φ} are obtained by:

$$\mathbf{m}_{k,s} = \frac{\mathbf{M}_{k,s}^{(1)}}{M_{k,s}^{(0)}} \quad (14)$$

$$\Sigma_{k,s} = \frac{\mathbf{M}_{k,s}^{(2)}}{M_{k,s}^{(0)}} - \mathbf{m}_{k,s} \mathbf{m}_{k,s}^T + \kappa \mathbf{I} \quad (15)$$

In Eq. 15, $\kappa \mathbf{I}$ means an identity matrix scaled by κ in order to avoid numerical singularity. Later it is namely required to calculate the inverse matrix of $\Sigma_{k,s}$. In our experiments, we choose κ to be $\kappa = 0.5$, which is efficient to avoid numerical problems in the process of iteration.

With the local mean vectors $\mathbf{m}_{v,s}$ and the local covariance matrices $\Sigma_{k,s}$, we can compute the likelihood function $p(\mathbf{I}_{k,l} | \mathbf{m}_{v,1}, \mathbf{m}_{v,2}, \Sigma_{v,1}, \Sigma_{v,2})$ for each pixel $v_{k,l}$. 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 $\mathbf{I}_{k,l}$ and the model parameter Φ . Here we model the pixel value $\hat{\mathbf{m}}_{k,l}$ and its covariance $\hat{\Sigma}_{k,l}$ for all pixels $v_{k,l}$ in the vicinity of the curve as the linear combination of $\mathbf{m}_{v,1}$ and $\mathbf{m}_{v,2}$:

$$\hat{\mathbf{m}}_{k,l} = a_{v,1}(d_{k,l}) \mathbf{m}_{v,1} + a_{v,2}(d_{k,l}) \mathbf{m}_{v,2} \quad (16)$$

To prevent high computational costs we model the covariance matrix $\hat{\Sigma}_{k,l}$ following the rule in [2], but we discard the following relation:

$$\hat{\Sigma}_{k,l} = a_{v,1}(d_{k,l}) \Sigma_{v,1} + a_{v,2}(d_{k,l}) \Sigma_{v,2} \quad (17)$$

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

$$p(\mathbf{I}_{k,l} | \mathbf{m}_{v,1}, \mathbf{m}_{v,2}, \Sigma_{v,1}, \Sigma_{v,2}) = p(\mathbf{I}_{k,l} | \hat{\mathbf{m}}_{k,l}, \hat{\Sigma}_{k,l}) \quad (18)$$

However, we require 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 intractable and the cost of computing will be very expensive. We can avoid these problems if we assume pixels to be drawn independently from the same distribution, namely independent and identically distributed (i.i.d) [4]. Thus we can model the likelihood function as:

$$p(\mathbf{I}_{\mathcal{V}} | \hat{\mathbf{m}}_{\mathcal{V}}, \hat{\Sigma}_{\mathcal{V}}) = \prod_l \prod_k p(\mathbf{I}_{k,l} | \hat{\mathbf{m}}_{k,l}, \hat{\Sigma}_{k,l}) \quad (19)$$

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

Having the likelihood function of observed pixels, as well as the input data and prior knowledge, now we can go into the parameters refinement stage to model the conditional distribution.

D. Refinement of Parameters

In this section, an Iterative Reweighted Least Square (IRLS) process is presented in order to refine the model parameter vector and the covariance matrix.

With the likelihood function in Eq. 19 and prior distribution in Eq. 1, we can model the conditional possibility distribution using:

$$p(\Phi | \mathbf{I}_{\mathcal{V}}) \propto p(\mathbf{I}_{\mathcal{V}} | \hat{\mathbf{m}}_{\mathcal{V}}(\Phi), \hat{\Sigma}_{\mathcal{V}}(\Phi)) p(\Phi | \mathbf{m}_{\Phi}, \Sigma_{\Phi}) \quad (20)$$

The difference between this conditional distribution and posterior distribution is that here the former function has not been normalized yet. If applying logarithm operation to the conditional possibility distribution, we can define a common cost function $\mathcal{Q}$, which is given by:

$$\mathcal{Q} = -2 \ln \left\{ p(\mathbf{I}_{\mathcal{V}} | \hat{\mathbf{m}}_{\mathcal{V}}(\Phi), \hat{\Sigma}_{\mathcal{V}}(\Phi)) p(\Phi | \mathbf{m}_{\Phi}, \Sigma_{\Phi}) \right\} \quad (21)$$

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

$$\hat{\Phi} = \arg \max_{\Phi} (\mathcal{Q}) \quad (22)$$

To optimize $\mathcal{Q}$ based on the estimated $\mathbf{m}_{\Phi}$, the approximation to the Hessian matrix can be adopted. First we compute the partial derivatives of $\mathcal{Q}$:

$$\begin{aligned} \nabla_{\Phi} \{ \mathcal{Q}(\Phi) \} &= 2 \{ \Sigma_{\Phi}^{-1} \}^T \Phi \\ &- \sum_{k=0}^{N_C-1} \sum_{l=0}^{2L-1} \left\{ \mathcal{J}_{a_{v,1}}^T \hat{\Sigma}_{k,l}^{-1} [\mathbf{I}_{k,l} - \hat{\mathbf{m}}_{k,l}(a_{v,1})] \right\} \end{aligned} \quad (23)$$

with:

$$\mathcal{J}_{a_{v,1}} = (\mathbf{m}_{k,1} - \mathbf{m}_{k,2}) (\nabla_{\Phi} a_{v,1}(d_{k,l}))^T \quad (24)$$

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

$$\mathcal{H}_{\Phi} \mathcal{Q} = \Sigma_{\Phi}^{-1} + \sum_{k=0}^{N_C-1} \sum_{l=0}^{2L-1} \left\{ \mathcal{J}_{a_{v,1}}^T \hat{\Sigma}_{k,l}^{-1} \mathcal{J}_{a_{v,1}} \right\} \quad (25)$$

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 formulated as:

$$\begin{aligned} \mathbf{m}_{\Phi}^{new} &= \mathbf{m}_{\Phi} - (\mathcal{H}_{\Phi} \mathcal{Q})^{-1} \nabla_{\Phi} \mathcal{Q} \\ \Sigma_{\Phi}^{new} &= c \Sigma_{\Phi} - (1 - c) (\mathcal{H}_{\Phi} \mathcal{Q})^{-1} \end{aligned} \quad (26)$$

c is empirically set to $\frac{1}{4}$. Note that the covariance matrix is updated by an exponential decay rule as well. Coefficient c specifies the maximum decrease of the covariance within one iteration step [1]. If c is very large, due to the slow reduction of covariance the convergence process will be very slow. On the other hand if c is very small, CCD might diverge.

V. APPLICATIONS AND EXPERIMENTAL RESULTS

To evaluate the applicability of the herein presented version of the CCD algorithm we ran three realistic tests in our kitchen lab on the sensor data from the PR2 robot. The tests were bootstrapped using parameters from the Table I and are detailed below.

experiments	γ_1	γ_2	γ_3	γ_4	α	κ	c	h	Δh	samples	degrees	dims
Manual	0.5	5	7	5	1.2	0.5	0.25	40	2	60	4	6
SIFT	0.5	5	7	5	1.2	0.5	0.25	48	1	80	4	6
Model-based	0.5	5	7	5	1.2	0.5	0.25	50	1	200	4	6

TABLE I: initialization parameters

A. Manual Initialization

In this experiment a partially occluded book was placed in a clutter. The control points of the initial contour were drawn manually. If the contour snaps to the book within the tolerance limit the test is deemed successful. The test image is depicted in Fig. 7 in different iteration steps. We ran the CCD algorithm with 40 different initializations and the convergence failed in two cases. The average convergence time was 3.35s, see also Table II.

B. SIFT-based Initialization

In the second experiments the initial contour was generated from the SIFT features matching a priori learned template of a book shown in Fig. 8. We ran the CCD algorithm 100 times which resulted in 9 convergence failures. The average convergence time was 3.52s, see also Table II for the rest of the quantitative results. Video of the test is available online ³.

³<http://www.youtube.com/watch?v=X2K-h4oHxig>

Fig. 7: Segmentation of a book after manual initialization of the contour.

Fig. 8: Segmentation of a book using SIFT-based initialization.

C. Mesh Model-based Initialization

We used CCD in order to detect and localize a spread-out T-shirt in the application of robotic folding. We ran the algorithm 100 times where we substantially varied the pose of the initially projected T-shirt hull. The algorithm failed 4 times and took on average 10.15 seconds to converge. While the failed cases are mostly due to the too large initial curve displacement, the run time got increased with the number of the control points. See also Table II.

D. CCD-based Tracker

A CCD tracker is obtained by applying the CCD algorithm to each frame of a sequence of images independently where the

Fig. 9: **Top row:** Mesh of the T-shirt. **Middle and Bottom Row:** Detection of the T-shirt using concave hull of above mesh for the application of robotic cloth folding. Image courtesy of Christian Bersch.

output of the parameters obtained from the previous frame is used in the current frame. Because of the dependency between the two successive images, the CCD tracker can achieve a high performance rate and be applied to the time critical problems. The video ⁴ demonstrates the tracking of a book on a cart using 5M pixel camera on a PR2 robot.

Experiments	Failure rate	Run time	Iteration	Tolerance
Manual initialization	5%	3.35	26	0.001
SIFT initialization	9%	3.52	5	0.001
Model-based	4%	10.15	28	0.001

TABLE II: Experimental Results: Failure rate is the ratio between failures and all test cases. Run time is the average run time of all successful test cases, which characterizes the computational cost. The tolerance is the converge criteria defined as the curve-displacement between two successive iterations.

VI. CONCLUSIONS AND FUTURE WORK

We presented various extensions to the existing contour fitting algorithm CCD which makes it applicable to solve

various tasks, such as e.g. object segmentation, needed in the mobile manipulation. The algorithm is robust to clutter, partial occlusions and even changes of illumination. In the future we plan to work on CUDA CCD implementation to make the convergence faster and use other features than RGB color values in order to improve the stability even further.

ACKNOWLEDGMENT

This work was supported by the DFG cluster of excellence CoTeSys (Cognition for Technical Systems).

REFERENCES

- [1] R. Hanek and M. Beetz, "The contracting curve density algorithm: Fitting parametric curve models to images using local self-adapting separation criteria," *International Journal of Computer Vision*, vol. 59, no. 3, pp. 233–258, 2004.
- [2] R. Hanek, "Fitting parametric curve models to images using local self-adapting separation criteria," Ph.D. dissertation, Technische Universität München, Universitätsbibliothek, 2004.
- [3] K. Wyrobek, E. Berger, H. M. V. der Loos, and K. Salisbury, "Towards a Personal Robotics Development Platform: Rationale and Design of an Intrinsically Safe Personal Robot," in *Proc. International Conference on Robotics and Automation (ICRA)*, 2008.
- [4] C. Bishop and S. O. service), *Pattern recognition and machine learning*. Springer New York, 2006, vol. 4.
- [5] E. Süli and D. Mayers, *An introduction to numerical analysis*. Cambridge Univ Pr, 2003.
- [6] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 333–356, 1988.
- [7] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International journal of computer vision*, vol. 1, no. 4, pp. 321–331, 1988.
- [8] G. Scott, "The alternative snake—and other animals," in *Proceedings 3rd Alvey Vision Conference (Cambridge)*, 1987, pp. 341–347.
- [9] T. McInerney and D. Terzopoulos, "Topologically adaptable snakes," in *iccv*. Published by the IEEE Computer Society, 1995, p. 840.
- [10] P. Brigger, J. Hoeg, and M. Unser, "B-spline snakes: A flexible tool for parametric contour detection," *Image Processing, IEEE Transactions on*, vol. 9, no. 9, pp. 1484–1496, 2000.
- [11] C. Xu and J. Prince, "Snakes, shapes, and gradient vector flow," *Image Processing, IEEE Transactions on*, vol. 7, no. 3, pp. 359–369, 1998.
- [12] C. Xu and J. L. Prince, "Gradient vector flow deformable models," *Handbook of Medical Imaging, Academic Press*, pp. 159–170, 2000.
- [13] C. Harris, "Tracking with rigid models," in *Active vision*. MIT Press, 1993, pp. 59–73.
- [14] D. Terzopoulos and D. Metaxas, "Dynamic 3D models with local and global deformations: Deformable superquadrics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 703–714, 1991.
- [15] A. Kelemen, G. Székely, and G. Gerig, "Three-dimensional model-based segmentation of brain MRI," in *Biomedical Image Analysis, 1998. Proceedings. Workshop on*. IEEE, 1999, pp. 4–13.
- [16] A. Kelemen, G. Székely, and G. Gerig, "Elastic model-based segmentation of 3-D neuroradiological data sets," *Medical Imaging, IEEE Transactions on*, vol. 18, no. 10, pp. 828–839, 1999.
- [17] S. Sclaroff and L. Liu, "Deformable shape detection and description via model-based region grouping," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 5, pp. 475–489, 2001.
- [18] L. Liu and S. Sclaroff, "Deformable shape detection and description via model-based region grouping," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on*, vol. 2. IEEE, 1999.
- [19] A. Blake, M. Isard, et al., *Active contours*. Springer London, 1998, vol. 2.
- [20] G. Storvik, "A Bayesian approach to dynamic contours through stochastic sampling and simulated annealing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 976–986, 1994.
- [21] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

⁴<http://www.youtube.com/watch?v=nr83zqQ6CCg>

- [22] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [23] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge Univ Pr, 2003.
- [24] G. Panin and A. Knoll, "Fully automatic real-time 3d object tracking using active contour and appearance models," *Journal of Multimedia 2006*, vol. 1, no. 7, pp. 62–70, 2006.
- [25] G. Panin, A. Ladikos, and A. Knoll, "An efficient and robust real-time contour tracking system," 2006.