Proceedings of the 2002 IEEE/RSJ
Intl. Conference on Intelligent Robots and Systems
EPFL, Lausanne, Switzerland • October 2002

Fast Image-based Object Localization in Natural Scenes

Robert Hanek, Thorsten Schmitt, Sebastian Buck, Michael Beetz
TU München, Institut für Informatik, Boltzmannstr. 3, 85748 Garching b. München, Germany
http://www9.in.tum.de/agilo/

Abstract

In many robot applications, autonomous robots must be capable of localizing the objects they are to manipulate. In this paper we address the object localization problem by fitting a parametric curve model to the object contour in the image. The initial prior of the object pose is iteratively refined to the posterior distribution by optimizing the separation of the object and the background. The local separation criteria are based on local statistics which are iteratively computed from the object and the background region. No prior knowledge on color distributions is needed. Experiments show that the method is capable of localizing objects in a cluttered and textured scene even under strong variations of illumination. The method is able to localize a soccer ball within frame rate.

1. Introduction

In many robot applications, autonomous robots must be capable of localizing the objects they are to manipulate. Robot soccer provides a good case in point. Autonomous soccer robots must at least localize the ball and opponent robots. In all RoboCup leagues image-based object localization is currently considerably simplified by two restrictions: 1. all objects on the pitch have a distinctive color. 2. the illumination is constant and roughly homogeneous. Due to these restrictions, classes of objects (e.g. robots, ball, lines, color markers, goals) can roughly be identified by color classification. To the best of our knowledge, all robot soccer teams participating in RoboCup use color classification [30, 33, 14, 16, 6, 3]. However, color classification is usually not feasible in a natural scene. In this paper we consider a natural scene to be a scene where the colors of objects and the illumination are not restricted and where texture, clutter, shading, and specularity complicate image interpretation.

Especially in natural scenes, knowledge of the object shape is very helpful and sometimes necessary. Parametric curve models also known as deformable models, snakes or active contours [17], have been proven as powerful means for incorporating shape knowledge into computer vision algorithms. For example, in order to segment a bone in a medical image or in order to visually track a person, models describing the possible contours of the objects of interest are used [25, 19, 5]. The parameters of the models specify object properties such as the pose, size, and shape. By fitting parametric curve models to the image data problems like self-localization and localization of other objects can be addressed [11].

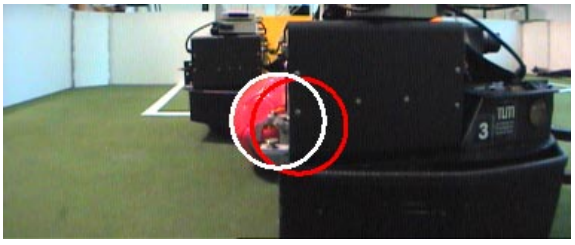


Figure 1: The proposed method correctly localizes the ball despite the partial occlusion. (red: initialization, white: estimated ball contour)

In this paper we propose a novel and fast method for fitting parametric curve models to image data. We estimate the relative position of objects (soccer ball, mug) observed by an autonomous mobile robot. We think that this work could be an important contribution towards the goal of playing robot soccer in a natural environment.

1.1 Related Work

The body of related work can be roughly classified into three categories: (i) **edge-based methods**, (ii) **region-based methods**, and (iii) **methods integrating edge-based and region-based criteria**.

(i) **Edge-based methods** rely on discontinuities of image data. Methods for different edge-profiles, i.e. types of discontinuities, exist (e.g. step-edge [2, 27, 7], roof-edge [2, 27], others [2, 27]). The problem of edge-based methods is that in practice usually the edge-profile is not known. Furthermore, the profile often varies heavily along the edge caused by e.g. clutter, shading, and texture. Due to these difficulties, usually a simple step-edge is assumed and the edge detection is performed based on a maximum image gradient. In Fig. 5a the color values on either side of the object contour are not constant even within a small vicinity. Hence, methods maximizing the image gradient have difficulties to separate the mug and the background.

(ii) **Region-based methods** such as [35, 9] rely on the homogeneity of spatially localized features (e.g. RGB values). The underlying homogeneity assumption is that the features of all pixels within one region are statistically independently distributed according to the same probability density function. However often this assumption does not

hold. In Fig. 5a the distributions of the RGB values of the foreground and the background depend on the locations within the image.

(iii) **Integrating methods** aim to overcome the individual shortcomings of edge-based and region-based approaches by integrating both approaches [31, 28, 8, 15]. These methods seek a tradeoff between an edge-based criterion, e.g. the magnitude of the image gradient, and a region-based criterion evaluating the homogeneity of the regions. However, it is questionable whether a tradeoff between the two criteria yields reasonable results when both the homogeneity assumption and the assumption regarding the edge profile do not hold as in Fig. 5a.

Model-based methods optimize the fit between the model and the image data. **Global optimization** methods like dynamic programming [1] and Monte Carlo optimization (particle filters, condensation algorithm [5]) are very successfully used (e.g., for tracking). However, dynamic programming requires a discretization of the search space, which leads to a limited accuracy, and particle filters show a very slow convergence if the sensor noise is low [32].

Local optimization methods may achieve a fast, i.e. quadratic, convergence. Approaches aiming to increase the area of convergence such as [21, 34] are edge-based. For methods maximizing the gradient, the area of convergence depends on the window size used to compute the spatial derivatives. Scale-space theory provides means for automatic scale selection [20]. However, blurring the image data eliminates useful high frequency information. Several segmentation methods integrate different image cues such as texture and color or brightness [4, 22, 23, 31].

The Contracting Curve Density (CCD) algorithm [12] does not assume a fixed separation criterion but learns from the model and the image data how to separate locally adjacent regions. By simultaneously fitting a distribution of curve to the image a big area of convergence and a fast optimization is achieved.

1.2 Main Contributions

The CCD algorithm as proposed in [12] achieves high robustness and accuracy in natural scenes. However, the method as proposed in [12] is not optimized for the speed requirements of a mobile robot. Here we present a fast version of the CCD algorithm which can be used for real robot control. While in the original version all pixels in the vicinity of the curve are used here we use just a few carefully selected pixels. Furthermore, we utilize some fast approximations. The modifications proposed in this paper lead to a speed-up of usually much more than 100, depending on the image resolution. This makes real-time object tracking based on the CCD algorithm feasible.

Overview of the paper: the remainder of this paper is organized as follows: in section 2 an overview of the Contracting Curve Density (CCD) algorithm is given. Sections 3 and 4 describe how the two main steps of the CCD algorithm can be performed efficiently. In section 5 the ball contour is modeled as a radially distorted projection of a sphere. Section 6 contains an experimental evaluation and finally in section 7 a conclusion is given.

2. Overview of the Contracting Curve Density (CCD) Algorithm

The CCD algorithm estimates the parameters of curve models from image data. The CCD algorithm can roughly be characterized as an extension of the EM algorithm [10] using additional knowledge. The additional knowledge consists of: (i) a curve model, which describes the set of possible boundaries between adjacent regions, and (ii) a model of the imaging process. The CCD algorithm, depicted in Fig. 2, performs the iteration of two steps, which roughly correspond to the two steps of the EM algorithm:

1. Local statistics of image data are learned from the vicinity of the curve. These statistics locally characterize the two sides of the edge curve. **2.** From these statistics the **estimation of the model parameters is refined** by optimizing the separation of the two sides. In the next iteration step this refinement in turn leads in to an improved statistical characterization of the two sides. During the process the uncertainty of the model parameters decreases. Thereby, the probability density of the curve in the image contracts to a single edge estimate. Therefore, we call the algorithm Contracting Curve Density (CCD) algorithm.

Input: the input of the CCD algorithm consists of the image data $\mathbf{I}^*$ and the curve model. The image data are local features, e.g. RGB values, given for each pixel of the image. The curve model consists of two parts: 1.) a differentiable curve function $\mathbf{c}$ describing the model edge curve in the image as a function of the model parameters $\mathbf{\Phi}$. The model parameters are the parameters to be estimated, for example the 3-D pose parameters of an object, 2.) a Gaussian a priori distribution $p(\mathbf{\Phi}) = p(\mathbf{\Phi} \mid \mathbf{m}^*_{\mathbf{\Phi}}, \mathbf{\Sigma}^*_{\mathbf{\Phi}})$ of the model parameters $\mathbf{\Phi}$, defined by the mean $\mathbf{m}^*_{\mathbf{\Phi}}$ and the covariance $\mathbf{\Sigma}^*_{\mathbf{\Phi}}$. (The superscript $*$ indicates input data.) Depending on the application the quantities $\mathbf{m}^*_{\mathbf{\Phi}}$ and $\mathbf{\Sigma}^*_{\mathbf{\Phi}}$ may be obtained for example by Monte Carlo optimization, by a human initialization, or from prediction over time.

Output: the output of the algorithm consists of the estimate $\mathbf{m}_{\mathbf{\Phi}}$ of the model parameters $\mathbf{\Phi}$ and the covariance $\mathbf{\Sigma}_{\mathbf{\Phi}}$ describing the uncertainty of the estimate. The estimate $\mathbf{m}_{\mathbf{\Phi}}$ and the covariance $\mathbf{\Sigma}_{\mathbf{\Phi}}$ define a Gaussian approximation $p(\mathbf{\Phi} \mid \mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ of the posterior density $p(\mathbf{\Phi} \mid \mathbf{I}^*)$.

Initialization: the estimate $\mathbf{m}_{\mathbf{\Phi}}$ of the model parameters and the associated covariance $\mathbf{\Sigma}_{\mathbf{\Phi}}$ are initialized using the mean $\mathbf{m}^*_{\mathbf{\Phi}}$ and covariance $\mathbf{\Sigma}^*_{\mathbf{\Phi}}$ of the a priori distribution. The factor c_1, e.g. $c_1 = 9$, increases the initial uncertainty and thereby enlarges the capture range of the CCD algorithm. The following two sections describe the two basic steps of the fast version of the CCD algorithm.

Input: image data $\mathbf{I}^*$, differentiable curve function $\mathbf{c}$, mean $\mathbf{m}_{\mathbf{\Phi}}^*$ and covariance $\mathbf{\Sigma}_{\mathbf{\Phi}}^*$
Output: estimate $\mathbf{m}_{\mathbf{\Phi}}$ of model parameters and associated covariance $\mathbf{\Sigma}_{\mathbf{\Phi}}$

Initialization: mean $\mathbf{m}_{\mathbf{\Phi}} = \mathbf{m}_{\mathbf{\Phi}}^*$, covariance $\mathbf{\Sigma}_{\mathbf{\Phi}} = c_1 \cdot \mathbf{\Sigma}_{\mathbf{\Phi}}^*$
repeat

1. **learn local statistics** of image data from the vicinity of the curve
   (a) compute pixels v in vicinity $\mathcal{V}$ of the image curve from $\mathbf{c}$, $\mathbf{m}_{\mathbf{\Phi}}$ and $\mathbf{\Sigma}_{\mathbf{\Phi}}$
   $\forall v \in \mathcal{V}$ compute vague assignment $\mathbf{a}_v(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ to the sides of the curve
   (b) $\forall v \in \mathcal{V}$ compute local statistics $\mathbf{S}_v$ of image data $\mathbf{I}_{\mathcal{V}}^*$
2. **refine estimation** of model parameters
   (a) update mean $\mathbf{m}_{\mathbf{\Phi}}$ by performing one iteration step of MAP estimation:

$$\mathbf{m}_{\mathbf{\Phi}} = \arg\min_{\mathbf{m}_{\mathbf{\Phi}}} \chi^2(\mathbf{m}_{\mathbf{\Phi}}) \quad \text{with}$$

$$\chi^2(\mathbf{m}_{\mathbf{\Phi}}) = -2\ln[p(\mathbf{I}_{\mathcal{V}} = \mathbf{I}_{\mathcal{V}}^* \mid \mathbf{a}_{\mathcal{V}}(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}}), \mathbf{S}_{\mathcal{V}}) \cdot \cdot p(\mathbf{m}_{\mathbf{\Phi}} \mid \mathbf{m}_{\mathbf{\Phi}}^*, \mathbf{\Sigma}_{\mathbf{\Phi}}^*)]$$

   (b) updated covariance $\mathbf{\Sigma}_{\mathbf{\Phi}}$ from Hessian of $\chi^2(\mathbf{m}_{\mathbf{\Phi}})$

until changes of $\mathbf{m}_{\mathbf{\Phi}}$ and $\mathbf{\Sigma}_{\mathbf{\Phi}}$ are small enough
Post-processing: estimate covariance $\mathbf{\Sigma}_{\mathbf{\Phi}}$ from Hessian of $\chi^2(\mathbf{m}_{\mathbf{\Phi}})$
return mean $\mathbf{m}_{\mathbf{\Phi}}$ and covariance $\mathbf{\Sigma}_{\mathbf{\Phi}}$

Figure 2: The CCD algorithm iteratively refines a Gaussian a priori density $p(\mathbf{\Phi}) = p(\mathbf{\Phi} \mid \mathbf{m}_{\mathbf{\Phi}}^*, \mathbf{\Sigma}_{\mathbf{\Phi}}^*)$ of model parameters to a Gaussian approximation $p(\mathbf{\Phi} \mid \mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ of the posterior density $p(\mathbf{\Phi} \mid \mathbf{I}^*)$.

3. Learn Local Statistics (Step 1)

In section 3.1 we describe how the pixels within the vicinity of the expected image curve are determined. For the resulting pixels local statistics of the image data are derived in section 3.2.

3.1 Determine Pixels in the Vicinity of the Curve (Step 1a)

Contrary to the original version of the CCD algorithm [12] here we use just a few carefully chosen pixels in the vicinity of the curve. The set $\mathcal{V}$ contains just pixels lying on specific perpendiculars to the expected image curve, see Fig. 3b. This causes a speed-up for two reasons: 1. the number of pixels taken into account is reduced. 2. for all pixels lying on the same perpendicular many quantities have to be computed only once per perpendicular.

The idea of taking only pixels into account which lie on specific perpendiculars is widely used in the tracking literature, e.g. [5]. However, here we do not use all pixels on the perpendicular but we use at most M_{max} (e.g. M_{max}=21) pixels per perpendicular. Hence, for an iteration step of our method the time complexity is limited independent of the image resolution. This allows to process high resolution images in a limited time.

3.1.1 Choosing Pixels

The pixels in the vicinity of the curve are chosen according to the local uncertainty of the curve. We describe the curve by a model curve function, similarly to [17] or [5]. The model curve function $\mathbf{c}(s, \mathbf{\Phi})$ maps the model parameters $\mathbf{\Phi}$ and a parameter s onto a curve point. The parameter s monotonously increases as the curve is traversed. Hence, s specifies a particular point on the curve. Contrary to [5], $\mathbf{c}$ does not have to be linear in the model parameters $\mathbf{\Phi}$ which is essential for many applications. The set $\Omega = \{s_1, ..., s_M\}$ of values for s specifies the set of perpendiculars to the image curve. Each perpendicular is defined by the curve point $\mathbf{C}_i = \mathbf{c}(s_i, \mathbf{m}_{\mathbf{\Phi}})$ and the corresponding normal vector $\mathbf{n}_i$. The values $s_i \in \Omega$ are usually chosen such that the perpendiculars are equally spaced along the curve.

The Gaussian distribution of model parameters $p(\mathbf{\Phi} \mid \mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ and the model curve function $\mathbf{c}(s, \mathbf{\Phi})$ define a probability distribution of the curve in the image. The pixels are chosen according to the local uncertainty of the curve in the direction perpendicular to the curve. This uncertainty can be efficiently estimated by approximating the curve $\mathbf{c}(s, \mathbf{\Phi})$ in the vicinity of the point $(s_i, \mathbf{m}_{\mathbf{\Phi}})$ by a linear Taylor series. The error of the approximation decreases if the covariance $\mathbf{\Sigma}_{\mathbf{\Phi}}$ decreases. This is the reason why our method achieves high accuracy despite local linear approximations. For the linearization the intersection of the curve and the perpendicular is Gaussian distributed. The mean point is $\mathbf{C}_i = \mathbf{c}(s_i, \mathbf{m}_{\mathbf{\Phi}})$ and the uncertainty, i.e. standard deviation, σ_i of the curve in the direction of the perpendicular is given by

$$\sigma_i = \sqrt{||\mathbf{J}_i^{\perp} \cdot \mathbf{A}||^2 + \widehat{\sigma}_i^2} \quad (1)$$

$$\text{where} \quad \mathbf{J}_i^{\perp} = \mathbf{n}_i^T \cdot \mathbf{J}_i. \quad (2)$$

The matrix $\mathbf{J}_i$ denotes the Jacobian of $\mathbf{c}$ in the point $(s_i, \mathbf{m}_{\mathbf{\Phi}})$ and $\mathbf{J}_i^{\perp}$ is the corresponding Jacobian for the direction perpendicular to the curve. The columns of the matrix $\mathbf{A}$ are the principle axis defined by the covariance matrix $\mathbf{\Sigma}_{\mathbf{\Phi}}$. The matrix $\mathbf{A}$ can be obtained by

$$\mathbf{A} = \left(\sqrt{\lambda_1} \cdot \mathbf{e}_1, ..., \sqrt{\lambda_{N_{\mathbf{\Phi}}}} \cdot \mathbf{e}_{N_{\mathbf{\Phi}}}\right) \quad (3)$$

where $\lambda_1, ..., \lambda_{N_{\mathbf{\Phi}}}$ are the eigenvalues and $\mathbf{e}_1, ..., \mathbf{e}_{N_{\mathbf{\Phi}}}$ are the corresponding eigenvectors of $\mathbf{\Sigma}_{\mathbf{\Phi}}$. The parameter $\widehat{\sigma}_i$ in

equation (1) incorporates other uncertainties apart from the uncertain model parameters $\mathbf{\Phi}$. Such uncertainties are for example uncertain internal camera parameters, uncertainties in the model, or image blurring caused by the imaging device. Usually for the first iterations $\widehat{\sigma}_i^2$ is very small compared to $||\mathbf{J}_i^{\perp} \cdot \mathbf{A}||^2$.

The pixels v_{ij} along a perpendicular i are chosen equally spaced within an interval, see Fig. 3b. The size of the interval depends linearly on the corresponding uncertainty σ_i. In order to avoid indices of indices we define for all variables $\mathbf{X}$ indexed by a pixel v_{ij} $\mathbf{X}_{ij} := \mathbf{X}_{v_{ij}}$. We denote the center points of pixels v_{ij} by $\mathbf{v}_{ij}$.

3.1.2 Assigning Pixels to Sides of the Curve

The distribution of image curves vaguely assigns pixels in the vicinity of the assumed curve to one side of the curve. The vague assignment $\mathbf{a} = (a_1, a_2)$ of a pixels consists of two probabilities. The first element a_1 is the probability of the pixel to lie on side 1 of the curve. The second element a_2 is the corresponding expression for side 2 given by $a_2 = 1 - a_1$. For the sake of efficiency, here the photosensitive area of the pixel is not taken into account, contrary to [12]. Here a pixel is simply modeled as a point, i.e. the center point. Based on the local perpendicular uncertainty σ_i of the curve the vague assignment $\mathbf{a}_{ij}$ for pixel v_{ij} can be obtained by

$$\mathbf{a}_{ij} = (a_{ij1}, 1 - a_{ij1}) \text{ where} \tag{4}$$

$$a_{ij1} = \frac{1}{2} \cdot erf\left(\frac{d_{ij}}{\sqrt{2} \cdot \sigma_i}\right) + \frac{1}{2}. \tag{5}$$

Here $erf(\cdot)$ denotes the error function and d_{ij} is the signed distance between the pixel center $\mathbf{v}_{ij}$ and the expected curve $\mathbf{c}(s, \mathbf{m}_{\mathbf{\Phi}})$. Using again the linear approximation of the curve the signed distance d_{ij} can be obtained by

$$d_{ij} = \mathbf{n}_i^T \cdot (\mathbf{v}_{ij} - \mathbf{C}_i). \tag{6}$$

Fig. 3c depicts the vague assignments to the background for two iteration steps. The pixels along a perpendicular can roughly be classified into two categories: 1.) pixels which are assigned to one side of the curve with high certainty. These pixels are used in the next section in order to estimate local statistics characterizing the corresponding side. 2.) pixels which are assigned with low certainty. In step 2 (see section 4) the estimation of the model parameters is refined such that these pixels are assigned to the side they fit best. The fitting criterion is based on the local image statistics.

3.2 Compute Local Statistics (Step 1b)

For the two sides separated by the curve local statistics of the image features are learned from the pixels which are assigned to one side with high certainty. We assume that the local statistical properties of both sides may vary along the curve. The computation of local statistics is done in two steps: 1.) local statistics are computed for each perpendicular independently. 2.) then the local statistics are smoothed along the curve.

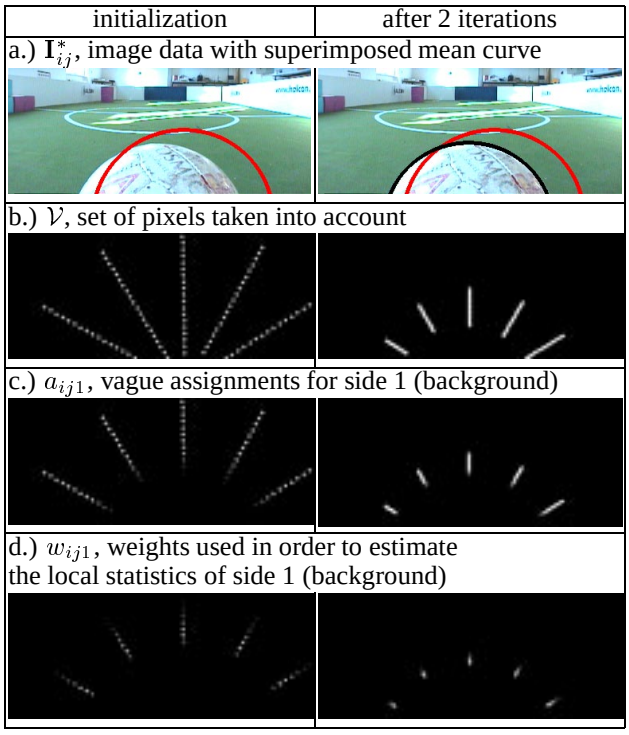


Figure 3: Within two iterations the model contour (black) is accurately aligned to the image contour. (red: initialization)

3.2.1 Statistics for one Perpendicular

The local statistics computed here characterize the two sides of the curve in the vicinity of the curve. This statistics define local Gaussian distributions which are used in step 2 in order to separate the two sides. For the two sides $s \in \{1, 2\}$ weighted statistical moments of the image features $\mathbf{I}$ (e.g. RGB values) are computed

$$\widehat{w}_{is} = \sum_{j \in \mathcal{V}_i} w_{ijs} \tag{7}$$

$$\widehat{\mathbf{M}}_{is} = \sum_{j \in \mathcal{V}_i} w_{ijs}\, \mathbf{I}_{ij} \tag{8}$$

$$\widehat{\mathbf{M}}^2_{is} = \sum_{j \in \mathcal{V}_i} w_{ijs}\, \mathbf{I}_{ij}\mathbf{I}_{ij}^T. \tag{9}$$

Here $\mathcal{V}_i$ denotes the set of all pixels v_{ij} belonging to the perpendicular indexed by i. The scalars w_{ijs} are weights specifying to which extend a pixel v_{ij} is taken into account for side s. For the choice of the weights w_{ijs} two conflicting considerations have to be taken into account. 1.) The

statistical dependency between two pixels usually decreases with the distance between the pixels. From this follows, local statistics should be computed from pixels close to the curve. 2.) However, pixels close to the curve are likely to belong to the wrong side. We compute a heuristic compromise by

$$w_{ijs} = E_p(d_{ij}, \sigma_i) \cdot E_s(a_{ijs}). \quad (10)$$

The function $E_p(d_{ij}, \sigma_i)$ evaluates the proximity of the pixel to the curve and $E_s(a_{ijs})$ evaluates the probability of the pixel to belong to the desired side s. For $E_p(d_{ij}, \sigma_i)$ we choose a zero mean Gaussian with a standard deviation depending linearly on σ_i. For $E_s(a_{ijs})$ we choose

$$E_s(a_{ijs}) \quad = \quad \max\left(0, [(a_{ijs} - C)/(1 - C)]^6\right) \quad (11)$$

with $C \in [0, 1[$. The function E_s is monotonously increasing and holds $\forall a_{ijs} \in [0, C] : E_s(a_{ijs}) = 0$ and $E_s(1) = 1$. In our experiments we use $C = 0.6$.

3.2.2 Smoothing Statistics along the Curve

In order to exploit statistical dependencies between pixels on different perpendiculars the statistical moments $\widehat{w}_{is}$, $\widehat{\mathbf{M}}_{is}$, and $\widehat{\mathbf{M}}^2_{is}$ computed for each perpendicular are blurred along the curve. We use an exponential filter since it allows for fast blurring with a time complexity independent of the window size. From the resulting blurred moments w_{is}, $\mathbf{M}_{is}$, and $\mathbf{M}^2_{is}$ the mean $\mathbf{m}_{is}$ and covariance $\mathbf{\Sigma}_{is}$ specifying local Gaussian distributions are obtained by

$$\mathbf{m}_{is} \quad = \quad \mathbf{M}_{is}/w_{is} \quad (12)$$

$$\mathbf{\Sigma}_{is} \quad = \quad \mathbf{M}^2_{is}/w_{is} - \mathbf{m}_{is}\mathbf{m}^T_{is} + \lambda\mathbf{I} \quad (13)$$

where $\lambda\mathbf{I}$ is a scaled identity matrix with very small λ used in order to avoid singularity for degenerated distributions.

4. Refine the Estimation of Model Parameters (Step 2)

In step 2 the estimation of the model parameters is refined using the vague pixel assignments and the local statistics obtained in step 1. Due to space limitation, unfortunately step 2 cannot be explained in detail here. For the basic idea and a mathematical description the reader is referred to [12]. Similarly to step 1, the speed-up in step 2 is achieved by mainly three modifications: 1.) with the reduced set $\mathcal{V}$ of pixels taken into account (see step 1) the overall computational load is reduced. 2.) for pixels lying on the same perpendicular several quantities have to be computed only once per perpendicular. 3.) by modeling a pixel as a point, integration over the photosensitive area of the pixel can be omitted.

5. Modeling the Ball Contour

In this section the contour of the ball is described (modeled) as a function of the ball position. In this modeling process knowledge of the ball (the object of interest) and the imaging device is incorporated. Here the ball is modeled as a sphere with known radius. We use a camera with known internal and external camera parameters. The method proposed here is developed for our non-omnidirectional vision system. However, the method can easily be adapted to omnidirectional vision systems which are quite popular in robotic soccer, e.g. [13, 24, 26].

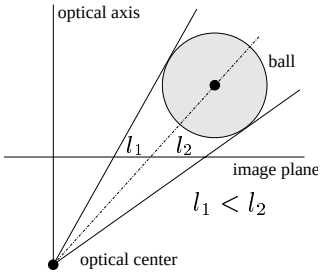


Figure 4: Projection of the ball into the image plane: the projection of the ball's center point does not yield the center of the ball contour

We denote the center point of the ball in the coordinate system of the observing robot by $\mathbf{M}_r$. We distinguish two cases: 1.) in the first case the ball is assumed to lay on the floor. Hence the center point $\mathbf{M}_r = (x, y, r)^T$ of the ball has two degrees of freedom, namely x and y. The z-coordinate is given by the radius r of the ball. 2.) in the second case we assume the ball may fly, i.e. not lay on the floor which sometimes happens in robotic soccer. In this case the ball position has three unknown coordinates: $\mathbf{M} = (x, y, z)^T$. While the second case is more general, it requires an optimization for more parameters and tends to be less precise, if the ball is on the floor. By $\mathbf{\Phi}$ we denote the unknown parameters of the ball position $\mathbf{M}_r$ (for the first case $\mathbf{\Phi} = (x, y)^T$, for the second case $\mathbf{\Phi} = (x, y, z)^T$).

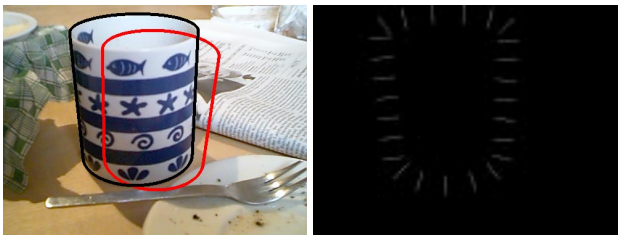
In the following the relation between $\mathbf{\Phi}$ and the pixel coordinates of the ball contour is derived. First the center point of the ball has to be expressed in camera coordinates $\mathbf{M}_c$:

$$\mathbf{M}_c = \mathbf{R} \cdot (\mathbf{M}_r - \mathbf{t}) \quad (14)$$

where $\mathbf{R}$ and $\mathbf{t}$ specify the orientation and location of the camera. The set of tangent lines to the sphere (the ball) passing through the optical center of the camera define a cone. As known, the intersection of the cone with the image plane yields an ellipse [29], (if the ball is visible). Note the intersection is a circle only if the center point of the sphere lies on the optical axis of the camera. In all other cases the projection of the center point is not the center point of the ball contour, see Fig. 4. The set of contour points can be described in undistorted image coordinates $\mathbf{u}$ by

$$\mathbf{u}(s) = \mathbf{m}_i + \cos(s) \cdot \mathbf{a}_1 + \sin(s) \cdot \mathbf{a}_2 \quad (15)$$

where $\mathbf{m}$ is the center, $\mathbf{a}_1$ and $\mathbf{a}_2$ are the two axis of the ellipse in undistorted image coordinates. The angle $s \in$



a.) b.)

Figure 5: Fitting a mug (modeled as a cylinder) into the image: a.) Despite the clutter, strong texture, and shading the proposed method correctly fits the model to the contour in the image (red: initialization, black: fitted contour). b.) Pixels used in the last iteration step.

$[-\pi, ..., \pi[$ specifies a particular point on the ellipse. Our lens causes substantial radial distortions. According to Lenz et al. [18] the distorted coordinates $\mathbf{d}$ can be approximated by

$$\mathbf{d} = (d_x, d_y)^T = 2\mathbf{u}/(1 + \sqrt{1 - 4\kappa|\mathbf{u}|^2}) \quad (16)$$

where κ is the distortion parameter of the lens. The pixel coordinates $\mathbf{c}$ of a point can be obtained by

$$\mathbf{c} = (d_x/S_x + C_x, d_y/S_y + C_y)^T \quad (17)$$

where S_x and S_y define the pixel size and C_x and C_y specify the center point of the camera.

6. Experiments

In our experiments we apply the proposed method to several scenes. In Fig. 5 the contour of a mug is fitted to the image data. The proposed method converges to the correct solution despite the strong clutter, texture, and shading.

In Fig. 3 the position of a ball is estimated. For close up view such as in Fig. 3 the average error is just a few millimeters, depending on the accuracy of the camera calibration. For this simple object and with the quite good initialization 2 iterations are sufficient. The computational time per iteration is about 18 ms on a 500 MHz computer.

In order to evaluate the performance for a partially occluded ball, we took two images with the same ball position, one with partial occlusion, Fig. 1, and one without occlusion, Fig. 6. The ball estimates of the two images differ by 2.2 cm, which is just 2.6% of the ball distance. However, the number of necessary iterations is about 5 times higher in the partially occluded case. Furthermore, in the non-occluded case the area of convergence is clearly bigger. The CCD algorithm not only yields an estimate of the ball position, but also a covariance matrix describing the expected uncertainty of the estimate. For the partially occluded ball the expected uncertainty is about three times higher than for the not occlude case. This allows a reasonable data fusion with other sensor data.

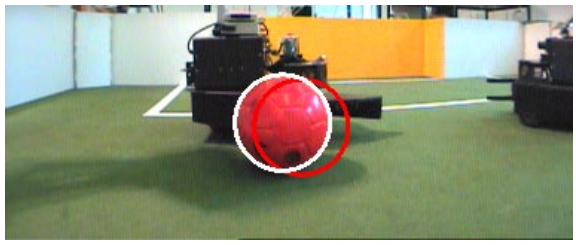


Figure 6: Same ball position as in Fig. 1: the two estimates differ by just 2.2 cm or 2.6% (red: initialization, white: estimated ball contour).

Figure 7: The mainly white non-RoboCup ball is accurately segmented in front of a white-grey background. This is hardly possible with color labeling (red: initialization, black: estimated ball contour).

Fig. 7 shows that the CCD algorithm can separate regions with similar color distributions. In the next experiment we vary the background of the ball and the illumination, see Fig. 8. For the five investigated images the standard deviation of the ball estimates is 1.1 cm which is 1.6% of the distance. Unfortunately, we do not have a sufficiently precise ground truth to compare our results with.

Figs. 8 and 5 show that the CCD algorithm can cope with strong texture even if the underlying distribution of the RGB values is not Gaussian but multi-modal. However, not all textures can directly be separated by the CCD algorithm. For example in order to separate two regions with the same texture but different orientation of the texture, a preprocessing step has to be applied yielding feature vectors which are sensitive to the orientation of the texture. The speed-up of the fast CCD algorithm depends mainly on the reduction of the pixels taken into account. In our experiments the fast CCD algorithm was more than 100 times faster than the original CCD algorithm.

7. Conclusion

We have proposed a novel and fast method for fitting parametric curve models to image data and we have applied this method to the problem of localizing objects observed by a mobile autonomous robot. Our method does not assume any given color distribution or specific edge-profile nor do we use any thresholds. We use local criteria based on local

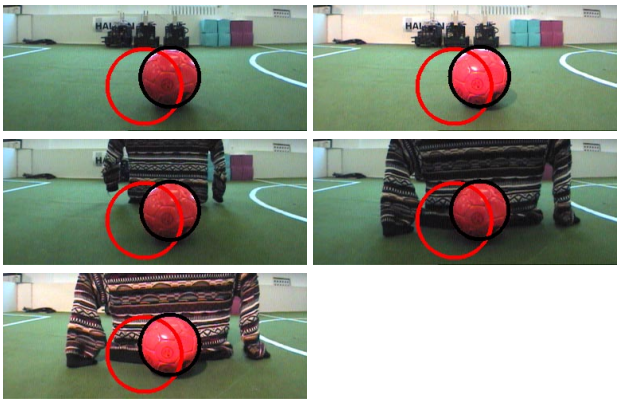


Figure 8: The strong variations of background and illumination cause just small variations of the ball estimates. For the five images the standard deviation of the ball estimates is 1.1 cm which is 1.6% of the distance.

image statistics in order to separate adjacent regions.

We have shown that the method achieves high robustness and accuracy even in the presence of strong texture, clutter, partial occlusion, and severe changes in illumination. Knowledge of the object of interest and the imaging sensor is explicitly modeled and exploited. This allows a straightforward adaption to other imaging devices and other problems. Due to the here proposed modifications of the CCD algorithm the method runs in real time for models with a small number of parameters, e.g. the ball. Since the method provides a confidence region for the estimates, a fusion with other uncertain sources of information can easily be accomplished, e.g. in the case of multiple observers.

References

[1] A.A. Amini, T.E. Weymouth, and R.C. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):855–867, September 1990.

[2] S. Baker, S.K. Nayar, and H. Murase. Parametric feature detection. *International Journal of Computer Vision*, 27(1):27–50, March 1998.

[3] T. Bandlow, M. Klupsch, R. Hanek, and T. Schmitt. Fast image segmentation, object recognition and localization in a RoboCup scenario. In *Third International Workshop on RoboCup (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science. Springer-Verlag, 1999.

[4] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Color- and texture-based image segmentation using the expectation-maximization algorithm and its application to content-based image retrieval. In *Proc. International Conference on Computer Vision*, pages 675–682, 1998.

[5] Andrew Blake and Michael Isard. *Active Contours*. Springer-Verlag, Berlin Heidelberg New York, 1998.

[6] J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In *International Conference on Intelligent Robots and Systems (IROS)*, 2000.

[7] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

[8] A. Chakraborty and J.S. Duncan. Game-theoretic integration for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(1):12–30, January 1999.

[9] C. Chesnaud, P. Refregier, and V. Boulet. Statistical region snake-based segmentation adapted to different physical noise models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1145–1157, November 1999.

[10] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Statist. Soc. B*, 39:1–38, 1977.

[11] R. Hanek and T. Schmitt. Vision-Based Localization and Data Fusion in a System of Cooperating Mobile Robots. In *Proc. of the IEEE Intl. Conf. on Intelligent Robots and Systems*, pages 1199–1204. IEEE/RSJ, 2000.

[12] Robert Hanek. The Contracting Curve Density Algorithm and its Application to Model-based Image Segmentation. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages I:797–804, 2001.

[13] F. Hundelshausen, S. Behnke, and R. Rojas. An omnidirectional vision system that finds and tracks color edges and blobs. In *5th International Workshop on RoboCup (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science. Springer-Verlag, 2001.

[14] M. Jamzad, B. Sadjad, V. Mirrokni, M. Kazemi, H. Chitsaz, A. Heydarnoori, M. Hajiaghai, and E. Chiniforooshan. A fast vision system for middle size robots in robocup. In *5th International Workshop on RoboCup (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science. Springer-Verlag, 2001.

[15] T.N. Jones and D.N. Metaxas. Image segmentation based on the integration of pixel affinity and deformable models. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 330–337, 1998.

[16] P. Jonker, J. Caarls, and W. Bokhove. Fast and Accurate Robot Vision for Vision based Motion. In P. Stone, T. Balch, and G. Kraetzschmar, editors, *4th International Workshop on RoboCup (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science, pages 72–82. Springer-Verlag, 2000.

[17] M. Kass, A.P. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, January 1988.

[18] R. Lenz and R. Y. Tsai. Techniques for Calibration of the Scale Factor and Image Center for High Accuracy 3-D Machine Vision Metrology. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(5):713–720, September 1988.

[19] M.E. Leventon, W.E.L. Grimson, and O. Faugeras. Statistical shape influence in geodesic active contours. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages I:316–323, 2000.

[20] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, November 1998.

[21] H. Luo, Q. Lu, R.S. Acharya, and R. Gaborski. Robust snake model. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages I:452–457, 2000.

[22] J. Malik, S. Belongie, J. Shi, and T. Leung. Textons, contours and regions: Cue integration in image segmentation. In *Proc. International Conference on Computer Vision*, pages 918–925, 1999.

[23] R. Manduchi. Bayesian fusion of color and texture segmentations. In *Proc. International Conference on Computer Vision*, pages 956–962, 1999.

[24] C. Marques and P. Lima. Vision-Based Self-Localization for Soccer Robots. In *International Conference on Intelligent Robots and Systems (IROS)*, 2000.

[25] T. McInerney and D. Terzopoulos. Deformable models in medical image analysis: a survey. *Medical Image Analysis*, 1(2):91–108, 1996.

[26] T. Nakamura, A. Ebina, M. Imai, T. Ogasawara, and H Ishiguro. Real-time Estimating Spatial Configurations between Multiple Robots by Triangle and Enumeration Constraints. In *International Conference on Intelligent Robots and Systems (IROS)*, 2000.

[27] V.S. Nalwa and T.O. Binford. On detecting edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):699–714, November 1986.

[28] N. Paragios and R. Deriche. Coupled geodesic active regions for image segmentation: A level set approach. In *Proc. European Conference on Computer Vision*, pages 224–240, 2000.

[29] J. Semple and G. Kneebone. *Algebraic projective geometry*. Oxford University Press, 1952.

[30] M. Simon, S. Behnke, and R. Rojas. Robust real time color tracking. In P. Stone, T. Balch, and G. Kraetzschmar, editors, *4th International Workshop on RoboCup (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science, pages 239–248. Springer-Verlag, 2000.

[31] B. Thirion, B. Bascle, V. Ramesh, and N. Navab. Fusion of color, shading and boundary information for factory pipe segmentation. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages II:349–356, 2000.

[32] S. Thrun, D. Fox, and W. Burgard. Monte Carlo localization with mixture proposal distribution. In *Proc. of the AAAI National Conference on Artificial Intelligence*, pages 859–865, 2000.

[33] T. Weigel, A. Kleiner, F. Diesch, M. Dietl, J.-S Gutmann, B. Nebel, P. Stiegeler, and B. Szerbakowski. CS Freiburg 2001. In *5th International Workshop on RoboCup (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Computer Science. Springer-Verlag, 2001.

[34] C.Y. Xu and J.L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing*, 7(3):359–369, March 1998.

[35] S.C. Zhu and A. Yuille. Region competition: Unifying snakes, region growing, and Bayes/MDL for multiband image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):884–900, September 1996.