2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

1 ms tracking of target boundaries using contour propagation

Niklas Bergström Masatoshi Ishikawa

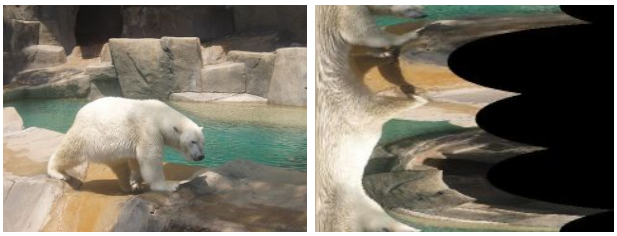
***Abstract*— We propose a method for tracking the boundary of an object at frame rates beyond $1\ kHz$, based on a novel contour propagation mechanism operating in polar image space. The work draws inspiration from well established methodologies in object tracking and segmentation. The main contribution is how the polar representation is exploited for tracking, including enabling parallelization for sub millisecond performance. The presented results show the feasibility of the method in a wide range of settings.**

I. Introduction

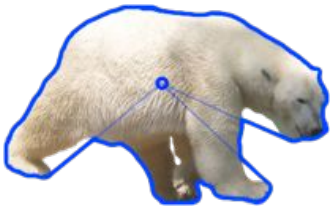
For a robot, object tracking is an essential capability, required for a wide range of tasks such as following the face of a human during interaction, or visual servoing for manipulation. One requirement of such methods is naturally that it needs to run fast enough for the robot to react. Algorithms referred to as *real-time* typically run in the order of tens of frames per second (fps), e.g. [1], framerates that humans perceive as smooth, and are often developed around the frame rate of standard video cameras.

On the contrary we believe that the capabilities of robots should exceed those of humans. High-speed cameras that operate at frame hundreds or thousands fps are becoming more affordable. This enables a whole new range of abilities that are impossible to humans and robots operating at lower frame rates. Two examples are accurately predicting the trajectory of a ball based on its rotation, or as we demonstrated in [2], predict the evolution of a folding cloth.

During the past decade some works on tracking algorithms operating at these frame rates have been presented. In [3] the authors tracked objects with a periodic motion pattern, such as a bird or a fan, at 1000 fps. By using a static camera and observing the frequency in intensity change for every pixel, such patterns could be observed and filtered from the background. In [4] a paramecium was tracked in a microscope at 500 fps. In this work the authors made strong assumptions about the characteristics of the object to improve tracking. The common point in both these works is that strong assumptions about the tracked objects are exploited to improve performance. In this work we instead want the method to be capable of dealing with *any* object. In [5] multiple objects were tracked at 2000 fps using color histograms, and matching these between frames. For successful tracking, the histogram of each object needs to be processed in advance, which assumes knowledge of the object. Differently, in this work we make very few prior assumptions about the tracked object.



(a) Original and polar images respectively



(b) The Polar Hull

Fig. 1. The top row shows an image with its polar counterpart. The pixels in the polar image correspond to angle and distance from a given point in the original image. The bottom row shows the concept of the *polar hull* given a point. The bear is segregated from the image using its boundary. The polar hull will be identical to the boundary as long as *the inside of the boundary* can be seen from the selected point. If not, illustrated by thin blue lines in the figure, parts of the background will be included in the hull.[2]

The demand for fast and robust tracking methods was emphasized in the work of Okumura *et al*. [6] In this work the authors presented a pan-tilt camera system that performs saccades at super human rates. This setup enables the robot to, differently to human saccades, keep track of the object *during* the saccade as well. It does this by capturing and processing video at a framerate of 1000 Hz and control the gaze using the mirrors based on visual input accordingly. To demonstrate the camera system the authors used color thresholding to track a ball [6]. While this worked well in the controlled setting of the paper, it will likely fail in other more cluttered environments, so there is a need for more robust algorithms that can cope with more difficult scenarios, as well as keep up with the framerate of the camera.

The high-speed tracking task puts high requirements on the algorithm in terms of both throughput and latency, so tradeoffs in terms of accuracy might naturally be necessary. On the other hand, knowing that images are captured with small time intervals makes it possible to exploit the fact that changes in the scene between images will be much smaller than for lower framerates. In [7] the authors explored

The authors are with the Department of Creative Informatics, Graduate School of Information Science and Technology, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan. niklas_bergstrom@ipc.i.u-tokyo.ac.jp masatoshi_ishikawa@ipc.i.u-tokyo.ac.jp

[2]Photo by Kevin ward

Fig. 2. The figure gives an example of the performance of the proposed method. The green contour is the initial guess which is taken from the previous frame and the red contour indicates the final result.[4]

tracking using a Markov Random Field (MRF) framework, This method searches for the best solution in a *global* sense by taking all pixels into account at every frame. In practice this means that the final solution is robust against differences in initialization. Exploiting assumptions about small changes between frames makes a *local* method equally tractable given that the contours of the object in the previous frame was accurately captured.

In this paper we propose a method for tracking the boundary of an object operating at framerates above 1 kHz. It draws inspiration from popular segmentation and tracking methods that propagate a contour using level-sets e.g. [8], [9], as well as the recent works in fixation-based segmentation [10]. A key concept that is introduced is the *polar hull*, which is illustrated in Fig. 1. This is a representation of the boundary of an object that makes a small tradeoff between accuracy and processing speed. For simple objects the polar hull will be identical to the boundary of the object, while for more complex objects parts of the background will be included in the hull. This is however not a problem for many commonly occurring objects. An example of the method can be seen in Fig. 2.

The paper is organized as follows: In the next section contributions and related work are presented. Sec. III describes the active contour model and Sec. IV describes the proposed approach. Sec. V presents some results of the method and finally in Sec. VI the paper is summarized.

II. Related Work and Contributions

Object tracking is a broad research field and can be arranged in different branches. Contrary to work assuming rigid [11] or known objects [12] we in this work focus on unknown deformable objects. Here tracking can further be divided into work just tracking the position or bounding box of an object and work tracking the actual contours of objects. Only tracking an object's position can be done at high frame rates using e.g. Mean Shift tracking [13]. This enables applications such as following the position of an object. However, as we exemplified in [2], tracking an object's boundary gives birth to a whole new range of applications. In this work we therefore focus on tracking of object boundaries.

A popular methodology for segmentation and boundary tracking is *active contours*. Here the problem is posed as the one of solving a partial differential equation (PDE) based on image and contour characteristics:

$$\frac{\partial C(s,t)}{\partial t} = F\vec{N}. \tag{1}$$

Here $C \in \mathcal{R}^2$ is the contour parameterized over a scalar $s \in [0,1]$, and time t. The equation says that the motion of the contour C over time is determined by a function over the image in the normal direction of the contour N. If the motion is zero everywhere, the contour has converged. One popular way of solving these PDE:s is through level-sets. With this formulation the contour is propagated by implicitly defining it as the zero-level on a 2D function Φ over the image domain, and updating Φ instead [14].

According to Eq. 1 the contour motion always occurs in its normal direction. This work deviate from this point as the motion occur in the *radial* direction with respect to the selected point of the polar transformation. While this might be limiting in some scenarios, the benefit is that all motion can be done in parallel.

One characteristic of the level-set formulation is that it is non-parametric and furthermore provides the ability easily change topology, which is a desired feature in e.g. medical imaging. The drawback however is that the propagation speed must be kept low for the optimization to find the true solution, resulting in slow convergence and thus long processing time. In [15] the authors introduced a way of solving the PDE using level-sets by updating Φ through direct propagation of the zero-level. This resulted in much improved convergence rate. This work draws from this idea and updates the contour directly. Differently however, the polar representation makes it possible to parallelize this propagation (see Sec. IV).

Contrary to the level-set formulation, the proposed method does not have the ability to change topology. This means that e.g. the space between the legs of the bear in Fig. 1 will be included in the segment, and that if there would have been two bears apart from each other, both might have been considered to be the same segment. For many objects this will however not be an issue. Rather, in most cases we are interested in tracking objects as *whole* objects, meaning that changes in topology is undesirable.

In [10] the authors performed segmentation by first converting images to polar space. One motivation was that the performance of the method will then be independent of object size. They formulated the problem as a minimization problem over an MRF and solved with graph-cuts [16]. Apart from using a global solver, the computational speed of the method was heavily impacted by the usage of boundary detection [17], making the segmentation run in the order of seconds

[4]Video by ChuΩng Vo

or tens of seconds. We recognize the benefits of using polar representation, but formulate the problem of tracking the boundary in a way that allows for orders of magnitude faster processing.

In [7] the authors use belief propagation optimized for the GPU to solve the MRF, providing real-time performance. This however demonstrates the limitations of a global method when looking for performance in the order of magnitudes faster. Therefore we here utilize a local method.

In the class of segmentation and tracking methods that use level-sets, objects need to be defined based on some criteria. Two popular ways are to model them using their boundary against the background [18], or as done in region based methods, model their color [8]. The force F in Eq. 1 is then determined by how well the current contour fits the respective models. We adopt the region based methodology and exemplify the performance of the proposed method compared to a traditional solver.

The main contributions of this work are:

- A tracker for non-rigid objects using contour propagation in polar image space. The polar space formulation gives advantage in terms of naturally restricting the tracking to *one* object, the center of the objects will be given directly by the polar representation, and parallelizability.
- An efficient parallel implementation of the algorithm enabling tracking at $> 1\ kHz$ frame rates.

III. Active Contours

As described in the previous sections, contour propagation using active contours is a popular way of segmenting and tracking objects in an image or video. In this section we give an introduction to the idea of contour propagation and performing this using level-set methods.

In level-set methods the contour C is defined implicitly through the zero level on a function

$$\Phi : (x, y, t) \rightarrow \phi. \tag{2}$$

which in turn is defined on the image domain. In other words:

$$C = \{(x', y') | \Phi(x', y', t) = 0\}. \tag{3}$$

In Eq. 1 the evolution of a parameterized curve was expressed. Using the implicit notation, given that $\vec{N} = \nabla\Phi/|\nabla\Phi|$, the propagation of the curve can be expressed as [19]:

$$\frac{\partial\Phi}{\partial t} = F|\nabla\Phi|,\ \Phi(x, y, 0) = \Phi_0(x, y). \tag{4}$$

Thus there is a simple relationship between the parameterized and level-set formulations.

It is common to formulate the right side of Eq. 4, i.e. the function influencing the contour motion, in terms of external and internal energies:

$$E(\Phi) = E_{ext}(\Phi) + \lambda E_{int}(\Phi). \tag{5}$$

The *internal* force affects the contour based on its length, and the *external* force affects the contour based on properties of the image. The former will typically have a smoothing effect on the curve, preventing it from overfitting to image data, while the latter will make sure that the contour moves towards actual boundaries in the image.

Propagating C means searching for the minimum energy E through gradient descent:

$$\frac{\partial\Phi}{\partial t} = \frac{\partial E}{\partial\Phi}. \tag{6}$$

In the discrete domain this can be formulated accordingly:

$$\frac{\Phi^{t+1} - \Phi^t}{\Delta t} = \frac{\partial E}{\partial\Phi} \Rightarrow \tag{7}$$

$$\Phi^{t+1} = \Phi^t + \Delta t \frac{\partial E}{\partial\Phi}. \tag{8}$$

Thus by iteratively updating Φ the contour implicitly defined by the zero level will converge towards some minima.

Algorithm 1:

input: Image sequence S, fixation point $p_0 = (x_0, y_0)$ and circle radius r_0

(A) $C_0 \leftarrow$ createContour(p_0, r_0)
(B) $I'_0 \leftarrow$ rgb2hsv(I_0), $\theta_0 \leftarrow$ createModels(I'_0, C_0)
for each $I_t \in S$ **do**
  (1) $I'_t \leftarrow$ rgb2hsv(I_t)
  (2) $I^p_t \leftarrow$ polarImage(I'_t, p_{t-1})
  (3) $C_t \leftarrow$ propagateContour$(I^p_t, \theta_{t-1}, C_{t-1})$
  (4) $\theta_t \leftarrow$ updateModels$(\theta_{t-1}, I^p_t, C_t)$
  (5) $p_t \leftarrow$ updateMean(C_t, p_{t-1})
end

IV. Novel Formulation

Here the proposed approach is introduced in detail. First we give a brief introduction to the different steps of the algorithm. Then describe how the object and background are modeled, and then the curve propagation using level-sets in the context of the model. After that we introduce the proposed approach and how it relates to the methods described above. Finally we provide some implementation details.

A. Algorithm Summary

Alg. 1 shows the different steps of the algorithm. It is initialized in our case by manually marking a point in the image and deciding the size of the circle to be used as initial contour (A). Initial models are then created based on this contour (B) (Sec. IV-B). After initialization, each image in the sequence is processed as follows: The image is converted to HSV space which generally gives better performance (1). Then this image is transferred to polar space (2) (Sec. IV-D.1). Using the polar image, current models and contour, the contour is propagated until convergence (3) (Sec. IV-D.2), and finally models and mean are updated (4),(5) (Sec. IV-D.3).

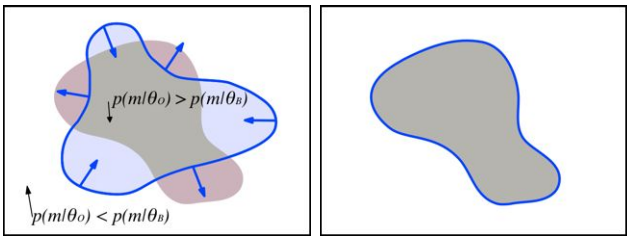


Fig. 3. The left figure illustrates the model concept as well as the concept for curve propagation. In the scene there is a gray object on a white background, modeled with histograms θ_o and θ_b respectively. The current estimate of the boundary is identified with a blue contour. Outside the gray object the probability of a measurement m belonging to the background is greater than the foreground, and vice versa for the inside of the gray object. Light blue areas indicate parts on the object that for the current contour estimate reside in the background and thus fit poorly to the model. The light red areas indicate the same parts in the background currently estimated to be on the object. Together these parts will add more to the total energy than if they would be estimated to belong to the other region. The right figure shows this case with the optimal contour in which the energy is minimized.

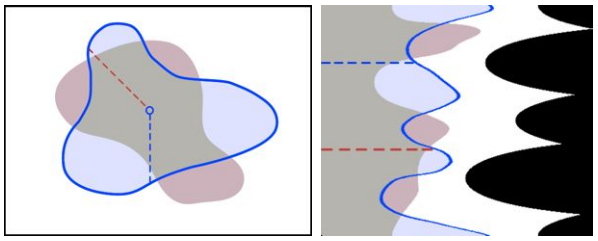


Fig. 4. The left figure shows the original image from Fig. 3, and the right the corresponding polar image generated from the indicated center. In both images the 90° and 225° lines from this center are displayed. Straight lines from the center in the original images correspond to lines along rows in the polar image.

B. Modeling and Initialization

In order to know in which direction to propagate the curve towards the object boundary, the object has to be modeled according to some criteria. There exists a large number of variations on Eq. 4 that take into account different strategies for both internal and external energies. One way is to model the boundary directly through discontinuities in image intensity [18]. The underlying idea is that the intensity of an object close to its boundary is different from the background around that boundary and therefore a good indicator of the object. Accurate boundaries are however inherently difficult to identify since e.g. internal texture of the object will generate discontinuities as well which will confuse the method.

The approach taken here is instead to represent the object and background using the color in respective regions, rather than the boundary separating them, similarly to what was proposed in [8]. This strategy will ensure that the regions inside and outside the contour are internally similar, but different between the regions. Given an image measurement m_i at pixel $i = (x_i, y_i)$ the probability of it belonging to either object o or background b is:

$$p(m_i \in \Omega_k) = \theta_k(m), \;\; k \in \{o, b\}. \tag{9}$$

The functions θ_k represents the color distribution over the image regions Ω_k with a histogram.

The energy that we want to minimize can then be formulated as

$$\begin{aligned} E_C(C) = &- \int_{i \in \Omega_o} \log p(m_i|\theta_o) d\Omega \\ &- \int_{i \in \Omega_b} \log p(m_i|\theta_b) d\Omega + \lambda S_C(C). \end{aligned} \tag{10}$$

S_C is a function describing the length of C. Short length implies low energy, which will force the contour to not overfit to the data. Fig. 3 shows the intuition behind the two first terms.

Before tracking can begin, the models need to be initialized. This could be done automatically by using attention mechanisms [20]. In this work we manually select a *fixation point*, i.e. a point on the object that will be the base of the polar transform, and create a circle around that fixation point. Object and background models are created from pixels lying inside or outside of the circle respectively. The model creation and updates can be done directly in polar space by adding to the histogram proportionally to the distance to the fixation point.

C. Level-Set Formulation

Before detailing the proposed method in Sec. IV-D, we describe how it can be done using level-set methods. In order to minimize the energy in Eq. 10 the common approach is to embed the contour in a function Φ over the image domain, as described in Sec. III. In order to do this the variables depending on the contour in Eq. 10 must be expressed using Φ. The contour C was defined in Eq. 3, and the regions Ω_o and Ω_b are defined as:

$$\Omega_o = \{(x, y) | \Phi(x, y, t) > 0\} \tag{11}$$

$$\Omega_b = \{(x, y) | \Phi(x, y, t) < 0\}. \tag{12}$$

Thus the energy from Eq. 10 can be reformulated as:

$$\begin{aligned} E(\Phi) = - \int_{i \in \Omega} [\; &\log p(m_i|\theta_o) H(\Phi_i) + \\ &\log p(m_i|\theta_b)(1 - H(\Phi_i)) \;] d\Omega \\ &+ \lambda S(\Phi). \end{aligned} \tag{13}$$

Here Φ_i is the value of Φ at pixel i and H is the *Heaviside* function:

$$H(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

The contour length can be expressed using Φ as [8]:

$$S_C(C) = S(\Phi) = \int_\Omega \delta_0(\Phi_i) |\nabla \Phi_i| d\Omega. \tag{15}$$

δ_0 is the dirac delta function, which is the derivative of H:

$$\delta_0(\Phi) = \frac{\partial H(\Phi)}{\partial \Phi}. \tag{16}$$

Eq. 13 is minimized using gradient descent, so we seek the Gâteaux derivative of the functional $E(\Phi)$:

$$\begin{aligned} \frac{\partial E(\Phi)}{\partial \Phi} = \delta_0(\Phi) \Big[&- \log p(m_i|\theta_o) + \log p(m_i|\theta_b) \\ &+ \lambda \text{div} \left(\frac{\nabla \Phi}{|\nabla \Phi|} \right) \Big]. \end{aligned} \tag{17}$$

Looking at Eq. 17 the first thing to observe is that Φ has no change outside the curve since $\delta_0(x) = 0, x \neq 0$. On the contour the force will be a competition between how well that point fits the object model compared to the background model, and the length of the contour. If the point fits the foreground better than the background this will generate an outward force and vice versa. The last term is a measure of curvature, which will act as a counter force whenever the curvature is high, effectively reducing the length of the contour.

Using H and δ_0 as defined above would result in numerically unstable solutions. Therefore approximations of these are commonly used:

$$H_\epsilon(x) = \frac{1}{2}\left(1 + \frac{2}{\pi}\arctan\left(\frac{x}{\epsilon}\right)\right) \tag{18}$$

$$\delta_e(x) = \frac{1}{\epsilon\pi(1 + (\frac{x}{\epsilon})^2)}. \tag{19}$$

In this way a portion around the border will be considered during the updates. As pointed out in [21] this is also an indicator of uncertainty of the location of the border. Using these smooth versions the curve can be updated iteratively by combining Eq. 8 and Eq. 17.

D. Propagation in Polar Space

Contrary to level-set methods we in this work we present a method for propagating the curve without explicitly computing Φ, but instead working directly on the contour defined by $\Phi = 0$. The takes advantage of the *polar representation* of the image as described below.

1) Polar Representation: The method formulation relies to a large extent on the polar transform of the image. This transform is based on a *fixation point* (x_f, y_f) in the original image. Each point (r, β) in the polar image corresponds to coordinates

$$(x, y) = (x_f + r\cos\beta, y_f + r\sin\beta). \tag{20}$$

in the original image. Each row in the polar image will thus correspond to one ray from the fixation point with an angle determined by the column. The number of columns in the polar image will differ depending on where the fixation point is selected, while the number of rows, B, is a parameter to the method. Fig. 4 shows an example of the transform.

We define the object contour in this representation using one contour point per row, which then is moved along that row. This corresponds to a move in the radial direction in the original image. This representation gives rise to the concept of the polar hull. Since two contour points cannot lie on the same radial line, there will be a discontinuity along these lines whenever the outside of the boundary faces the fixation point. In practice however, many objects will not present these discontinuities, and for those who do, this effect will be negligible.

2) Contour Propagation: In the active contour formulation, propagation in the tangential direction can be seen as a re-parameterization of the contour. Hence propagation is done only in the negative or positive normal direction of the contour. As noted above, we propagate the contour in the radial direction, which will often correspond to a linear combination of normal and tangential direction in the original image. Here, however, the tangential part cannot be expressed through re-parameterization since the motion of the contour is restricted.

Algorithm 2: Contour propagation

input : $C^t = \{c_i^t\}$ (the contour at time t), measurements m, and models θ_o, θ_b
output: $C^{t+1} = \{c_i^{t+1}\}$

for each $c_i^t \in C^t$ **do**
  $f_o \leftarrow \frac{\sum_{j<c_i^t} r_o(j)\log p(m_i|\theta_b)}{\sum_{j<c_i^t} r_o(j)\log p(m_i|\theta_o)}$
  $f_b \leftarrow \frac{\sum_{j>c_i^t} r_b(j)\log p(m_i|\theta_o)}{\sum_{j>c_i^t} r_b(j)\log p(m_i|\theta_b)}$
  if $f_o > 1 \wedge f_b < 1$ **then**
    $c_i^{t+1} \leftarrow c_i^t + 1$
  else if $f_o < 1 \wedge f_b > 1$ **then**
    $c_i^{t+1} \leftarrow c_i^t - 1$
  else
    $c_i^{t+1} \leftarrow c_i^t$
  end
end

From Eq. 17 and Eq. 19 we know that the sign of $\partial E/\partial\Phi$ around $\Phi = 0$ will effectively determine the direction of curve propagation, i.e. whether the motion will be inward or outward. We therefore propose instead to move the contour one step inward or outward depending on the directions computed from Eq. 17 in a surrounding of the contour.

We divide the propagation into two steps: One for external and internal energy respectively. The first step will propagate the contour according to the object and background models, and the second step will smooth the contour. This is similar to what was done in [15]. The first step is done according to Alg. 2. For each contour point the fit to object and background models are measured. r_o and r_b is used for the same purpose as δ_ϵ, i.e. account for contour uncertainty. They give high values close to the contour and low values away from the contour giving more importance to values around the contour while still including information further away from it. The intuition behind f_b and f_o is that they indicate how well inside and outside of the contour fit to the object or the background. The contour should then be moved accordingly. If both indicate that inside and outside are correctly estimated nothing should happen, which is also the case if the both inside and outside are wrongly estimated. In this case it is unclear in which direction to move, but if surrounding points can move, the smoothness term will generate a force on the point in question as well.

In the second step the contour smoothness is considered. Shi *et al*. made the observation that contour evolution due to its curvature computed by the Laplacian, is equivalent of smoothing the contour with a Gaussian [15]. In the polar

space this can be realized by filtering the contour with a 1D Gaussian filter. In this work however, high curvature is not necessarily a bad thing, which can be seen by considering a contour that lies in the radial direction from the fixation point. To account for this we adapt the variance of the Gaussian filter based on how well the inside and outside of the contours fit their respective model.

3) Tracking: The contour is propagated in each frame using the method described in the previous section. Between frames several steps need to be taken, like updating the fixation point and the color models.

After the contour has converged the color model is updated. Histograms $\theta'_k,\ k \in \{o, b\}$ are computed given the new contour in frame t and the models from frame $t-1$ are updated:

$$\theta_k^t = (1-\alpha)\theta_k^{t-1} + \alpha\theta'_k,\ k \in \{o, b\}. \tag{21}$$

α is a parameter of the method that governs the adaptivity of the model.

Using the high frame rate assumption, i.e. that the object only moves marginally between frames, we can initialize the contour of frame $t+1$ with the converged contour from frame t. However, when the contour has converged in the previous section it is likely that the object mean position has changed, and a new one needs to be calculated. It would be too costly to transfer the polar representation of the contour back to cartesian space, compute the mean and then convert back to polar space. Instead we introduce the *polar mean* computed as:

$$\overline{x} = \sum_{i \in C} r_i \cos\beta_i / |C|, \quad \overline{y} = \sum_{i \in C} r_i \sin\beta_i / |C|. \tag{22}$$

This gives the change in x and y relative to the previous fixation point which is updated accordingly. This will not give the same result as the cartesian mean, but rather strives to keep the variance of the contour in polar space low. We did not find any drawbacks of using this mean in the experiments.

When the mean has been updated, the contour needs to be updated as well since it depends on the fixation point. Here the polar hull comes into play since the new mean might observe two of the old contour points in the same direction. In this case we keep the outermost one.

4) Implementation Details: All parts of the algorithm have been implemented and tested on a standard 2.6 GHz Intel Core i7 CPU with 16 GB RAM, using multi threading and vector instructions. There are four time consuming parts: 1) Conversion to HSV color space, 2) polar transform and 3) histogram updates and 4) contour propagation. For a VGA-image, color conversion takes ~ 0.16 ms. Using $B = 360$ rows in the polar transform takes ~ 0.30 ms, histogram updates ~ 0.20 ms and contour propagation between 0.20 and 0.30 ms.

The the first step contour propagation can be done completely in parallel, while in the second step each point is depending on their closest neighbors. We assign a range of rows to each thread, and let each thread in addition process one point extra on each side of the range. In this way both steps can be done in parallel.

V. Experiments

In this section we present a number of sequences highlighting different aspects of the method. The sequences are recorded with a wide range of high-speed imaging devices operating at frame rates from 200 to 1000 fps. In this work processing has been done offline by first loading the sequences into memory, and then process them to evaluate the algorithm performance. Parameters are set to: $B = 360$, $\alpha = 0.01$. The histogram size was kept to 20 for each channel. If nothing is said, the Hue and Saturation channels were used. In the future we plan to implement the algorithm on the saccade mirror [6] for online experiments.

In the figures, green outlines indicate the initial contour for that frame, and red outlines indicate the propagated contour. In each case the first image of the sequence is shown with the manually selected initial contour (circle).

a) Ski sequence: This sequence, seen in Fig. 5(a), was recorded using a compact consumer camera with a resolution of 512×384 pixels at 240 fps. The skier's body is constantly changing position, with arms and legs pointing in different directions. Despite the polar representation, the extremities are well covered by the contour. All three color channels were used.

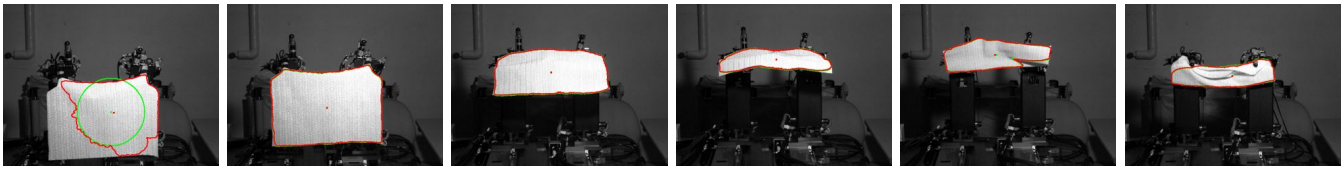
b) Cloth folding sequence: This sequence, seen in Fig. 5(b), is taken from our previous work [2], and was captured using a EoSens camera at 1000 fps with VGA resolution. The images are captured in grayscale, so a 1-D histogram was used for the models.

c) Ball throwing: Fig. 5(c) shows this sequence, which was recorded with using the same camera as the ski sequence. The surface of the ball turns inside out when thrown, making it change appearance rapidly from orange to green. However, as the appearance change, the models will adapt and eventually the contour will capture the entire ball. As a reference we run this example using the same model but with the solver proposed in [21]. The results are shown in Fig. 7. Using the same initialization, this method will initially give similar results as the proposed method, but will then converge towards covering the hand as well. Furthermore, after a while the contour will change topology, resulting in two separate segments.

d) Ball dribbling: The ball sequence is recorded with the saccade mirror system [6]. The images are 128×128 pixels and recorded at 1000 fps. The difficulty in this sequence is the similarity with the blue ball to the background (see Fig. 6). However, the polar representation makes it difficult for the contour to propagate to the vehicle. The sequence is shown in Fig. 5(d).

e) Shaky sequence: These sequences were captured with VGA resolution at 200 Hz using a Point Grey Grashopper camera, by observing at a static object with the camera shaking fiercely. The first sequence (Fig. 5(e)) is correctly handled by the algorithm, but its limitations can be observed in the second case (Fig. 5(f)). Here the camera moves more than the width of the object between two frames, and thus the initial estimate in the second frame ends up entirely outside the object. The simple solution would be to use a camera with

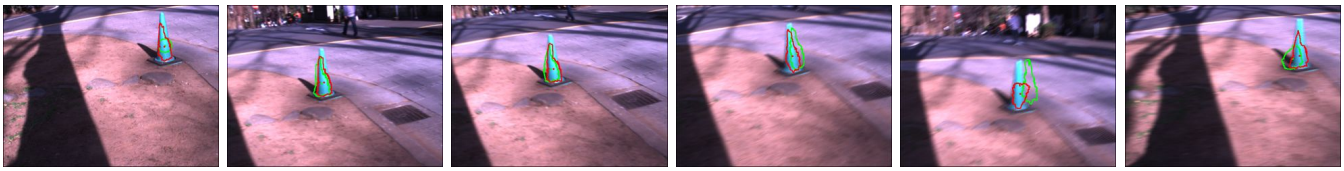
(a) **Ski**.The human body is articulated and subject to the limitations of the polar hull. Even so it is well captured in the sequence. Small effects can be observed in the third and fifth frame which lack a part of the arm, and the last frame where the background is included in the object.



(b) **Cloth folding**. This sequence is hard due to the many shadows on the cloth, as well as the fact that there is only one color channel. Between the fourth and fifth frame the cloth changes from having a visible front side to showing its back side to the camera, which introduces many new shadows on the cloth. The method recovers from this and continuously keeps track of the contour.

(c) **Ball throwing**. The initial circle is small, only capturing an orange part of the ball. As the ball is thrown, more parts get included until the entire ball is captured by the contour.

(d) **Ball dribbling**. Even though parts of the contour spread to the vehicle, the ball is always enclosed. When it starts to move it will therefore drag the contour along with it effectively keeping the contour around the ball.



(e) **Shaky sequence 1**. The moving camera introduces a large amount of blur in the image, as well as providing bad initial guesses to the contour propagation. This effect can be observed in the fifth frame where the propagated contour is a bad fit to the object. This is due to the contour being propagated with the fixation point from the green contour. By iteratively propagating the contour and updating the mean the contour could therefore be more accurately captured at the expense of reduced performance.

(f) **Shaky sequence 2**. In this sequence the motion of the camera is too large resulting in the initial segment in the fifth frame being completely outside the object. In the last frame there is no measurements close to the foreground model inside the contour, so it shrinks to a single point. The effect here efficiently demonstrate what would happen in many of the presented examples if the frame rate was reduced.

Fig. 5. Each figure shows six frames from different sequences. Except for the last sequence, in which the four last frames are adjacent in time, frames with the order of 100 of frames in between them are selected.

Fig. 6. The skier is well captured by the method, as is the blue ball. The method captures the cone well, but also parts of the background and splits into several segments.

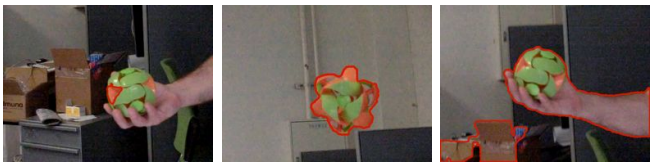


Fig. 7. Three frames from the ball throwing sequence. In the first and second frames for both methods converge to roughly the same contour. However, in before the third frame the method has adapted too much of the hand and the segment then spreads to both the arm and a distinct object in the vicinity.

higher frame rate, but another solution would be to estimate the velocity of the mean in addition to the mean itself.

For comparison we implemented the same model with the level-set solver in [21]. Fig. 6 and 7 show these results. The results from the proposed method is comparable, while being generated at orders of magnitude faster. The skier is naturally captured more precisely, but the compared method also over adapts and changes topology which can be seen in the ball throwing and shaky sequences.

VI. Discussion

Robots have the potential to perform tasks that widely exceeds what humans are able to perform. Hardware and control mechanisms already exist that enable the robot to get visual input from its environment and react to it faster than humanly possible. This puts high requirements on the vision algorithms designed for these scenarios. In this paper we presented a method for tracking an object's boundary operating at frame rates above 1000 fps. Inspired by the level-set methodology, the method operates in polar image space and propagates the contour along each row in the polar image.

Through the new formulation, contour propagation can be parallelized, the mean of the segment is part of the definition, and the contour topology is not allowed to change, effectively eliminating ambiguous situations involving double segments. Our experiments show that the method is applicable in a wide range of scenarios where video sequences are captured at high speed. Furthermore, the tracking results are for these scenarios comparable to standard level-set methods while being generated at orders of magnitude faster.

In the future we want to explore how to minimize the need for human intervention. While fairly robust to method parameters, currently the human needs to select e.g. histogram size, fixation point and initial contour size. Trough automatic model selection and sailency detection through e.g. motion the need for a human to supervise the system should be minimized.

Acknowledgements: This research was supported by Japan Society for the Promotion of Science (JSPS).

References

[1] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1619–1632, Aug.

[2] N. Bergström, C. H. Ek, D. Kragic, Y. Yamakawa, T. Senoo, and M. Ishikawa, "On-line learning of temporal state models for flexible objects," in *Humanoids 2012*, Osaka, Japan, November 2012.

[3] I. Ishii, I. Ohara, T. Tatebe, and T. Takaki, "1000-fps target tracking using vibration-based image features," in *ICRA 2011*, May 2011, pp. 1837–1842.

[4] X. Fei, Y. Igarashi, and K. Hashimoto, "Parallel computation of level set method for 500 hz visual servo control," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, ser. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 2126, November 2008.

[5] Q. Gu, T. Takaki, and I. Ishii, "2000 fps multi-object tracking based on color histogram," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 8437, 2012, p. 12.

[6] K. Okumura, H. Oku, and M. Ishikawa, "High-speed gaze controller for millisecond-order pan/tilt camera." in *ICRA*, 2011, pp. 6186–6191.

[7] M. Björkman and D. Kragic, "Active 3d scene segmentation and detection of unknown objects," in *ICRA*, 2010, pp. 3114–3120.

[8] T. Chan and L. Vese, "Active contours without edges," *Image Processing, IEEE Transactions on*, vol. 10, no. 2, pp. 266 –277, feb 2001.

[9] C. Li, C. Xu, C. Gui, and M. D. Fox, "Distance regularized level set evolution and its application to image segmentation," *Trans. Img. Proc.*, vol. 19, no. 12, pp. 3243–3254, December 2010.

[10] A. K. Mishra and Y. Aloimonos, "Active segmentation," *I. J. Humanoid Robotics*, vol. 6, no. 3, pp. 361–386, 2009.

[11] Y. Boykov and D. Huttenlocher, "Adaptive bayesian recognition in tracking rigid objects," in *CVPR, 2000. Proceedings. IEEE Conference on*, vol. 2, 2000, pp. 697–704 vol.2.

[12] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 7, pp. 932–946, Jul.

[13] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *CVPR, 2000. Proceedings. IEEE Conference on*, vol. 2, 2000, pp. 142–149 vol.2.

[14] S. Osher and J. A. Sethian, "Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulations," *Journal of Computational Physics*, vol. 79, no. 1, pp. 12–49, 1988.

[15] Y. Shi and W. Karl, "Real-time tracking using level sets," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, june 2005, pp. 34 – 41 vol. 2.

[16] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 11, pp. 1222–1239, 2001.

[17] D. Martin, C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 5, pp. 530 –549, may 2004.

[18] V. Caselles, "Geometric models for active contours," in *Image Processing, 1995, International Conference on*, vol. 3, October 1995, pp. 9–12 vol.3.

[19] M. Rousson and N. Paragios, "Shape priors for level set representations," in *Computer Vision — ECCV 2002*, ser. Lecture Notes in Computer Science, A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, Eds. Springer Berlin / Heidelberg, 2002, vol. 2351, pp. 416–418.

[20] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *PAMI, IEEE Transactions on*, vol. 20, no. 11, pp. 1254 –1259, nov 1998.

[21] C. Bibby and I. Reid, "Robust real-time visual tracking using pixel-wise posteriors," in *Proceedings of ECCV*, 2008.