A Combinatorial Solution for Model-based Image Segmentation and Real-time Tracking

Thomas Schoenemann and Daniel Cremers
Department of Computer Science
University of Bonn, Germany

***Abstract*—We propose a combinatorial solution to determine the optimal elastic matching of a deformable template to an image. The central idea is to cast the optimal matching of each template point to a corresponding image pixel as a problem of finding a minimum cost cyclic path in the three-dimensional product space spanned by the template and the input image.**

We introduce a cost functional associated with each cycle which consists of three terms: a data fidelity term favoring strong intensity gradients, a shape consistency term favoring similarity of tangent angles of corresponding points and an elastic penalty for stretching or shrinking. The functional is normalized with respect to the total length to avoid a bias toward shorter curves.

Optimization is performed by Lawler's Minimum Ratio Cycle algorithm parallelized on state-of-the-art graphics cards. The algorithm provides the optimal segmentation and point correspondence between template and segmented curve in computation times which are essentially linear in the number of pixels. To the best of our knowledge this is the only existing globally optimal algorithm for real-time tracking of deformable shapes.

I. Introduction

Image segmentation and the tracking of objects are two of the most prominent topics in computer vision. Numerous authors have tried to solve these problems based on low-level information such as edges or region statistics [25], [34], [3], [42], [21]. However, their success has been limited: in real-world images the low-level information is often corrupted, e.g. by changing lighting conditions and low contrast between object and background. As an example consider Figure 1 where a car is tracked in rainy weather.

To cope with such challenges researchers have endeavored to integrate prior knowledge into the respective segmentation processes. In numerous studies [19], [4], [13], [7] this was shown to significantly improve the resulting segmentations. However, most of these methods find local minima and hence require an initialization in the vicinity of the solution. The one that does find globally optimal segmentations [13] has a quadratic memory complexity. It is hence well-suited for tracking tasks but for pixel-accurate image segmentation only rather coarse resolutions can be handled.

In this work we present the first globally optimal shape-based segmentation method able to yield pixel-accurate segmentations in effectively linear time.

A. Related Work

Image segmentation and tracking are closely related problems, yet each with its own history. We therefore review them separately.

1) Tracking Deformable Objects: The tracking of objects has traditionally been based on feature points [17], [22]. Starting from the KLT-tracker [42], subsequent feature-based methods appeared in [21], [30].

More recently methods have become popular that treat the object as an entity [11], [6], [20] rather than an independent number of parts. Denzler and Niemann [11] consider a set of patches which are linked by a ray-model. Cremers [6] models the temporal evolution of shapes by a dynamical, autoregressive model in a level set framework. This is extended by Gui et al. [20] to the case of competing priors.

While many of these methods are based on minimizing a suitable energy, none guarantees to find the global optimum. To improve performance and avoid poor local minima, researchers have resorted to stochastic methods such as particle filtering [2], [12] or Kalman filtering [10]. However, such methods give neither a guarantee to find good (i.e. low energy) solutions nor a means to verify if a solution is optimal.

To determine global optima in the presence of significantly deforming curves has remained an open challenge. Furthermore, real-world applications typically require fast algorithms that can run in real-time. Exactly this challenge is solved in this paper.

2) Shape-based Image Segmentation: The task to partition an image into meaningful regions has received considerable attention in the past. When the limits of low-level methods [25], [34], [3] became apparent, researchers have endeavored to integrate prior knowledge into segmentation processes. The amount of prior knowledge varies from a part-based (deformable) object structure [15], [14], [35] over a collection of shapes [19], [4], [29], [43], [9], [37], [7], [8], [28] to a single shape [13]. We will focus on the latter two categories since they are closer to our work.

For a long time global approaches to the problem were not available, which inspired a number of local approaches based on contour evolution [19], [4], [9], in particular using the level set method [29], [43], [37], [7]. Such methods are bound to find local optima of the energy they are optimizing and heavily depend on the initial contour. In addition, they are based on rather simple shape similarity measures which do not attempt to establish correspondences of parts or points.

Recently Cremers et al. [8] dealt with the first point: starting from an implicit representation of shapes and segmentations, they are able to find globally optimal segmentations while taking into account shape similarity to a number of training shapes. The lack of point correspondences remains, however.

Fig. 1. Tracking a car in bad weather: Despite bad visibility, reflections and camera shake, the proposed method allows reliable tracking over a hundreds of frames.

In contrast, a number of discrete approaches do allow shape priors based on point correspondences while guaranteeing global optimality: Coughlan et al. [5] are able to match open contours to images, taking into account an elastic shape similarity measure [31], [32]. Being based on dynamic programming, the method is in principle parallelizable. However, it is limited to open contours and hence does not provide a segmentation. Although the method could be extended to closed contours by performing a complete search over the start point, in practice this would be far too time/consuming.

The first globally optimal shape-based segmentation algorithm was proposed by Felzenszwalb [13]. It is based on dynamic programming in chordal graphs. The algorithm is easy to parallelize and invariant with respect to translation, rotation and scale changes. In practice, however, due to its quadratic memory complexity, pixel-accurate segmentations can only be computed on rather coarse resolution.

B. Contribution

In this paper we present an effectively linear-time algorithm to match contours to images. For the first time, the globally optimal matching of closed contours to images becomes feasible in a matter of seconds (on current graphics hardware). While the employed model bears resemblance to the one in [5], we address the computationally more challenging case of closed contours and reduce the bias towards short curves by reverting to ratio functionals and minimum ratio cycle computation.

The proposed method supports different amounts of invariances, including translational and rotational ones. By exploiting its high parallelizability real-time tracking becomes feasible.

Preliminary versions of this paper appeared in [38] for image segmentation and [39] for tracking. The present paper contains a more in-depth description of the system as well as a refined discretization of the underlying continuous functional. In the experimental section we also rely on results of our recent work [41] on finding global geodesics.

II. Matchings as Cycles in a Product Space

We are given a prior contour $\mathbf{S} : \mathbb{S}^1 \to I\!R^2$ (where $\mathbb{S}^1$ is the unit circle) with a uniform parameterization. The task is to match this contour to a given image $I : \Omega \to I\!R$, where $\Omega \subseteq I\!R^2$ is the (typically rectangular) image domain. The placed contour $\mathbf{C} : \mathbb{S}^1 \to \Omega$ should be similar to the input contour and fulfill some data-driven criteria. In this work we want it to be located at image edges.

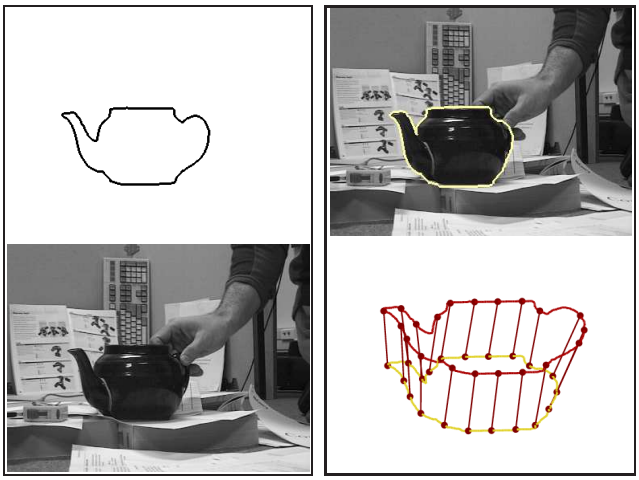


Fig. 2. Starting from a prior contour (top left) and an input image (bottom left), the proposed method simultaneously locates the (possibly deformed) contour in the image (top right) and computes a correspondence function between the two curves (bottom right).

Figure 2 gives an illustration of our approach: when input a contour and an image, the algorithm locates a deformed version of the contour in the image and computes an alignment to the prior contour. Mathematically this alignment is an orientation-preserving function $m : \mathbb{S}^1 \to \mathbb{S}^1$ which puts the points on both curves into correspondence. This allows to use correspondence-based shape similarity measures which were shown to be important for reproducing human notions of shape similarity [18], [26].

In this paper the optimal pair of m and $\mathbf{C}$ is characterized as the global optimum of a ratio energy. We present a method which allows to find the global optima of a large variety of cost functions.

The key idea and contribution of this paper is to cast the problem as an optimization problem over cycles in a product space. While this was known for open curves [5], the computationally much more challenging case of closed curves has so far not been solved.

The product space arises by combining the functions $\mathbf{C}$ and m into a single function

$$\mathbf{\Gamma} : \mathbb{S}^1 \to \Omega \times \mathbb{S}^1$$

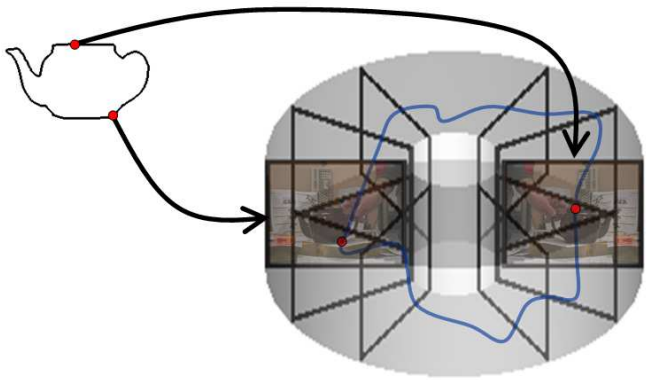


Fig. 3. Cyclic paths in a $3D$ graph (no edges are shown): For any point on the prior contour there are K copies of the image in the graph. Any assignment of pixels in the image to corresponding points on the template contour corresponds to a cyclic path in this graph.

which is called a *cycle*. The space in which these cycles live is visualized in Figure 3. It has the form of a torus and arises by placing a copy of the image for each point on the (one-dimensional) prior contour. When splitting a closed contour at some point, it can be viewed as an open one. The space would then be a solid block. When additionally imposing that start and end point are identical, the respective end faces of the block have to meet and the torus is formed.

A curve (with winding number 1) in this space now allows to read off the desired information: the curve $\mathbf{C}$ is obtained by projecting $\mathbf{\Gamma}$ to the first two dimensions. The correspondences of the points on $\mathbf{C}$ can be read off in the third dimension.

III. Assigning a Cost to each Cycle

We now present an exemplary energy functional for matching shapes to images. The presented method applies to a much larger class of functionals. For example, in [40] we used a more sophisticated data term based on patch comparison. Before we state the cost function, we briefly discuss how curves are represented.

A. Representing curves

There are infinitely many ways to parameterize a specific curve. Naturally an optimization problem should not depend on the chosen parameterization.

The functional we consider in this paper is indeed invariant with respect to re-parameterizations. For most of this section we will therefore not assume any specific parameterization of the contour $\mathbf{C}$ to be optimized. Yet, in a few places it will be convenient to have a uniform parameterization, i.e. with constant derivative $\|\mathbf{C}_s(s)\| = \|\mathbf{C}\|$ everywhere. In the given setting the correspondence function m is dependent on the contour $\mathbf{C}$: $m(s)$ will always denote the correspondence of the point $\mathbf{C}(s)$. Hence, if the parameterization of $\mathbf{C}$ is changed, the function m changes as well.

In subsequent sections we prefer the combined function $\mathbf{\Gamma} : \mathbb{S}^1 \to \Omega \times \mathbb{S}^1$. Since the objective function is *not* invariant against re-parameterizations of $\mathbf{\Gamma}$ (data term and shape measure are not coupled), we state it in terms of $\mathbf{C}$ and m.

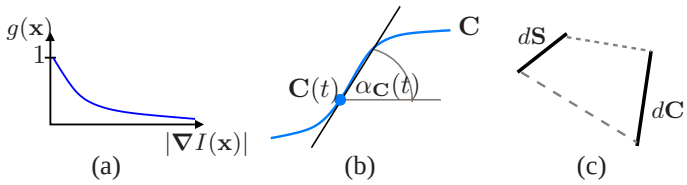


Fig. 4. The three ingredients of the proposed method: (a) an edge detector function assigning low values to high image gradients. (b) computation of tangent angles of the contours $\mathbf{C}$ and $\mathbf{S}$ (shown for $\mathbf{C}$, the tangent is drawn in black). (c) computation of length distortion.

We allow self-intersecting contours $\mathbf{C}$ since we have no means to exclude them. We found them to arise only seldomly as long as the desired object is truly contained in the image.

B. Characterizing the Optimal Matching

The optimal contour is characterized as the global minimum of a ratio energy. It combines three terms which are visualized in Figure 4 and now described in greater detail:

1) **Data term.** The proposed method supports a variety of different data terms. In [40] we used a patch comparison along the contour. Since the contribution of the present work lies in the integration of shape knowledge into globally optimal image segmentation, we restrict ourselves to a simple data term here. It exploits the knowledge that region boundaries typically coincide with image edges and is based on the simple edge detector
$$g(\mathbf{x}) = \frac{1}{1 + |\nabla I(\mathbf{x})|}, \tag{1}$$
as shown in Figure 4a. Integrating such a positive function along the contour $\mathbf{C}$ entails a strong bias towards short curves. To alleviate the problem we normalize by the contour length, which results in averaging the edge detector function $g(\cdot)$ along the contour $\mathbf{C}$:
$$\frac{\int_{\mathbb{S}^1} g(\mathbf{C}(s)) \, \|\mathbf{C}_s(s)\| \, ds}{\int_{\mathbb{S}^1} \|\mathbf{C}_s(s)\| \, ds} = <g>_{\mathbf{C}} \tag{2}$$
Shorter curves are still favored, but not as strongly as before. The remaining bias is counteracted by the shape similarity measure.
2) **Similarity of Curve Attributes.** This is the first of two terms constituting the shape similarity measure. It consists of a comparison of local attributes of corresponding points $\mathbf{C}(s)$ and $\mathbf{S}(m(s))$. In this work – as shown in Figure 4b – we choose the tangent angles of the curves as attributes. This implies translational invariance since translating a contour does not change its tangent angles. The integration of rotational invariance requires extra effort. This is deferred to Section VI-C.
The similarity of corresponding tangent angles is measured by their squared cyclic difference $|\alpha_{\mathbf{C}}(s) - \alpha_{\mathbf{S}}(m(s))|^2_{\mathbb{S}^1}$, where the difference is taken on the manifold $\mathbb{S}^1$. Again we divide by the contour length and get

the average deformation cost

$$\frac{\int\limits_{\mathbb{S}^1} \left|\alpha_{\mathbf{C}}(s) - \alpha_{\mathbf{S}}(m(s))\right|^2_{\mathbb{S}^1} \|\mathbf{C}_s(s)\|\, ds}{\int\limits_{\mathbb{S}^1} \|\mathbf{C}_s(s)\|\, ds} = \int\limits_{\mathbb{S}^1} \left|\alpha_{\mathbf{C}}(s) - \alpha_{\mathbf{S}}(m(s))\right|^2_{\mathbb{S}^1} ds \qquad (3)$$

where the second line holds for a uniform parameterization of $\mathbf{C}$.

3) **Penalties for Stretching and Shrinking.**
Aside from the monotonicity of the correspondence function, another regularity assumption is made: the local stretching and shrinking of the contour are disfavored. The functional hence favors curves which preserve the scale of the prior contour. This counteracts the shrinking bias of the data term. In practice we observe a robustness with respect to gradual scale changes.
The amount of stretching and shrinking is characterized as length distortion: consider Figure 4c where a piece $d\mathbf{S}$ of $\mathbf{S}$ corresponds to a piece $d\mathbf{C}$ of $\mathbf{C}$, say at the point $\mathbf{C}(s)$. Then the length distortion is given by the quotient $|d\mathbf{S}|/|d\mathbf{C}|$. With a uniform parameterization of $\mathbf{S}$ this quotient can be expressed as[1]

$$\frac{|d\mathbf{S}|}{|d\mathbf{C}|}(s) = \frac{\|\mathbf{S}\|\,|m_s|}{\|\mathbf{C}_s(s)\|}$$

Numerous ways to penalize length distortion are conceivable. Before we explain our choice we state a few properties we consider essential for a penalty function:

a) A ratio of 1 (no distortion) should be assigned the penalty 0.
b) As the ratio approaches ∞ (i.e. $|d\mathbf{C}| \to 0$ for fixed $|d\mathbf{S}|$), so should the corresponding penalty. This point is crucial as the edge-based data term favors short curves and the shape attribute comparison is independent of scale.
c) Preferably the shape similarity measure should be symmetric, i.e. comparing $\mathbf{C}$ to $\mathbf{S}$ should give the same cost as comparing $\mathbf{S}$ to $\mathbf{C}$. This implies that the ratio $|d\mathbf{S}|/|d\mathbf{C}|$ should be given the same penalty as its inverse $|d\mathbf{C}|/|d\mathbf{S}|$.

The penalty function chosen in this work satisfies all three requirements:

$$\Psi\left(r = \frac{|d\mathbf{S}|}{|d\mathbf{C}|}\right) = \begin{cases} r - 1 & \text{if } K \geq r \geq 1 \\ r^{-1} - 1 & \text{if } \frac{1}{K} \leq r < 1 \\ \infty & \text{otherwise} \end{cases}, \qquad (4)$$

Here $K \in \mathbb{N}$ is a predefined constant limiting the maximum length distortion. This limit will be exploited in the optimization algorithm.

[1]This term is indeed invariant against re-parameterization of $\mathbf{C}$: any such re-parameterization will also change the correspondence function m and with it its derivative.

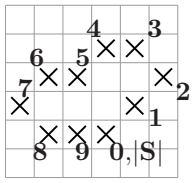


Fig. 5. A shape is represented as a sequence of pixels on a discrete grid.

Like the two previous terms, the corresponding regularity term takes on the form of a ratio:

$$\frac{\int\limits_{\mathbb{S}^1} \Psi\left(\frac{\|\mathbf{S}\|\,|m_s|}{\|\mathbf{C}_s(s)\|}\right) \|\mathbf{C}_s(s)\|\, ds}{\int\limits_{\mathbb{S}^1} \|\mathbf{C}_s(s)\|\, ds} = \int\limits_{\mathbb{S}^1} \Psi\left(\frac{\|\mathbf{S}\|\,|m_s|}{\|\mathbf{C}_s(s)\|}\right) ds \qquad (5)$$

where again equivalence holds for a uniform parameterization of $\mathbf{C}$. This term is scale-invariant in the sense that scaling both $\mathbf{C}$ and $\mathbf{S}$ by the *same* amount does not affect the cost.

Using positive weighting factors $\lambda, \nu > 0$ these terms are glued together into a single functional:

$$\int\limits_{\mathbb{S}^1} g(\mathbf{C}(s))\, ds \;+\; \nu \int\limits_{\mathbb{S}^1} \left|\alpha_{\mathbf{C}}(s) - \alpha_{\mathbf{S}}(m(s))\right|^2_{\mathbb{S}^1} ds + \lambda \int\limits_{\mathbb{S}^1} \Psi\left(\frac{\|\mathbf{S}\|\,|m_s|}{\|\mathbf{C}\|}\right) ds \qquad (6)$$

We state this for a uniform parameterization of $\mathbf{C}$. The respective terms for arbitrary parameterizations are given in the discussion above.

IV. Discretizing Cost and Product Space

To optimize over the cycles $\mathbf{\Gamma} : \mathbb{S}^1 \to \Omega \times \mathbb{S}^1$, both the cost function and the product space are discretized. This section deals with the discretization, the optimization algorithm is detailed in the next one.

The key idea is to represent $\mathbf{C}$ as a polygonal curve with (an a priori unknown number of) vertices on the pixel grid. In addition the correspondence m is assumed to be linear along each polygonal line segment. It is therefore uniquely defined by assigning point correspondences to the two end points of such a segment. Specifically we consider line segments connecting neighboring pixels on the pixel grid, where we choose an 8-neighborhood.

A cycle $\mathbf{\Gamma}$ can now be composed out of a finite set of basic parts $\Delta\mathbf{\Gamma} = (\Delta\mathbf{C}, \Delta m)$.

A. Discretizing Prior Contour and Correspondence

In addition to the cycle $\mathbf{\Gamma}$, also the prior contour $\mathbf{S}$ is discretized. As shown in Figure 5, we represent it in the same form as the contour $\mathbf{C}$, i.e. as an ordered set $\mathbf{s}_0, \ldots \mathbf{s}_{|\mathbf{S}|}$ of points on a suitable pixel grid, where – for ease of notation – $\mathbf{s}_0 = \mathbf{s}_{|\mathbf{S}|}$ is represented twice. To get a dense representation

of the contour we require that $\mathbf{s}_i$ be among the eight closest neighbors of $\mathbf{s}_{i-1}$.

The discrete correspondence function assigns each image pixel on $\mathbf{C}$ one of these $|\mathbf{S}|+1$ prior points. To ensure a monotone matching we enforce that the start pixel of a segment $\Delta\mathbf{C}$ is assigned a shape point with index lower or equal to that of the endpoint. Closure of the matching is obtained by the fact that $\mathbf{s}_{|\mathbf{S}|} = \mathbf{s}_0$.

The length distortion penalizer (4) gives two hard constraints which limit the minimal and maximal distortion ratio. In the discrete setting these are realized in terms of the indices of the two shape points assigned to a line segment. The upper limit corresponds to an index difference of at most K. Ensuring the lower limit is more intricate since here several line segments $\Delta\mathbf{C}$ may correspond to the same part $\mathbf{s}_{i-1}\mathbf{s}_i$ of $\mathbf{S}$. We therefore allow the two indices to be equal. However, for any shape point $\mathbf{s}_i$ there may be at most K parts $\Delta\mathbf{\Gamma}$ where both endpoints of ΔC correspond to $\mathbf{s}_i$.

In practice this is realized by modifying the correspondence function m: in the discrete setting m maps to pairs (i,k) where i gives the shape point and $k<K$ gives the number of image pixels already corresponding to $\mathbf{s}_i$. If m maps to the same i at the beginning and end of the contour segment, then the index k must be one higher for the end node. This is formalized in the following section.

B. From Product Space to Product Graph

With the described parts $\Delta\mathbf{\Gamma}$ it is straightforward to discretize the product space $\Omega\times\mathbb{S}^1$ into a graph. The pieces $\Delta\mathbf{\Gamma}$ serve as edges in this product graph: a part is the connection of two image pixels with assigned point correspondences for each pixel. Combinations of an image pixel and a point correspondence can be represented in the state space

$$\mathcal{V} = \mathcal{P}\times\{0,\dots,K\cdot|\mathbf{S}|\}$$

where $\mathcal{P}$ is the set of image pixels. In this setting, consider a part $\Delta\mathbf{\Gamma} = (\Delta\mathbf{C}, \Delta m)$ where $\Delta\mathbf{C}$ connects the pixels $\mathbf{p}$ and $\mathbf{q}$ with $\mathbf{p}$ corresponding to (i,k) and $\mathbf{q}$ corresponding to (i',k'). This part can be seen as a directed edge

$$(\mathbf{p}, i\cdot K+k) \to (\mathbf{q}, i'\cdot K+k')$$

By construction we have $\mathbf{s}_0 = \mathbf{s}_{|\mathbf{S}|}$, so the indices 0 and $|\mathbf{S}|$ are merged into a single state. The result is a torus-like graph (of the same form as the product space depicted in Figure 3), where each point on the prior contour is assigned K frames of nodes, each containing one node for every pixel in the image.

For a set of parts (or edges) $\Delta\mathbf{\Gamma}$ to represent a continuous cycle $\mathbf{\Gamma}$, two properties must be fulfilled: (1) for each edge ending in a node, there must be an edge starting in the node and (2) for each (i,k) there may be at most one edge whose start node has (i,k) as second index.

Property (1) is the usual characterization of a cycle in a graph, whereas property (2) states that a cycle may not wrap around multiple times in the torus-graph. Computing the optimal $\mathbf{\Gamma}$ therefore corresponds to finding the optimal cycle in a certain subset of all cycles of the torus graph.

C. Setting up the Edge Weights

So far we have not considered how to represent the cost of a cycle $\mathbf{\Gamma}$. To this end each edge $\Delta\mathbf{\Gamma}$ is assigned a numerator weight n and a denominator weight d which represent the respective integral of the cost function along $\Delta\mathbf{\Gamma}$. The denominator weight always corresponds to the length of the line segment $\Delta\mathbf{C}$.

Setting up the numerator weights is more intricate. In this point we differ from the edge weights given in [38]: We give a refined discretization of the length distortion penalty, which better approximates the continuous functional.

In the graph we have two types of edges, called type I and type II. Type I edges correspond to the assignment of a new shape pixel. They are of the form

$$(\mathbf{type\ I}) \quad (\mathbf{p}, i\cdot K+k) \to (\mathbf{q}, j\cdot K) \quad \text{with } i<j\le i+K$$

where $0\le i,j<|\mathbf{S}|$ and $0\le k<K$ are integers and $\mathbf{p}$ and $\mathbf{q}$ are neighboring pixels in the image. Edges of type two correspond to assigning an entire line segment to the same shape point. They are of the form

$$(\mathbf{type\ II}) \quad (\mathbf{p}, j\cdot K+k) \to (\mathbf{q}, j\cdot K+k+1) \quad \text{with } k+1<K$$

The numerator weights for these edges are composed as follows:

1) **Data term.** For both types of edges, the data term contributes
$$\frac{1}{2}\|\mathbf{p}-\mathbf{q}\|\ (g(\mathbf{p})+g(\mathbf{q})).$$
2) **Comparison of Tangent Angles** We denote by $\varphi(\mathbf{x},\mathbf{y})$ the tangent angle of the line segment connecting $\mathbf{x}$ and $\mathbf{y}$. It can be computed via the function `atan2` contained in most standard programming languages. The contribution of this term is then
$$\nu\,\|\mathbf{p}-\mathbf{q}\|\ \left|\varphi(\mathbf{p},\mathbf{q})-\varphi(\mathbf{s}_j,\mathbf{s}_{j-1})\right|^2_{\mathbb{S}^1},$$
again for edges of both types.
3) **Length Distortion** A simple discretization of the length distortion penalty would be to add a term λ for type II edges and $\lambda(j-i-1)$ for type I edges. Both could be further multiplied with the distance $\|\mathbf{p}-\mathbf{q}\|$. We used this discretization in previous publications.
The problem with this discretization is that it assigns no cost when a diagonal line (length $\sqrt{2}$) in the image is matched to a horizontal or vertical line (length 1) in the prior template. We therefore now present a refined discretization which is closer to the continuous functional.
Type I Suppose the last aligned shape point was point i. For an edge of type I one either moves to the direct predecessor point $j=i+1$ or skips up to $K-1$ shape points. In both cases there can be length distortion. This distortion is computed as
$$r = \frac{|d\mathbf{S}|}{|d\mathbf{C}|} = \frac{\sum_{j'=i+1}^{j}\|\mathbf{s}_{j'}-\mathbf{s}_{j''-1}\|}{\|\mathbf{p}-\mathbf{q}\|}$$

It is straightforward to compute the penalty by evaluating the function (4) on this term. In fact, for edges of type I one could use any other penalty function.

Type II This case is more intricate as now several line segments are matched to the same line $\mathbf{s}_{j-1}\mathbf{s}_j$ of the prior template and the single evaluation of (4) has to be distributed over several edge weights. Here we exploit that once it is known that the length distortion ratio is below 1, the function (4) is linear in $r^{-1}-1$. Since all polygonal lines have length at least 1 and an edge of type I must already have been aligned to $\mathbf{s}_{j-1}\mathbf{s}_j$, it is safe to exploit this for all type II edges.

Let $\mathbf{p}_0\mathbf{p}_1,\ldots,\mathbf{p}_{k-1}\mathbf{p}_k$ be *all* line segments of $\mathbf{C}$ aligned to $\mathbf{s}_{j-1}\mathbf{s}_j$. Then the inverse ratio can be written as

$$r^{-1}=\frac{\sum_{k'=1}^{k}\|\mathbf{p}_{k'}-\mathbf{p}_{k'-1}\|}{\|\mathbf{s}_j-\mathbf{s}_{j-1}\|}=\sum_{k'=1}^{k}\frac{\|\mathbf{p}_{k'}-\mathbf{p}_{k'-1}\|}{\|\mathbf{s}_j-\mathbf{s}_{j-1}\|}$$

An edge of type II corresponds to only one of these line segments, so the aim is to write the penalty for this ratio as a sum over the line segments. If the ratio for the line $\mathbf{p}_0\mathbf{p}_1$ was one or greater the type II edge can simply be given the penalty term

$$\lambda\frac{\|\mathbf{p}-\mathbf{q}\|^2}{\|\mathbf{s}_j-\mathbf{s}_{j-1}\|}$$

Our present implementation simply assumes that the first ratio was at least one. The only case where this assumption is violated is when a horizontal or vertical line segment of $\mathbf{C}$ is matched to a diagonal one of $\mathbf{S}$. If one wants to handle this case correctly, the state space needs to be augmented. Type I edges with this property would then end in a new state. Type II edges leaving this state would be assigned different length distortion cost.

V. Computing the Optimal Matching

In Section IV-B it was shown that the set of all $\mathbf{\Gamma}$ (satisfying the discussed discretization) corresponds to a certain subset of all cycles in the torus-like graph spanned by image and prior template. Each edge e in the graph was given two weights $n(e)$ and $d(e)$, one reflecting the numerator integral for the corresponding line segment and matching, the other the denominator integral. Functional (6) can now be approximated as

$$E(\Gamma)=\frac{\sum_{e\in\Gamma}n(e)}{\sum_{e\in\Gamma}d(e)} \tag{7}$$

This function can be optimized over *all* cycles in the graph via the Minimum Ratio Cycle algorithm of Lawler [27]. This algorithm was introduced to computer vision by Jermyn and Ishikawa [24]. We will review this algorithm first, then discuss how to restrict the optimization to cycles that correspond to a valid continuous Γ.

A. Ratio-Optimization over all Cycles

Given arbitrary numerator weights $n(e)$ and non-negative denominator weights $d(e)$ where no cycle is assigned a denominator of 0, Lawler's algorithm finds the cycle of optimal ratio (7) in the graph.

The basic principle is negative cycle detection: let τ be some ratio and define edge weights

$$w(e)=n(e)-\tau\,d(e)$$

The following equivalence transformations show that τ is above the optimal ratio if and only if there is a negative cycle Γ w.r.t. $w(e)$:

$$\begin{aligned}
&\sum_{e\in\Gamma}w(e) &&<0\\
\Leftrightarrow\quad&\sum_{e\in\Gamma}[n(e)-\tau d(e)] &&<0\\
\Leftrightarrow\quad&\sum_{e\in\Gamma}n(e) &&<\tau\cdot\sum_{e\in\Gamma}d(e)\\
\Leftrightarrow\quad&\frac{\sum_{e\in\Gamma}n(e)}{\sum_{e\in\Gamma}d(e)} &&<\tau
\end{aligned}$$

Notice that the final transformation only holds since the denominator sum is positive for all cycles.

After initializing τ with some upper bound on the optimal ratio, the Minimum Ratio Cycle algorithm proceeds to check if a negative cycle exists. If one is found, τ is set to its ratio and the process is repeated. At some point, the graph will no longer contain negative cycles. The last found cycle must then be optimal: it has cost 0 for the current ratio and since there is no negative cycle, there cannot be a better one.

To achieve a (weakly) polynomial running time one additionally assumes that the edge weights $n(e)$ and $d(e)$ are integers. This guarantees a certain minimal distance between two distinct ratios [24] and a polynomial amount of iterations in the worst case. In practice we multiply the continuous weights given in the last section with a factor of 1000, then round to the nearest integer.

It remains to show how a negative cycle can be detected. This is done by a slight modification of the Moore-Bellman-Ford algorithm [16], [33], [1] for distance calculations in a graph with negative edge weights, but without negative cycles. Initially all distance labels are set to ∞, except for a given root node which is initialized with 0. Then repeatedly all nodes are checked for whether their distance label can be improved. If the graph does not have negative cycles, the process terminates in polynomial time. Otherwise it does not terminate and at some point the predecessor graph will permanently contain cycles. These cycles correspond to negative cycles and are found by regularly checking the predecessor graph for cycles.

B. Restriction to the Set of Valid Cycles

We are now in a position to find the optimal cycle in the graph. In about 2% of all cases this cycle does not correspond to a valid continuous Γ since it wraps around multiple times in the graph. A straightforward way to exclude this would be a complete search over the initial correspondence. The frame 0 would then contain only one node in each pass. This would however result in a quadratic run-time.

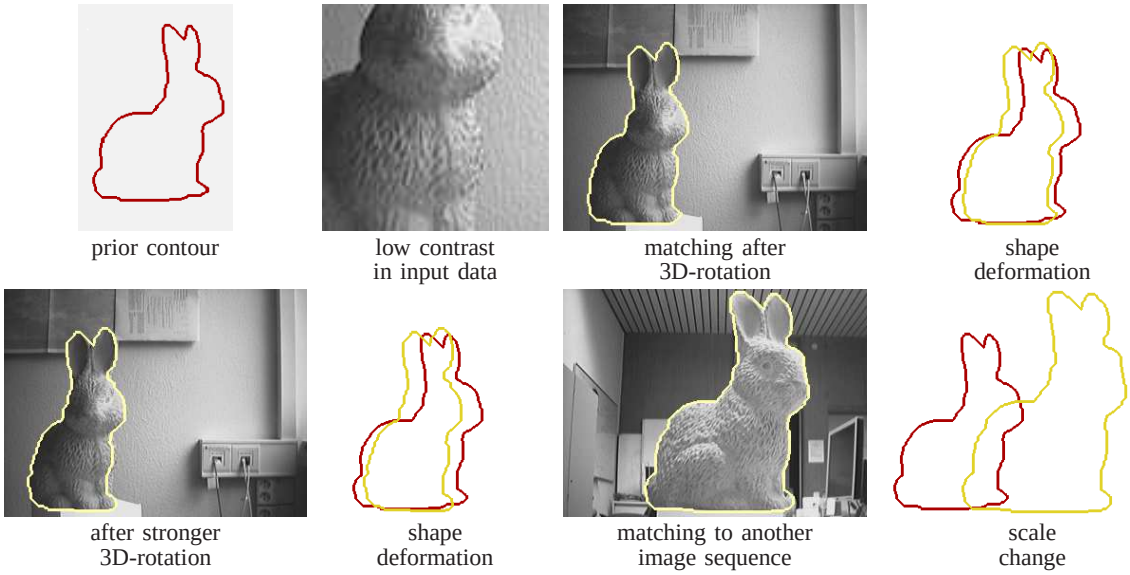


Fig. 6. Segmentation with a single template: Despite significant deformation, scale change and translation, the initial template curve (red) is accurately matched to each image.

Instead we employ a recursive splitting strategy. First a search for the optimal cycle in the graph is run as described in the previous section. If this cycle is valid one terminates. Otherwise frame 0 is split into as many parts as there are wrap-arounds in the found cycle, so that all nodes of frame 0 contained in the cycle are assigned to different components.

For each of these components a separate ratio minimization is run recursively on a graph with modified frame 0: this frame only contains the nodes of the component (in practice all nodes are kept and the distance labels of the other nodes in the frame are fixed to ∞). As initial upper bound the ratio of the last found valid cycle is taken (or the initial one if no valid cycle was found).

It is possible that these calls again find invalid optimal cycles. In this case further splits and recursive calls are made. In the end the globally best continuous Γ corresponds to the best cycle found in any of the components.

C. Complexity of the Method

To conclude this section, we analyze the running time of the described algorithm. In practice it is run with quantized numerator and denominator weights, i.e. all weights are multiples of some rational $\epsilon > 0$. We start by analyzing a single call of Lawler's algorithm. In such a call, each execution of the Moore-Bellman-Ford algorithm has a complexity of $\mathcal{O}(|\mathbf{S}|^2|\mathcal{P}|^2)$. From the bound given in [24] it follows that the employed linear search scheme invokes at most $\mathcal{O}(|\mathbf{S}|^3|\mathcal{P}|^3 w_d^2 w_n/\epsilon^3)$ calls of the Moore-Bellman-Ford algorithm, where w_n and w_d are the maximal numerator and denominator weights before quantization.

Finally, the recursive subdivision scheme makes $\mathcal{O}(|\mathcal{P}|^2)$ calls in the worst case. Together this results in a complexity of $\mathcal{O}(|\mathbf{S}|^5|\mathcal{P}|^7 w_d^2 w_n/\epsilon^3)$. The memory complexity is linear with $\mathcal{O}(|\mathbf{S}|\,|\mathcal{P}|)$.

As detailed in Section VIII-B, in practice we observe a linear run-time dependence: the Moore-Bellman-Ford algorithm usually terminates after having visited every node at most 10 times for all tested problem instances. During a call of Lawler's algorithm, normally less than 20 ratio adjustments are made. Moreover, there are usually less than 6 recursive calls in the subdivision schemes. These numbers already refer to rare cases, the algorithm is usually much faster.

In addition, the algorithm can be sped up by a constant factor. In particular the distance calculation can be parallelized. We will come back to this in Section VII, where we show that tracking can be solved in real-time. For now we present some results on image segmentation with prior knowledge.

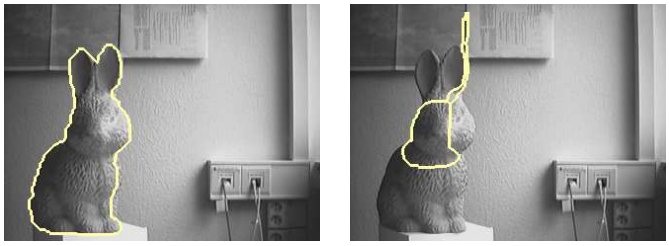
VI. Experiments I: Shape-based Image Segmentation

In this section we present results for image segmentation with translation invariance: the globally optimal matching of prior curves to a variety of images will be demonstrated on real-world data. We also treat the topic of how to incorporate rotational invariance.

We treat images with significant distortion. As a consequence we allow $K = 5$ image pixels to be matched to a single shape point and set a low length distortion weight with $\lambda = 0.1$. The tangent angles are given more weight with $\nu = 0.5$ – this term really drives the process.

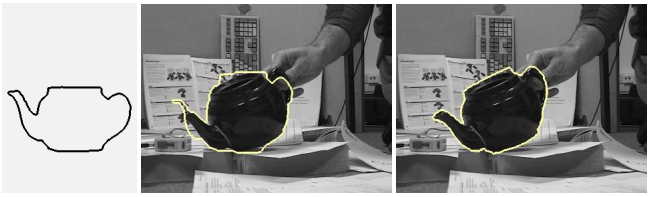
A. Translation-invariant Matching

In Figure 6 the contour of a rabbit (viewed from the side) is matched to images from two different sequences. In the first the rabbit is shown from different viewpoints but at the same scale. Despite low contrast between object and background the algorithm relocates the object reliably. We emphasize that despite the similar scale there is significant *local* length distortion. The second sequence demonstrates matching in the presence of a global scale change.



proposed method (with length normalization) | without length normalization

Fig. 7. The length normalization removes the bias towards shorter curves.



prior contour | no rotational invariance | with rotational invariance

Fig. 8. By optimizing over all possible rotations, the method becomes invariant to rotations of the shape, as well.

B. On the Effect of Length Normalization

In Section III-B we introduced the length normalization to reduce the bias towards shorter curves. This effect is demonstrated in Figure 7: the figure shows the global optima for the ratio functional and for the numerator integral alone. The latter corresponds to the geodesic energy we proposed in [41]:

$$E_{geo}(\mathbf{\Gamma}) = \|\mathbf{C}\| \cdot E(\mathbf{\Gamma})$$

and is minimized globally by a combination of branch and bound and shortest path algorithms.

Clearly the ratio functional produces longer curves. We observe this whenever there is low contrast in some regions along the desired curve.

C. Including Rotational Invariance

Aside from translational invariance, sometimes one also wants rotational invariance. The proposed framework can be easily extended to include this: one simply samples the rotation angle in sufficiently dense intervals. The prior contour is rotated by the specified amount and the obtained contour is matched to the image. When all angles have been processed, the match with lowest energy is output.

The run-time of this process depends linearly on the number of sampled angles. In practice, substantial speed-ups are gained by exploiting a property of the optimization algorithm: for the subsequent comparisons one can initialize the ratio with the last determined one.

Figure 8 shows an application for a sequence[2] containing a significant rotation of the object. Here we sample the rotation angle between $-90°$ and $+90°$ in steps of $2°$. Where translational invariance alone failed, the algorithm now finds the desired solution.

[2]Image data courtesy of Bodo Rosenhahn.

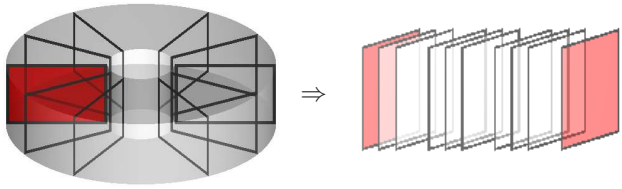


Fig. 9. The graph is cut open at frame 0 (shown in red) and the frame is doubled. The arising graph is acyclic. This is the key for efficient optimization.

VII. Speeding up the Algorithm

It will now be detailed how to obtain an efficient implementation of the described algorithm, where we consider both memory efficiency and run-time. Some points will apply to any kind of hardware platform (sequential or parallel) whereas others are tailored to parallel platforms.

A. Efficient Memory Management

We described the algorithm in terms of a graph. Yet, due to the large search space standard graph representations are not sensible: they would easily require TeraBytes of memory. Below we describe three points to reduce the memory consumption to an amenable level. Together they allow to process images of size 320×240 with roughly one GigaByte of memory.

1) Implicit storage: It turns out that we can use a much smarter representation: although the weights are assigned to edges, the optimization algorithm stores only distance labels for *nodes*. Edges can hence be computed on-the-fly, which in easy due to the regular structure of the nodes. This is a significant advantage over maxflow-based algorithms which store intermediate flows at the edges.

Distance labels are conveniently stored in matrices. In addition one needs to store predecessor entries for each node. These only need to code the number of the incoming edge. For $K \leq 5$ this can be encoded in one byte, otherwise two bytes are needed. We used four bytes in our implementation.

2) Sweep-based distance calculation: To obtain further speed-ups we implement the distance calculation by a sequence of distance calculations on a related, acyclic graph. This graph is obtained by introducing a copy of frame 0, assigned the number $|\mathbf{S}| \cdot K$. All edges previously ending in frame 0 are connected to the respective nodes in the new frame. This is visualized in Figure 9.

Now the distance calculation can be performed in sweeps. For the first sweep all distance labels in frame 0 are initialized with 0. Then, in each sweep the distance labels and predecessor entries for the frames 1 to $|\mathbf{S}| \cdot K$ are determined. Since the graph is acyclic and no edges connect nodes in the same frame, this can be done by dynamic programming. After each sweep the distance labels in frame 0 are compared to the respective ones in frame $|\mathbf{S}| \cdot K$. If the latter label is below the label in frame 0, this label and the corresponding predecessor entry are updated. Simultaneously the optimal path ending in a node is backtraced. If a (possibly invalid) cycle is found it must be negative (since the initial distance was 0) and the ratio is updated.

Otherwise the ratio is kept and possibly another sweep has to be performed: this is the case if one of the distance labels in frame 0 was updated. In practice we seldom observe more than three sweeps for one ratio.

3) Intermediate Storage: The final storage-based improvement is less obvious but very important in practice. Before moving to the details we first observe that during a sweep one does not need all distance labels: the dynamic programming algorithm processes frames in the order of their number. Since no edge skips more than K^2 frames, one only needs to store the last K^2 distance matrices.

Less obvious, but quite important, is the observation that in fact only $2K$ instead of K^2 distance matrices need to be stored. Recall that edges of type I are of the form $(\mathbf{p}, i \cdot K + k) \to (\mathbf{q}, j \cdot K)$. The important point is that the corresponding edge weights do *not* depend on k. In the distance calculation only the best of these K edges is needed. It corresponds to the optimal start node. This node need be computed only once for each shape point i and each pixel. Since the decision is needed for the following K shape points, this precomputation not only saves memory, but also gives considerable savings in run-time.

Taking all these improvements together, images of size 376×284 are processed with less than 750 MegaByte.

B. Parallel Implementation

In the previous section we introduced an auxiliary acyclic graph. This graph has the nice property that edges never link two nodes in the same frame. The distances for an entire frame can therefore be determined in parallel. We have implemented this on a graphics card with 128 parallel threads, using the CUDA framework. This reduces the run-times by a factor[3] of about 12.

C. Profiting from Smart Initializations

Our final optimization is to reduce the number of ratio adjustments in the ratio minimization. In practice this number depends on the quality of the initial upper bound. If the prior contour fits entirely into the image, such a bound is easy to determine computationally: one can simply try several placements of the undeformed contour and calculate the respective ratios.

We have made particularly good experiences for the problem of tracking, where we try displacements of up to 5 pixels in each direction. For the considered problems this resulted in no more than one ratio adjustment per frame, with only minimal computational overhead. The results are given in the next section.

VIII. Experiments II: Shape-based Tracking

In this section we presents results for the problem of tracking deformable objects (or contours). In the first frame the contour $\mathbf{S}$ is given. Then, subsequently we map the contour determined for the previous frame to the current frame. This performs better than keeping a fixed template since large-scale deformations are decomposed into a sequence of smaller ones.

In contrast to image segmentation, for tracking translation invariance is generally not desired. Instead temporal coherence is exploited, i.e. the knowledge that the next contour will be close to the previous contour. We therefore extend functional (6) to include a motion penalty:

$$\min_{\Gamma=(\mathbf{C},m)} \int_{\mathbb{S}^1} g(\mathbf{C}(s))\, ds \;+\; \nu \int_{\mathbb{S}^1} \left|\alpha_{\mathbf{C}}(s) - \alpha_{\mathbf{S}}(m(s))\right|^2_{\mathbb{S}^1} ds \\ + \lambda \int_{\mathbb{S}^1} \Psi\left(\frac{\|\mathbf{S}\|\,|m_s|}{\|\mathbf{C}\|}\right) ds \;+\; \int_{\mathbb{S}^1} \rho(\mathbf{C}(s), \mathbf{S}(m(s)))\, ds \quad (8)$$

Here ρ is a penalty function for the movement of points on the contour $\mathbf{C}$. Our framework makes no assumptions on this function - it can be non-convex, negative and need not be differentiable.

We experimented with several motion penalties and found it best to simply limit the maximal motion and treat all remaining motions equally:

$$\rho(\mathbf{x}, \mathbf{y}) = \begin{cases} 0 & \text{if } \|\mathbf{x} - \mathbf{y}\|_\infty \leq D_{\text{max}} \\ \infty & \text{else} \end{cases}$$

The limiting distance D_{max} is set between 10 and 20 pixels in practice. A value of 15 performs well on real-world traffic sequences and gives real-time performance in some cases. This performance is due to the optimizations presented in the previous section, but also due to the structure of ρ: the limit on the distance allows to shrink the frames to a small window each.

All experiments were carried out on a Core2 Duo machine with 2.66 GHz. The machine is equipped with a GTX 8800 graphics card which is accessed via the CUDA interface. We give run-times for a sequential implementation on a single core and a parallel version on the GPU.

A. Fast Tracking with Real-time Potential

We first present tracking results on challenging real-world traffic data, where the algorithm has to deal with difficult weather conditions such as rain, shadows and sunlight falling directly into the camera. In addition to the poor signal quality, one also has to deal with varying shutter times. Nevertheless, the proposed combination of edge-based data terms and shape consistency performs very robustly on these data.

In some cases this is already possible in real-time: the sequence in Figure 1 (on page 2) was recorded with 25 fps and is processed at the same frame-rate on the GPU.

Since the run-time depends on the length of the prior contour, bigger objects cannot yet be processed in real-time. The sequence in Figure 10 runs at 3 fps. Here the car is tracked over the entire sequence (roughly 140 frames), despite contour deformation, scale changes and shadows.

Our final experiment in Figure 11 shows the tracking of a transparent bottle in the presence of camera roll, where we are using a maximal displacement of $D_{\text{max}} = 25$.

[3] In our conference paper [38] we erroneously reported a factor of 300. The confusion is probably due to an inappropriate choice of data structures in the CPU code used for [38].

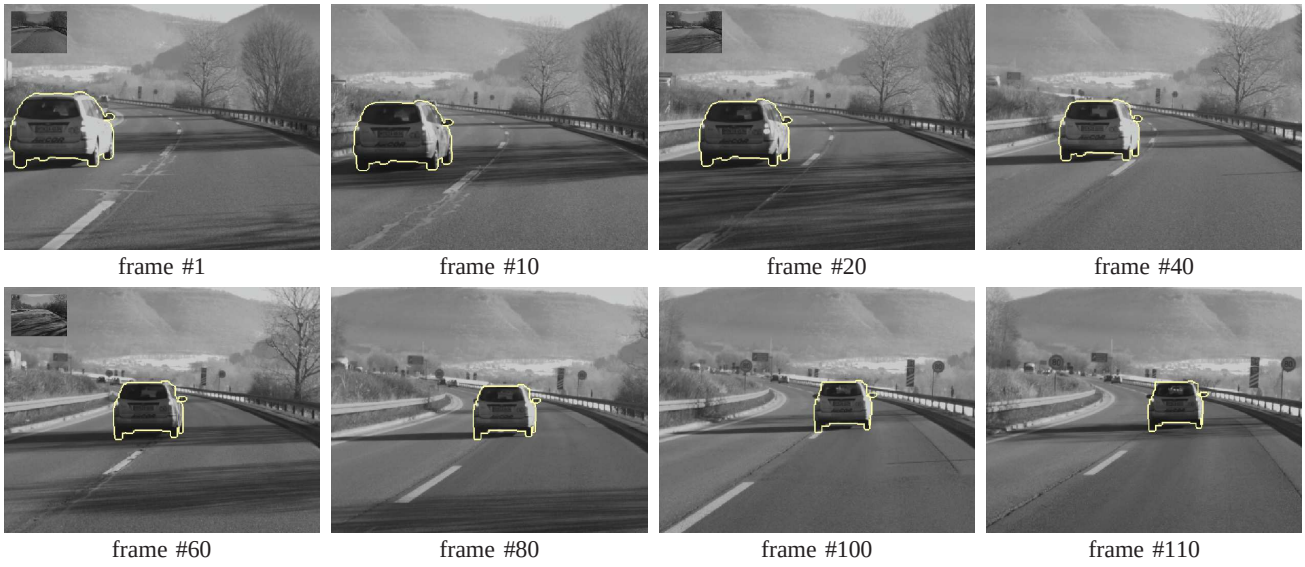


Fig. 10. Stable tracking of deforming silhouettes. The tracking performance is quite robust to changing lighting conditions (caused by shadows and varying shutter times).

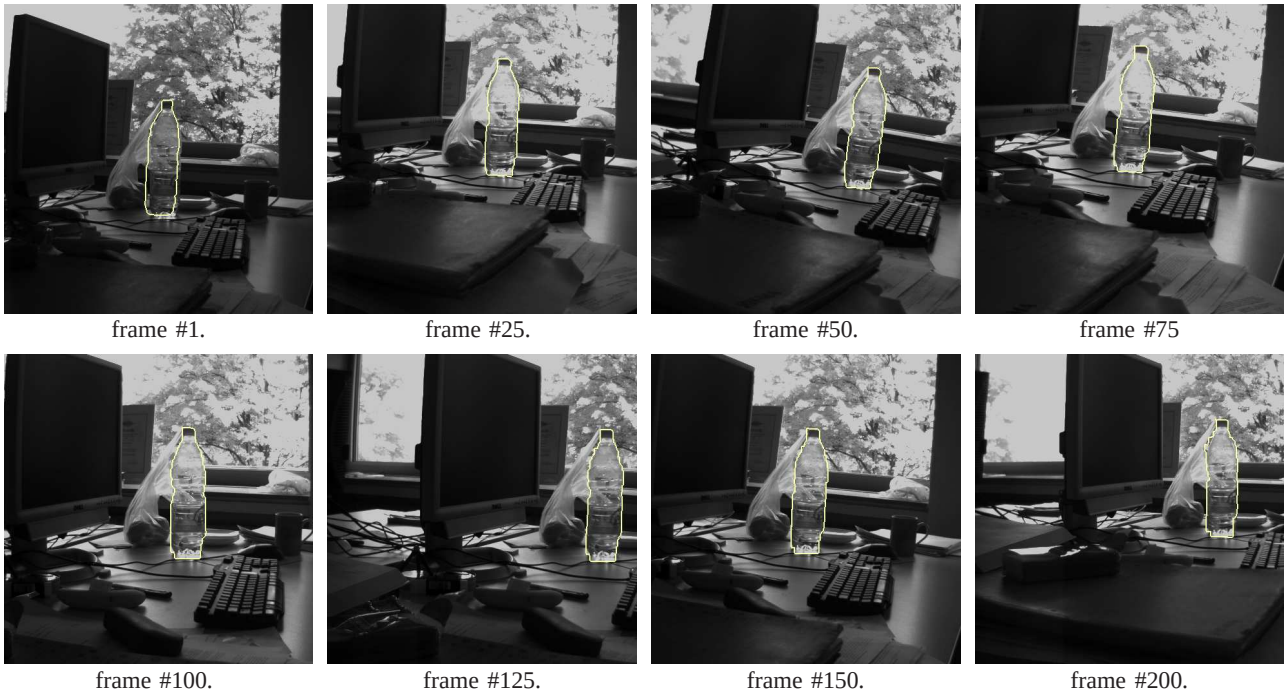


Fig. 11. Even transparent objects like the bottle can be tracked over several hundred frames despite camera rotation and substantial background clutter.

B. Run-time Analysis

To analyze the dependence of the running time of the algorithm on the input size, we processed the first 25 frames of the sequence in Figure 11 with varying window sizes. These experiments were run on a GTX 280 card and a 2.66 Ghz QuadCore computer (where only a single core was used).

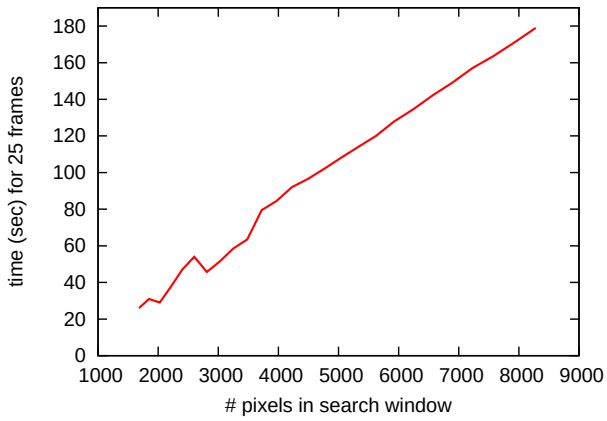
The resulting run-times are plotted in Figure 12. Clearly the dependence is far from quadratic and very close to linear.

Finally, as for tracking one has a rather small number of threads, the CPU may be faster than the GPU. For the sequence in Figure 1 this is evaluated for various window sizes in Figure 13: as soon as the size of the search window exceeds $D_{\text{max}} = 10$, the GPU implementation is faster.

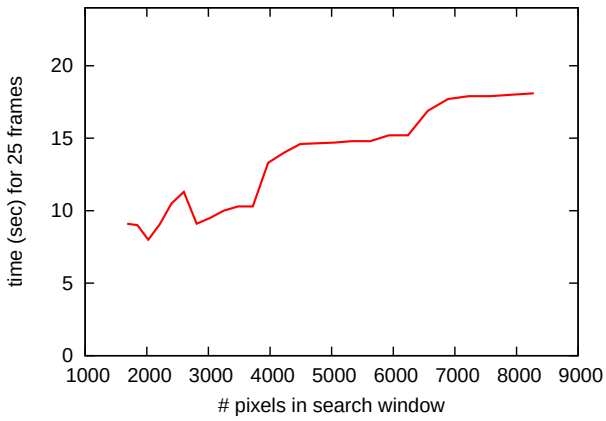
C. Comparison to Other Methods

We compare the proposed method to two other shape-based tracking methods, based on the level set method for local contour evolution. Approaches based on this method usually use region-based data terms since they are less likely to provide poor local optima.

We select two different data terms. The first one models a non-parametric intensity distribution via histograms. This proved robust for the application in [36], but in our case the object is lost after 4 frames already. The approach is stabilized by a patch-based data term, but after 15 frames the object is lost, too. In contrast, the proposed method is successful over the entire 100 frames of the sequence.



Run-time dependence on the CPU.



Run-time dependence on the GPU.

Fig. 12. Dependence of the run-time on the size of the search space (resulting from choosing $D_{\max} \in [20, 45]$): clearly the practical running times are sub-quadratic, both on CPU and GPU. These run-times are for tracking the first 25 frames of the bottle sequence in Figure 11.

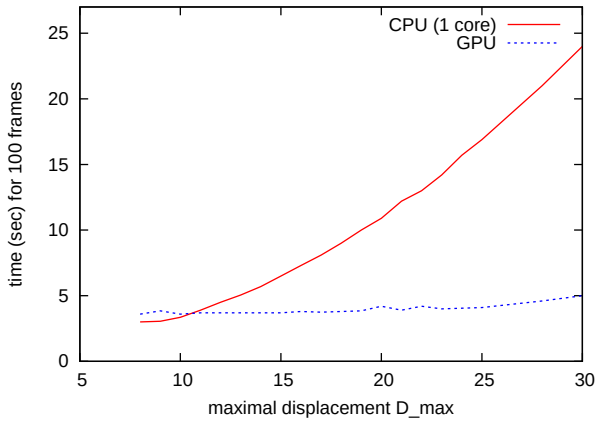


Fig. 13. Run-times for a sequential and a parallel platform for the sequence in Figure 1. The plot shows the dependence of running-times on the maximal displacement $D_{\max}$. Note that the number of pixels in the search window increases quadratically with $D_{\max}$.

Method in [36] on frames 1, 5 and 15:
the object is lost almost immediately.

Results with patch comparison (same frames):
here, too, the object is lost soon.

In contrast, tracking with the proposed
method is stable.

Fig. 14. While both simple and sophisticated methods fail after a few frames, the proposed method tracks the object over the entire one hundred frames - in real-time.

IX. Conclusion

In this paper we introduced a polynomial-time solution for matching a given contour to an image despite translation, rotation, scale change and deformation. The central idea is to cast the assignment of an image pixel to each template point as a problem of finding optimal ratio cycles in a $3D$ graph that represents the product space of image and template. The energy that is optimized globally consists of an edge-based data term and a shape similiarity measure favoring similarity of local edge angles and minimal distortion (stretching/shrinking) of the template curve.

On a variety of challenging segmentation and tracking tasks the proposed algorithm provides reliable results. Specifically we showed that one can track cars or even semi-transparent objects over hundreds of frames despite camera motion, camera shake, changing illumination and prominent background clutter.

In contrast to most existing methods for real-time tracking, the proposed method is globally optimal: Among all conceivable solutions the algorithm provides the best one in polynomial time. Experimentally we observed that the algorithm is effectively linear. In addition, real-time performance can be obtained by imposing an upper bound on the maximal velocity, constructing a smart initialization and reverting to an efficient parallel implementation on state-of-the-art graphics hardware.

An extension of this algorithm which simultaneously infers the location of articulated parts by computing cyclic paths in a respective $4D$ graph was recently presented in [40].

ACKNOWLEDGMENTS

We thank Bodo Rosenhahn and Daimler Research for providing image data. We thank Thomas Pock for helpful comments on efficient GPU-programming and to Frank R. Schmidt for many fruitful discussions. This research was supported by the German Research Foundation, grants #CR-250/1-1 and #CR-250/2-1.

REFERENCES

[1] R.E. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.
[2] A. Blake and M. Isard. *Active Contours*. Springer Verlag, London, 1998.
[3] V. Caselles, R. Kimmel, G. Sapiro, and C. Sbert. Minimal surfaces based object segmentation. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 19(4):394–398, 1997.
[4] T. Cootes and C.J. Taylor. Active shape model search using local grey-level models: A quantitative evaluation. In *British Machine Vision Conference (BMVC)*, pages 639–648, 1993.
[5] J. Coughlan, A. Yuille, C. English, and D. Snow. Efficient deformable template detection and localization without user initialization. *Computer Vision and Image Understanding*, 78(3):303–319, 2000.
[6] D. Cremers. Dynamical statistical shape priors for level set based tracking. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 28(8):1262–1273, August 2006.
[7] D. Cremers, S.J. Osher, and S. Soatto. Kernel density estimation and intrinsic alignment for shape priors in level set segmentation. *Int. Jour. of Comp. Vision*, 69(3):335–351, September 2006.
[8] D. Cremers, F.R. Schmidt, and F. Barthel. Shape priors in variational image segmentation: Convexity, Lipschitz continuity and globally optimal solutions. In *IEEE Int. Conf. on Comp. Vision and Patt. Recog.*, Anchorage, Alaska, June 2008.
[9] D. Cremers, F. Tischhäuser, J. Weickert, and C. Schnörr. Diffusion snakes: Introducing statistical shape knowledge into the Mumford–Shah functional. *Int. Jour. of Comp. Vision*, 50(3):295–313, 2002.
[10] F. Dellaert and C. Thorpe. Robust car tracking using Kalman filtering and Bayesian templates. In *Conference on Intelligent Transportation Systems*, 1997.
[11] J. Denzler and H. Niemann. Active rays: Polar-transformed active contours for real-time contour tracking. *Real-Time Imaging*, 5:203 – 213, 1999.
[12] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice (Statistics for Engineering and Information Science)*. Springer Verlag, New York, 2001.
[13] P.F. Felzenszwalb. Representation and detection of deformable shapes. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 27(2), 2005.
[14] P.F. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *Int. Jour. of Comp. Vision*, 61(1), 2005.
[15] M.A. Fischler and R.A. Eschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 22(1):67–92, 1973.
[16] L.R. Ford. Network flow theory. Paper P-923, The Rand Corporation, Santa Monica, 1956.
[17] W. Förstner and E. Gülch. A fast operator for detection and precise localization of distinct points, corners and circular features. In *Intercommission Conference on Fast Processing of Photogrammetric Data*, pages 281–305, Interlaken, Switzerland, 1987.
[18] Y. Gdalyahu and D. Weinshall. Flexible syntactic matching of curves and its application to automatic hierarchical classication of silhouettes. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 21(12):1312–1328, 1999.
[19] U. Grenander, Y. Chow, and D.M. Keenan. *Hands: A Pattern Theoretic Study of Biological Shapes*. Springer Verlag, New York, 1991.
[20] L. Gui, J. Thiran, and N. Paragios. Joint object segmentation and behaviour classification in image sequences. In *IEEE Int. Conf. on Comp. Vision and Patt. Recog.*, Minneapolis, Minnesota, 2007.
[21] G.D. Hager and P.N. Belhumeur. Real-time tracking of image regions with changes in geometry and illumination. In *IEEE Int. Conf. on Comp. Vision and Patt. Recog.*, pages 403–410, San Francisco, California, 1996.
[22] C. Harris and M. Stephens. A combined corner and edge detector. In *The Fourth Alvey Vision Conference*, pages 147–151, Manchester, 1988.
[23] A. Jalba, M. Wilkinson, and J. Roerdink. CPM: A deformable model for shape recovery and segmentation based on charged particles. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 26(10):1320–1335, 2004.
[24] I.H. Jermyn and H. Ishikawa. Globally optimal regions and boundaries as minimum ratio weight cycles. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 23(10):1075–1088, 2001.
[25] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int. Jour. of Comp. Vision*, 1(4):321–331, 1988.
[26] L.J. Latecki and R. Lakämper. Shape similarity measure based on correspondence of visual parts. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 22(10):1185–1190, 2000.
[27] E.L. Lawler. Optimal cycles in doubly weighted linear graphs. In *Theory of Graphs: International Symposium*, pages 209–213, New York, USA, 1966. Gordon and Breach.
[28] V. Lempitsky, A. Blake, and C. Rother. Image segmentation by branch-and-mincut. In *Europ. Conf. on Comp. Vision*, Marseille, France, October 2008.
[29] M. Leventon, W. Grimson, and O. Faugeras. Statistical shape influence in geodesic active contours. In *IEEE Int. Conf. on Comp. Vision and Patt. Recog.*, volume 1, pages 316–323, Hilton Head Island, South Carolina, 2000.
[30] D. Lowe. Object recognition from local scale-invariant features. In *IEEE Int. Conf. on Comp. Vision*, Corfu, Greece, September 1999.
[31] M. Maes. Polygonal shape recognition using string-matching techniques. *Pattern Recognition*, 24(5):433–440, 1991.
[32] R. McConnell, R. Kwok, J.C. Curlander, W. Kober, and S.S. Pang. $\psi - s$ correlation and dynamic time warping: two methods for tracking ice floes in SAR images. *IEEE Transactions on Geosciences and Remote Sensing*, 29:1004–1012, 1991.
[33] E.F. Moore. The shortest path through a maze. In *International Symposium on the Theory of Switching*, pages 285–292. Harvard University Press, 1959.
[34] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Comm. Pure Appl. Math.*, 42:577–685, 1989.
[35] D. Ramanan and C. Sminchisescu. Training deformable models for localization. In *IEEE Int. Conf. on Comp. Vision and Patt. Recog.*, pages 206–213, New York, June 2006.
[36] M. Rousson and D. Cremers. Efficient kernel density estimation of shape and intensity priors for level set segmentation. In *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, volume 1, pages 757–764, 2005.
[37] M. Rousson and N. Paragios. Shape priors for level set representations. In *Europ. Conf. on Comp. Vision*, pages 78–92, Copenhagen, Denmark, 2002. Springer Verlag.
[38] T. Schoenemann and D. Cremers. Globally optimal image segmentation with an elastic shape prior. In *IEEE Int. Conf. on Comp. Vision*, Rio de Janeiro, Brazil, October 2007.
[39] T. Schoenemann and D. Cremers. Globally optimal shape-based tracking in real-time. In *IEEE Int. Conf. on Comp. Vision and Patt. Recog.*, Anchorage, Alaska, June 2008.
[40] T. Schoenemann and D. Cremers. Matching non-rigidly deformable shapes across images: A globally optimal solution. In *IEEE Int. Conf. on Comp. Vision and Patt. Recog.*, Anchorage, Alaska, June 2008.
[41] T. Schoenemann, F.R. Schmidt, and D. Cremers. Image segmentation with elastic shape priors via global geodesics in product spaces. In *British Machine Vision Conference (BMVC)*, Leeds, UK, September 2008.
[42] J. Shi and C. Tomasi. Good features to track. In *IEEE Int. Conf. on Comp. Vision and Patt. Recog.*, Seattle, Washington, June 1994.
[43] A. Tsai, A. Yezzi, W. Wells, C. Tempany, D. Tucker, A. Fan, E. Grimson, and A. Willsky. Model–based curve evolution technique for image segmentation. In *IEEE Int. Conf. on Comp. Vision and Patt. Recog.*, pages 463–468, Kauai, Hawaii, 2001.
[44] X. Xie and M. Mirmehdi. MAC: Magnetostatic active contour model. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 30(4):632 – 646, 2008.
[45] C. Xu and J. Prince. Generalized gradient vector flow external forces for active contours. *Signal Processing*, 71(2):131–139, 1998.

Thomas Schoenemann received a BS (Vordiplom, 2002) and the MS (Diplom, 2005) in Computer Science from the Technical University RWTH Aachen, Germany. Since January 2006 he is a Ph.D. student at the Computer Vision Group at the University of Bonn, Germany. His research is focused on combinatorial optimization and applications to Computer Vision.

Daniel Cremers received the BS in Mathematics (1994) and Physics (1994), and an MS (Diplom) in Theoretical Physics (1997) from the University of Heidelberg. In 2002 he obtained a PhD in Computer Science from the University of Mannheim, Germany. Subsequently he spent two years as a postdoctoral researcher at the University of California at Los Angeles and one year as a permanent researcher at Siemens Corporate Research (Princeton). Since October 2005 he heads the Research Group for Computer Vision, Image Processing and Pattern Recognition at the University of Bonn, Germany. His research is focused on statistical and variational methods in computer vision. He received several awards, in particular the *Best Paper of the Year 2003* by the Pattern Recognition Society, the *Olympus Award 2004*, and the *UCLA Chancellor's Award for Postdoctoral Research 2005*.