Silhouette Area Based Similarity Measure for Template Matching in Constant Time

Daniel Mohr and Gabriel Zachmann

Clausthal University
{dmoh,zach}@tu-clausthal.de

Abstract. In this paper, we present a novel, fast, resolution-independent silhouette area-based matching approach. We approximate the silhouette area by a small set of axis-aligned rectangles. This yields a very memory efficient representation of templates. In addition, utilizing the integral image, we can thus compare a silhouette with an input image at an arbitrary position in constant time.
Furthermore, we present a new method to build a template hierarchy optimized for our rectangular representation of template silhouettes. With the template hierarchy, the complexity of our matching method for n templates is $O(\log n)$. For example, we can match a hierarchy consisting of 1000 templates in $1.5ms$. Overall, our contribution constitutes an important piece in the initialization stage of any tracker of (articulated) objects.

Key words: pose estimation, tracking, template matching, rectangle packing problem

1 Introduction

Most template-based object tracking systems compare a segmented input image with a set of templates at numerous positions in the input image, especially at initialization. The main focus of this paper is to present a novel, very fast algorithm for this stage of a complete tracking system. In a complete tracking system, this initial match would then be used by the next stage to estimate position, orientation and pose.

Usually, when a model of an articulated object is available, there is a large number of templates that must be compared with the input image. Since the template matching stage does very little besides the comparisons, it is crucial that each comparison can be performed extremely fast.

In this paper, we propose a novel method for very fast approximate area silhouette comparison between model templates and the segmented input image. For one template comparison, Stenger el al. [1] achieved a computation time proportional to the contour length of the template silhouette. We propose a new method, which reduces the computation time to be *constant* in the contour length and image resolution. To achieve this, we first approximate all template silhouettes by axis-aligned rectangles, which is done in a preprocessing step. In

the online phase, we compute the integral image [2, 3] of the segmented image. With this, the joint probability of a rectangle to match to an image region can be computed by four lookups in the integral image. Moreover, we present an algorithm to build a template hierarchy that can compare a large set of templates in sublinear time. The *main contributions* are:
An algorithm that approximates arbitrary shapes by a minimal set of axis-aligned rectangles. This results in a resolution-independent, very memory efficient silhouette area representation.
An algorithm to compare an object silhouette in $O(1)$. In contrast the algorithm proposed by [1] needs $O(\text{contour length})$.
We propose an algorithm to cluster templates hierarchically guided by their mutually overlapping areas. Our method builds on the recently developed batch neural gas clustering algorithm, which yields better results than more classical algorithms. This hierarchy further reduces the matching complexity for n templates from $O(n)$ to $O(\log n)$.

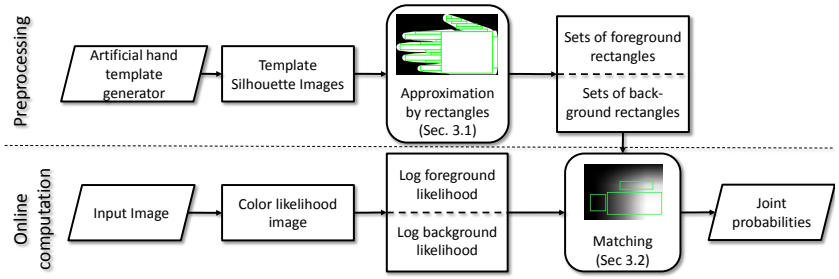

Fig. 1. Overview of our approach using rectangle sets to approximate a silhouette. This speeds up the matching by a factor 5–30 compared to the approach proposed by Stenger et al. [1].

Our approach only requires that binary silhouettes of the model in an arbitrary pose can be generated and that the input image can be segmented. The segmentation result does not necessarily need to be binarized. The approach can handle scalar segmentations as well.

It should be obvious that our proposed methods are suitable for any kind of template based matching of silhouettes. For sake of clarity, though, we will describe our novel methods in the following by the example of the human hand, since human hand tracking is our long-term target application. This includes the full 26 DOFs of the hand, not only a few poses or only the 2D position. To achieve this challenging task, we mainly use two different features for matching: edge gradients and skin color. In this paper, we focus on the skin color feature. We use a skin segmentation algorithm that computes for each image pixel the probability to represent skin or background, resp. We generate our templates by

an artificial 3D hand model. This model can be rendered in any desired state, and it can be easily projected onto 2D and binarized to get the hand silhouette. Given an input image, the goal then is to find the best matching hand silhouette.

We use the joint probability as proposed by Stenger et al. [1] to compare the silhouettes with the segmentation result. A simple area overlap, of course, could be used, too. The only difference is that the sum instead of the product of probabilities would have be computed. For details, see Sec. 3.

2 Related Work

A lot of object tracking approaches based on silhouette comparison have been proposed. The approaches can be divided into two classes. The first class needs a binary silhouette of both, the model and the query image. The second class compare binary model silhouette area with the likelihood map of the query image.

A simple method belonging to the first class is used in [4, 5]. The difference between the model silhouette and segmented foreground area in the query image is computed. The exponential of the negative squared difference is used as silhouette matching probability. A slightly different measure is used by Kato et al. [6]. First, they define the model silhouette area A_M, the segmented area A_I and the intersecting area $A_O = A_I \cap A_M$. The differences $A_I - A_O$, $A_M - A_O$ and $A_I - A_M$ are integrated in the same way, as described above, into the overall measure. In [7], the non-overlapping area of the model and the segmented silhouettes are integrated into classical optimization methods, e.g. Levenberg-Marquardt or downhill simplex. Nirei et al. [8] first compute the distance transform of both the input and model silhouette. Regarding the distance transformed images as vectors, they compute the normalized scalar product of these vectors. Additionally, the model is divided into meaningful parts. Next, for each part, the area overlap between the part and the segmented input image is computed. Then, a weighted sum of the quotient between this overlap and the area of the corresponding model part is computed. The final similarity is the sum of the scalar product and the weighted sum. In [9, 10] a compact description of the hand model is generated. Vectors from the gravity center to sample points on the silhouette boundary, normalized by the square root of the silhouette area, are used as hand representation. During tracking, the same transformations are performed to the binary input image and the vector is compared to the database. A completely different approach is proposed by Zhou and Huang [11]. Although they extract the silhouette from the input image, they use only local features extracted from the silhouette boundary. Their features are inspired by the SIFT descriptor [12]. Each silhouette is described by a set of feature points. The chamfer distance between the feature points is used as similarity measure.

All the aforementioned approaches have the same drawback: to ensure that the algorithms work, a binary segmentation of the input image of high quality is necessary. The thresholds, needed for the binarization, are often not easy to determine.

To our knowledge, there are much less approaches working directly on the color likelihood map of a segmentation. In [13] the skin-color likelihood is used. For further matching, new features, called likelihood edges, are generated by applying an edge operator to the likelihood ratio image. But, in many cases, this leads to a very noisy edge image. In [1, 14, 15], the skin-color likelihood map is directly compared with hand silhouettes. The product of all skin probabilities at the silhouette foreground is multiplied with the product of all background probabilities in the template background. Stenger et al. [14] proposed a method for the efficient computation of this joint probability. The row-wise prefix sum in the log-likelihood image is computed. The original product along all pixels in a row reduces to three lookups in the prefix sum. Thus, the complexity to compute the joint probability is linear in the number of pixels along the template border.

Nevertheless, the above mentioned approach has some disadvantages. First of all, the template representation is resolution dependent. Typically, the distance of the object from the camera is not constant, and thus different sizes of the templates need to be considered. Consequently, for each scale, an extra set of the templates has to be kept in memory. Also, the higher the resolution of the images, the higher is the matching cost.

Our approach does not have all these disadvantages.

3 Silhouette Representation

The key issue of our fast matching approach is the representation of the template silhouettes. Figure 1 shows an overview of our approach.

To avoid the issues mentioned in the previous section, we propose a novel *resolution-independent representation* of template silhouettes. With such a representation, one can perform silhouette matching at arbitrary resolutions in constant time with respect to the template size. We propose to approximate a silhouette by a set of axis-aligned mutually disjoint rectangles. In the remainder of this paper, we denote the integral image of a gray scale image I by II:

$$\mathit{II}(x, y) = \sum_{\substack{0 \le i \le x \\ 0 \le j \le y}} I(i, j) \tag{1}$$

Let R be an axis-aligned rectangle with upper left corner $\mathbf{u}$ and lower right corner $\mathbf{v}$, both inside I. The sum of the area R of all pixels in I is given by

$$\sum_R I(i, j) = \mathit{II}(\mathrm{v}_x, \mathrm{v}_y) + \mathit{II}(\mathrm{u}_x - 1, \mathrm{u}_y - 1) - \mathit{II}(\mathrm{v}_x, \mathrm{u}_y - 1) - I(\mathrm{u}_x - 1, \mathrm{v}_y) \tag{2}$$

Let T_S with $T_S(x, y) \in \{0, 1\}$ be a binary image representing a template T. Let S and $\bar{S}$ denote the set of foreground and background pixels in T_S, resp. We compute a set of n mutually non-overlapping rectangles $\mathcal{R} = \{R_i\}_{i=1 \cdots n}$ that cover S. The number of rectangles n depends on the silhouette shape and thus varies slightly from silhouette to silhouette. Figure 2 shows some example silhouettes with their approximating rectangles.

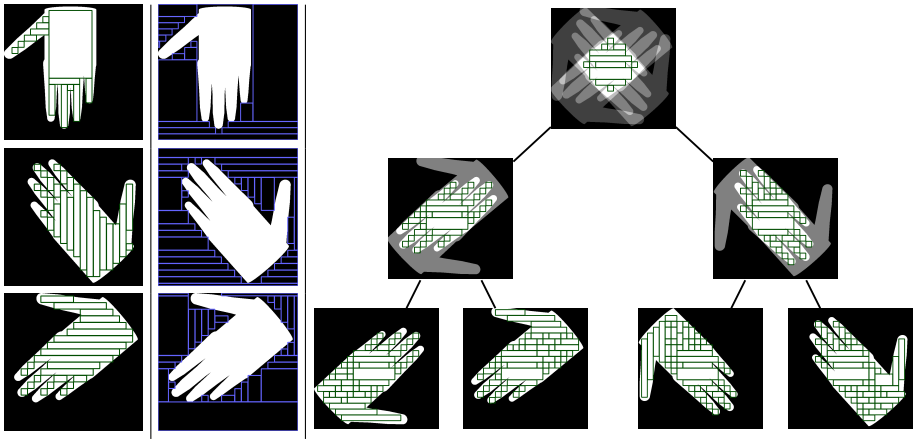

Fig. 2. Example silhouettes approximated by a set of rectangles (at 32×32 squares). The *left* column shows rectangles approximating the foreground, the *middle* one the rectangles approximating the background. The *right* one shows the template hierarchy generated by our approach in Sec. 3.3. For the sake of clarity, only the rectangles approximating the foreground are shown.

3.1 Rectangle Covering Computation

In the following, we denote a set of rectangles approximating S with $\mathcal{R}_S$. To obtain a good approximation, one has to minimize the non-overlapping area A of S and $\mathcal{R}_S$,

$$A = \min_{\mathcal{R}_S} \left| (S \cup \bigcup_{R_i \in \mathcal{R}_S} R_i) \setminus (S \cap \bigcup_{R_i \in \mathcal{R}_S} R_i) \right| \tag{3}$$

Obviously, there is a trade-off between A and $\mathcal{R}_S$. The smaller the number of rectangles, the faster the matching is, but also the more inaccurate.

A lot of work solving similar problems exists. One has to differentiate between rectangle covering [16–18] and partitioning [19, 20] problems. Covering allows an arbitrary overlap between the rectangles in $\mathcal{R}_S$, partitioning does not. Most covering and partitioning algorithms compute solutions under the constraint that the rectangles lie completely inside the polygon to be covered. Our problem is similar to standard partitioning in that we do not allow overlaps between the rectangles $\mathcal{R}_S$, but it differs from partitioning because we do not need rectangles to lie completely in the silhouette S. In fact, we even encourage a solution where some rectangles lie slightly outside. The reason is that S never perfectly matches the observed real hand. Therefore, we can allow $A > 0$, which usually leads to solutions with much smaller numbers of rectangles $\mathcal{R}_S$. In the following, we present a simple algorithm to obtain a solution with $A < \delta$, where δ is application-dependent.

First, the model (here, the human hand) is rendered at a given state and rasterized at a high resolution. We obtain the resulting template T and, after

thresholding, the binary image T_S. For simplification, we normalize the image dimensions to be in $[0, 1]$. Next, we subdivide the image into $r \times s$ uniform boxes. The rectangles to cover S are oriented at the raster defined by these boxes. Basically, we compute the covering of an $r \times s$ image, which we denote by S_{rs}.

In the first step of our dynamic programming approach, we perform the following initialization: we define a benefit value $g_i = g(R_i)$ for each feasible rectangle R_i in S_{rs}, which indicates the benefit of a rectangle when included in the final set of covering rectangles $\mathcal{R}_S$. This value is computed as:

$$g(R_i) = -\theta + \sum_{(x,y) \in R_i} (T_S(x, y) - \tau) \tag{4}$$

The parameter $\tau \in [0, 1]$ controls the penalty for covering a background box by a rectangle and the gain for covering a foreground box. For a value close to 0, the algorithm covers more background boxes in order to cover more foreground boxes as well. If τ is close to 1, the rectangles tend to cover no background rectangles and, thus, are nearly completely inside the silhouette. For now, we assume that $\tau = 0.5$. In Sections 3.2 and 3.3 we will need other values for τ.

The parameter θ adds a penalty to each rectangle R_i in the covering set $\mathcal{R}$. The parameter controls the aforementioned trade-off between the covering error A and the number of rectangles in $\mathcal{R}$. Because θ is a local control parameter, we cannot directly control the global error A. Instead, we set θ to an initial value, compute the covering, evaluate the error A and, if it is to high, we decrease θ and run the algorithm again.

We compute the optimal covering as follows. Let $\mathcal{R}^*$ denote the optimal covering for silhouette S. Let $R^{\mathbf{u}}_{\mathbf{v}} = R^{\mathrm{u}_x, \mathrm{u}_y}_{\mathrm{v}_x, \mathrm{v}_y}$ denote a rectangle with upper left corner $\mathbf{u}$ and lower right corner $\mathbf{v}$. Assume $R^{\mathbf{u}}_{\mathbf{v}}$ or a subset is part of the optimal covering, and let $\mathcal{D}(R^{\mathbf{u}}_{\mathbf{v}})$ denote the "benefit" value of this sub-covering. Then either $R^{\mathbf{u}}_{\mathbf{v}} \in \mathcal{R}^*$ or $R^{\mathbf{u}}_{\mathbf{v}}$ contains a number of non-overlapping rectangles that are in $\mathcal{R}^*$. Thus, the covering problem exhibits the *optimal substructure property* and dynamic programming can be applied. Therefore, we can compute

$$\begin{aligned} \mathcal{D}(R^{\mathbf{u}}_{\mathbf{v}}) = \max\Big\{ 0,\, g(R^{\mathbf{u}}_{\mathbf{v}}),\, \max_{\mathrm{u}_x < x < \mathrm{v}_x} \big\{ \mathcal{D}(R^{\mathrm{u}_x, \mathrm{u}_y}_{x, \mathrm{v}_y}) + \mathcal{D}(R^{x, \mathrm{u}_y}_{\mathrm{v}_x, \mathrm{v}_y}) \big\}, \\ \max_{\mathrm{v}_x < y < \mathrm{v}_y} \big\{ \mathcal{D}(R^{\mathrm{u}_x, \mathrm{u}_y}_{\mathrm{v}_x, y}) + \mathcal{D}(R^{\mathrm{u}_x, y}_{\mathrm{v}_x, \mathrm{v}_y}) \big\} \Big\} \end{aligned} \tag{5}$$

Obviously, the optimal solution is obtained through $\mathcal{D}(R^{0,0}_{r,s})$ and the base case is $\mathcal{D}(R^{x,y}_{x+1,y+1}) = g(R^{x,y}_{x+1,y+1})$.

In our implementation, we try a number of different solutions $r \times s = 2 \times 2, \cdots, 32 \times 32$. As soon as the covering accuracy criterion is fulfilled, we terminate the computation.

3.2 Matching Silhouettes

In the previous section, we have developed an algorithm to compute for each template silhouette a resolution-independent compact representation consisting

of axis-aligned rectangles. In the following, this representation will be used for fast silhouette area based template matching.

Our goal is to compare a silhouette S with an input image I at a given position $\mathbf{p}$ using the joint probability (see Stenger et al. [14]). The first step is the foreground/background segmentation. Due to its higher robustness compared to binary segmentation, we want to use the color likelihood. In the following, the color likelihood image of an input image I is denoted with $\tilde{L}$ with $\tilde{L}(x, y) \in [0, 1]$. To convert the product in the joint probability into sums, we take the pixel-wise logarithm: $L(x, y) = \log \tilde{L}(x, y)$.

Utilizing Eq. 2, we can compute the joint probability at position $\mathbf{p}$ by:

$$P_S(\mathbf{p}) = \sum_{R_i \in \mathcal{R}_S} (IL(\begin{pmatrix} \mathrm{v}_x^i \\ \mathrm{v}_y^i \end{pmatrix} + \mathbf{p}) + IL(\begin{pmatrix} \mathrm{u}_x^i \\ \mathrm{u}_y^i \end{pmatrix} + \mathbf{p}) - IL(\begin{pmatrix} \mathrm{v}_x^i \\ \mathrm{u}_y^i \end{pmatrix} + \mathbf{p}) - IL(\begin{pmatrix} \mathrm{u}_x^i \\ \mathrm{v}_y^i \end{pmatrix} + \mathbf{p})) \quad (6)$$

The rectangle set $\mathcal{R}_S$ approximates only the silhouette foreground. To get the appropriate match probability for a template, one has to take into account the background distribution, too.

Fortunately, the set of background pixels $\bar{S}$ of a silhouette image, obviously, can be approximated by a set of rectangles with the same algorithm described in the last section. Having computed $\mathcal{R}_{\bar{S}}$, we can compute $P_{\bar{S}}$.

P_S and $P_{\bar{S}}$ are resolution-dependent and need to be normalized. In the following, we explain the normalization for P_S. $P_{\bar{S}}$ can be normalized analogously. A naive approach is to normalize P_S. However, this fails in cases where the template is partially outside the input image. Therefore, we propose a "smart" normalization as follows.

For each pixel not covered by any rectangle, including *all* pixels of the template image that are outside the image borders, we assume a likelihood value of 0.5. The value is motivated by the assumption that at a pixel not yet observed, the probability to be foreground or background is equal. Lets denote the number of pixels of rectangle R_i inside the input image at position p in an input image by $N_{R_i}^{\mathbf{p}}$. Then we normalize P_S as follows:

$$P_S^N(\mathbf{p}) = \frac{1}{|S|}(P_S(\mathbf{p}) \cdot \log(0.5)(|S| - N_R^{\mathbf{p}})) \ , \qquad \text{with} \ \ N_R^{\mathbf{p}} = \sum_{R_i \in \mathcal{R}_S} N_{R_i}^{\mathbf{p}} \quad (7)$$

To ensure, that $|S| - N_R^{\mathbf{p}}$ is positive, we set the parameter τ from Eq. 4 to 0.95. The final joint probability is

$$P^N = \exp(\frac{1}{2}(P_S^N + P_{\bar{S}}^N)) \quad (8)$$

where $P_{\bar{S}}^N$ is the normalized background joint probability. Treating the joint probabilities for the foreground and background equally takes care of the fact that different silhouette shapes have different area relative to their bounding box used in the template: in a silhouette with fewer foreground pixels, the matching of the background pixels should not have a bigger weight then the foreground pixels and vice versa.

Using the same template at different sizes, i.e. when the distance from an object to the camera changes, is straight forward: simply scale the rectangles accordingly. No additional representation has to be stored. Comparability between the same template at different sizes is ensured by the normalization.

3.3 The Template Hierarchy

In the previous section, we have described a novel method to match an arbitrary template T to an input image I. In a typical tracking application, especially when dealing with articulated objects, a huge number of templates must be matched. A suitable approach to reduce the complexity from $O(\#templates)$ to $O(\log \#templates)$ is to use a template hierarchy. However, building a well working one is still a challenging task.

We propose an approach to build a hierarchy that naturally fits with our representation of the silhouettes by rectangles. In addition, it even further reduces the computational effort per template matching. We build the tree structure by utilizing a hierarchical clustering algorithm. Vectors, describing the similarity between templates, are computed and used as input for the clustering algorithm [21]. The output are k disjoint clusters, where k defines the number of children per tree node. At each node in the template tree, rectangles covering the intersection of all template silhouette of all children are pre-stored.

For matching n templates, we traverse the hierarchy. The rectangles from the root to one leaf constitutes a covering of that template, which is thus being matched incrementally during traversal. At the same time, we prune large parts of the hierarchy (i.e. large numbers of templates), because we descend only into those subtrees with largest probability. Figure 2 illustrates the basic idea of our template hierarchy. Due to space limitations, we can not provide a detailed description of the template tree generation and traversal.

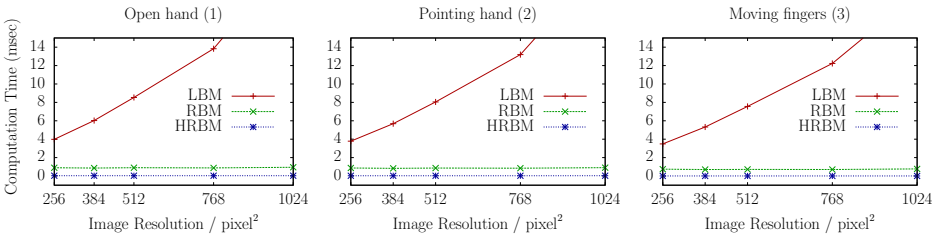

Fig. 3. Each plot shows the average computation time for all three approaches: LBM [1], RBM (our approach), HRBM (our approach incl. hierarchy). Clearly, our approaches are significantly faster and, even more important, resolution independent.

4 Results

For all our experiments, we have chosen to set the silhouette image discretization to $r = s = 32$ boxes. The parameters in Eq. 4 were set initially to $\tau = 0.95$ and $\theta = (1-\tau) * 10^{-4}$. In order to achieve a small enough global error $A < \delta$, θ was halved successively. In our experience, 5 iterations were sufficient.

4.1 Rectangle Approximation

First, we evaluated the quality of our approach approximating silhouettes by axis-aligned rectangles. The two important criteria are the area of the covered silhouette and the number of rectangles needed. Let us denote the benefit value for the perfect covering (i.e. all foreground and no background pixels are covered) by $\mathcal{D}_P$ and its solution by $\mathcal{R}_P$. For an accuracy measure we use:

$$Q = \frac{\mathcal{D}(R_{r,s}^{0,0}) + \theta|\mathcal{R}^*|}{\mathcal{D}_P + \theta|\mathcal{R}_P|} \tag{9}$$

In our experiments, where we have tried to cover a representative set of postures and orientations, we have observed that on average, we need about 20 rectangles to obtain a covering accuracy of $Q = 0.7$. In practice, we have observed that this value is appropriate for the similarity measure. Covering only a part of the silhouette can even increase the matching quality because we obtain a higher tolerance to slightly varying shapes in the input image.

4.2 Matching Quality

We compare our approach with a state-of-the-art approach proposed by Stenger et al. [1], because our approach was inspired by theirs and the application (hand tracking) is the same.

In the following, we will denote the algorithm from [1] as *line-based matching (LBM)*, ours as *rectangle-based matching (RBM)*, and ours including the hierarchy with *hierarchical matching (HRBM)*. It is not quite fair to compare a hierarchical approach to non-hierarchical ones. The reason for this is that, during the traversal, the decision which child nodes are visited is based only on the information of the children itself, not on the whole subtree. Thus, there is no guarantee that the subtree containing the best matching template is visited at all. Nevertheless, mostly HRBM provides a result very similar to the best matching template and, therefore, we add the results of the hierarchical match to our plots to analyze the potential of the hierarchy.

In the following, we will evaluate the difference between the methods with regard to resolution-independence, computation time, and accuracy. We generated templates with an artificial 3D hand model. We used the templates also as input images. There are two reasons to use such synthetic input datasets. First, we have the ground truth and second, we can eliminate negative influences like

differences between hand model and real hand, image noise, bad illumination, and so on.

We generated three datasets for evaluation. Dataset 1, consisting of 1536 templates, is an open hand at different rotation angles. Dataset 2 is a pointing hand rendered at the same rotation angles as dataset 1. In dataset 3, consisting of 1080 templates, we used an open hand with moving fingers. Additionally, for each position of the fingers, we rendered the model at different rotations.

First, we examined the dependence between the resolution and computation time. We used input images at 5 different resolutions. We averaged the time to compute the joint probability for all frames at 49 positions. The result is shown in Figure 3. Clearly, LBM's computation time depends linearly on the resolution, while our approaches exhibit constant time.

Second, we compared the matching quality of the three approaches. We expect LBM to work best on the artificial datasets because, for each template, there is an exactly matching input image. Supposing the LBM templates are available at the same resolution than the hand found in the input image, there is a pixelwise identically template for each input image. For evaluation we used an input image resolution of 256×256 and compared the template at 5 different scalings (from 70×70 up to 200×200). The scalings are chosen such that one of the five scales matches to the hand in the input image with an accuracy of ± 1 pixel. All three approaches always found the correct location of the hand in the input image. Thus, for evaluation, we recorded at this position the 10 best matching templates (rank 0–9). Please see Fig 4 for the results. In the open-hand and pointing-hand datasets, LBM and RBM work nearly equally well. Apparently, all approaches have some difficulties to find the correct template in the moving-fingers dataset. The reason for that is that, in this set, there are many templates with nearly identical silhouette: they differ only by one finger flexed by a few degrees. Due to the difference in scale by one pixel, even the LBM can fail to find the best matching template.

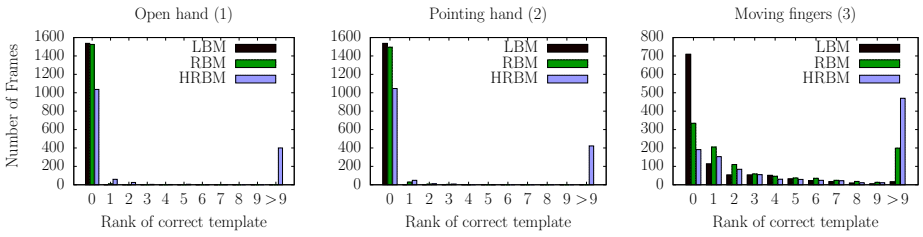

Fig. 4. The histograms show the matching accuracy for all three approaches: LBM [1], RBM (our approach) and HRBM (our approach incl. hierarchy). Rank k means that the correct template is found to be the $k-$th best match. Lower ranks are better.

5 Conclusions

In this paper, we have developed a silhouette area based *similarity measure* for template matching with constant time complexity. We get a significant increase in template matching speed and reduction of storage space by accepting a slight decrease of matching accuracy. We have also proposed a novel method to compute such a rectangle covering based on dynamic programming. Additionally, we have presented a template hierarchy, which exploits our representation of the silhouettes. This hierarchy reduces the computational complexity for a set of templates from linear to logarithmic time. Please remember that our contributions constitute just one of the many pieces of a complete hand tracking system.

Overall, we need about 0.7 μs on average to compare one template silhouette to one position in an input image at an arbitrary resolution. This is about a factor 25 faster than the state-of-the-art approach from [1] at a resolution of 1024×1024. Furthermore, the template representation is very memory efficient. For example, for 1500 templates, the complete hierarchy consumes less then 1 MByte storage space.

In the future, we plan to implement our approach in a massively parallel programming paradigm. Furthermore, we will extend our hierarchical approach to a random forest approach, which we expect to improve the template matching quality significantly. To get different classifiers at each node, one can choose a random subset instead of all covering templates to cluster a tree node for further subdivision. We also plan to build a hierarchy for our templates based on edge features and combine it with the one proposed in this paper.

References

1. Stenger, B., Thayananthan, A., Torr, P.H.S., Cipolla, R.: Model-based hand tracking using a hierarchical bayesian filter. In: IEEE Transactions on Pattern Analysis and Machine Intelligence. (2006)
2. Crow, F.C.: Summed-area tables for texture mapping. In: SIGGRAPH: Proceedings of the 11th annual conference on Computer graphics and interactive techniques. (1984)
3. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: IEEE Conference on Computer Vision and Pattern Recognition. (2001)
4. Lin, J.Y., Wu, Y., Huang, T.S.: 3d model-based hand tracking using stochastic direct search method. In: International Conference on Automatic Face and Gesture Recognition. (2004)
5. Wu, Y., Lin, J.Y., Huang, T.S.: Capturing natural hand articulation. In: International Conference on Computer Vision. (2001)
6. Kato, M., Chen, Y.W., Xu, G.: Articulated hand tracking by pca-ica approach. In: International Conference on Automatic Face and Gesture Recognition. (2006)
7. Ouhaddi, H., Horain, P.: 3d hand gesture tracking by model registration. In: Workshop on Synthetic-Natural Hybrid Coding and Three Dimensional Imaging. (1999)

8. Nirei, K., Saito, H., Mochimaru, M., Ozawa, S.: Human hand tracking from binocular image sequences. In: 22th International Conference on Industrial Electronics, Control, and Instrumentation. (1996)
9. Amai, A., Shimada, N., Shirai, Y.: 3-d hand posture recognition by training contour variation. In: IEEE Conference on Automatic Face and Gesture Recognition. (2004)
10. Shimada, N., Kimura, K., Shirai, Y.: Real-time 3-d hand posture estimation based on 2-d appearance retrieval using monocular camera. In: IEEE International Conference on Computer Vision. (2001)
11. Zhou, H., Huang, T.: Okapi-chamfer matching for articulated object recognition. In: IEEE International Conference on Computer Vision. (2005)
12. Lowe, D.G.: Object recognition from local scale-invariant features. In: IEEE International Conference on Computer Vision. (1999)
13. Zhou, H., Huang, T.: Tracking articulated hand motion with eigen dynamics analysis. In: IEEE International Conference on Computer Vision. (2003)
14. Stenger, B.D.R.: Model-based hand tracking using a hierarchical bayesian filter. In: Dissertation submitted to the University of Cambridge. (2004)
15. Sudderth, E.B., Mandel, M.I., Freeman, W.T., Willsky, A.S.: Visual hand tracking using nonparametric belief propagation. In: IEEE CVPR Workshop on Generative Model Based Vision. (2004)
16. Kumar, V.A., Ramesh, H.: Covering rectilinear polygons with axis-parallel rectangles. In: Annual ACM Symposium on Theory of Computing. (1999)
17. Wu, S., Sahni, S.: Covering rectilinear polygons by rectangles. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. (1990)
18. Heinrich-Litan, L., Lübecke, M.E.: Rectangle covers revisited computationally. In: ACM Journal of Experimental Algorithmics, Vol. 11. (2006)
19. Liou, W., Tan, J.J.M., Lee, R.C.T.: Minimum rectangular partition problem for simple rectilinear polygons. In: IEEE Transactions on Computer-Aided Design. (1990)
20. O'Rourke, J., Tewari, G.: Partitioning orthogonal polygons into fat rectangles in polynomial time. In: In Proc. 13th Canadian Conference on Computational Geometry. (2001)
21. Cottrell, M., Hammer, B., Hasenfuß, A., Villmann, T.: Batch neural gas. In: 5th Workshop On Self-Organizing Maps. (2005)