Polygonal Models for Clothing

Jan Stria, Daniel Průša, and Václav Hlaváč

Center for Machine Perception, Department of Cybernetics,
Faculty of Electrical Engineering, Czech Technical University
Karlovo nám. 13, 121 35 Prague 2, Czech Republic
{striajan,prusapa1,hlavac}@cmp.felk.cvut.cz
http://cmp.felk.cvut.cz

Abstract. We address the problem of recognizing a configuration of a piece of garment fairly spread out on a flat surface. We suppose that the background surface is invariant and that its color is sufficiently dissimilar from the color of a piece of garment. This assumption enables quite reliable segmentation followed by extraction of the garment contour. The contour is approximated by a polygon which is then fitted to a polygonal garment model. The model is specific for each category of garment (e.g. towel, pants, shirt) and its parameters are learned from training data. The fitting procedure is based on minimization of the energy function expressing dissimilarities between observed and expected data. The fitted model provides reliable estimation of garment landmark points which can be utilized for an automated folding using a pair of robotic arms. The proposed method was experimentally verified on a dataset of images. It was also deployed to a robot and tested in a real-time automated folding.

Keywords: clothes folding, robotic manipulation

1 Introduction

We present a solution for identifying an arrangement of a piece of garment spread out on a flat surface. Our research is motivated by the needs of the European Commission funded project Clothes Perception and Manipulation (CloPeMa) [16]. This project focuses on a garments manipulation (sorting, folding, etc.) by a two armed industrial robot which is shown in Fig. 1. The general aim is to advance the state of the art in the autonomous perception and manipulation of limp materials like fabrics, textiles and garments, placing the emphasis on universality and robustness of the methods.

The task of clothes state recognition has already been approached by Miller et al. [9]. They consider a garment fairly spread on a green surface, which allows to segment images using simple color thresholding. The obtained garment contour is fitted to a parametric polygonal model specific for a particular category of clothing. The fitting procedure is based on iterative estimation of numeric parameters of the given model. The authors report quite accurate results. However, the main drawback is a slow performance. It takes 30–150 seconds for a single contour and a single model. This makes the algorithm practically unusable for a

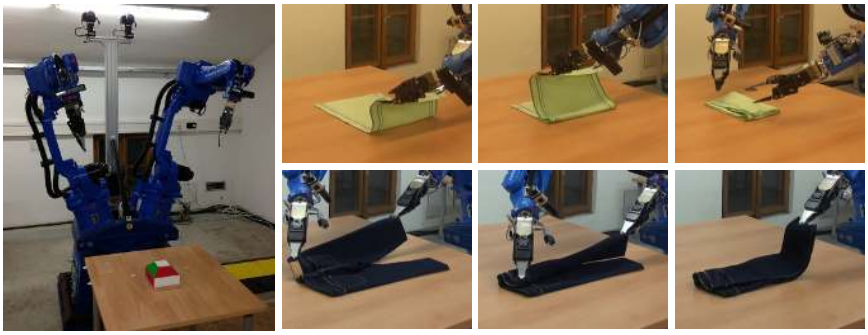


Fig. 1: Our robotic test platform utilizes two industrial hollow-wrist welding manipulators Motoman MA1400 with mounted cameras, Kinect-like rangefinders and dedicated grippers. We use it as the experimental platform for folding of various types of clothes.

real-time operations. The authors also use the parametric model for recognition and fitting of pairs of socks [15]. Information about texture of socks was utilized here as well. Another successful application is an automated folding of towels based on a robust visual detection of their corner points [8]. The two-armed robot starts with a towel randomly dropped on a table and folds it in a sequence of manipulations performed both on the table and in the air.

Kita et al. use single-view image [5] and stereo image [6] to estimate state of the hanging clothes being held by a gripper. Their approach is based on matching a deformable model to the observed data. Hata et al. [3] solve the problem of lifting a single towel from a pile of highly wrinkled towels and grasping it for its corner. The solution is based on detection of the highest point of the pile followed by corner detection in stereo data. Ramisa et al. [12] are also interested in determination of the grasping point. They combine features computed from both color and range images in order to locate highly wrinkled regions. The research area of cloth modeling is explored mainly by the computer graphics community. Hu et al. [4] give an overview of the known methods.

In this work, we propose a complete pipeline for clothes configuration recognition by estimating positions of the most important landmark points (e.g. all four corners of a towel). The identified landmarks can be used for automated folding performed by robotic arms. We introduce our own polygonal models describing contours of various categories of clothing and develop a fast, dynamic programming based methods for an efficient fitting of an unknown contour to the models. Moreover, we have modified the grabcut image segmentation algorithm to work automatically without being initialized by a user input, utilizing a background model learned in advance from training data. The recognition pipeline can be summarized as follows:

1. *Capturing input:* The input is a single color image of a piece of garment spread on a table. We assume that type of the clothing (e.g. towel, pants,

shirt) is known in advance. The image is taken from a bird's eye perspective by a camera attached to the robot. Since the relative position of the table and the camera is known, all pixels not displaying the table and the garment lying on it can be cropped.
2. *Segmentation:* The goal is to segment the garment and its background which is a wooden table in our case. We assume that the table and the garment have statistically dissimilar colors. We also assume that the table is invariant and thus its color properties can be learned from data. These assumptions make it possible to modify the grabcut segmentation algorithm [13] in a way that it does not require manual initialization.
3. *Contour detection:* The binary mask obtained from the segmentation is processed by Moore's algorithm [2] for tracing 8-connected boundary of a region. This gives a bounding polygon of the garment. Vertices of the polygon are formed by individual contour pixels.
4. *Polygonal approximation:* The dense boundary is then approximated by a polygon having fewer vertices. Their exact count depends on a model of garment which we want to fit in the following step. Generally, the number of vertices is higher than the number of landmark points for a specific model.
5. *Model fitting:* The polygonal approximation of the garment contour is matched to a polygonal model defined for the corresponding type of garment. The matching procedure employs dynamic programming approach to find correspondences between approximating vertices and landmark points defining the specific polygonal model. The matching considers mainly local features of the approximating polygon. As there are more vertices than landmarks, some of the contour vertices remain unmatched.

2 Contour extraction

2.1 Learning the background color model

The background color model is a conditional probabilistic distribution of RGB values of background pixels. The distribution is represented as a mixture of K 3D Gaussians (GMM):

$$p(z) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}(z; \mu_k, \Sigma_k) = \sum_{k=1}^{K} \pi_k \, \frac{\exp\left(-\frac{1}{2}(z-\mu_k)^T \Sigma_k^{-1}(z-\mu_k)\right)}{\sqrt{(2\pi)^3 |\Sigma_k|}} \qquad (1)$$

Here π_k is a prior probability of k-th component and $\mathcal{N}(z; \mu_k, \Sigma_k)$ denotes 3D normal distribution having a mean vector μ_k and a covariance matrix Σ_k.

The mixture is learned from training data, i.e. from a set $Z = \{z_n = (z_n^R, z_n^G, z_n^B)^T \in [0,1]^3\}$ of vectors representing RGB intensities of $|Z|$ background pixels. The number of GMM components K is determined empirically based on the number of visible clusters in RGB cube with visualized training data. E.g. for a nearly uniform green background one component should be sufficient, for the table in our experiments we choose three components.

To train the GMM probabilistic distribution, we split the training data to K clusters $C_1 \dots C_K$ at first, employing the binary tree algorithm for the palette design [10]. The algorithm starts with all training data Z assigned to a single cluster and it iteratively constructs a binary tree like hierarchy of clusters in a top-bottom manner. In each iteration, the cluster having the greatest variance is split to two new clusters. The separating plane passes through the center of the cluster and its normal vector is parallel with the principal eigenvector of the cluster. The algorithm stops after $K-1$ iterations with K clusters.

Prior probability π_k, mean vector μ_k and covariance matrix Σ_k for the k-th GMM component is computed using the maximum likelihood principle [1] from data vectors contained in the corresponding cluster C_k:

$$\pi_k = \frac{|C_k|}{|Z|}, \quad \mu_k = \frac{1}{|C_k|} \sum_{z_n \in C_k} z_n, \quad \Sigma_k = \frac{1}{|C_k|} \sum_{z_n \in C_k} (z_n - \mu_k)(z_n - \mu_k)^T \quad (2)$$

2.2 Unsupervised segmentation

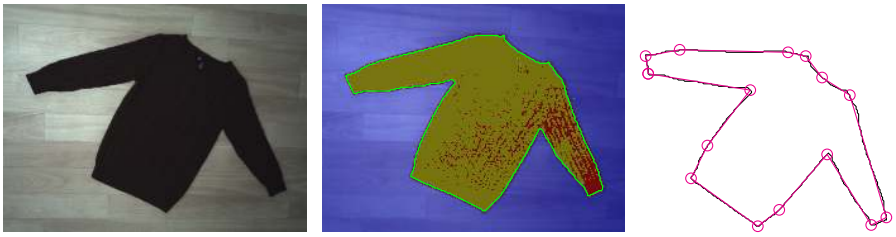
The segmentation is based on the grabcut algorithm [13] which is originally a supervised method. It expects an RGB image $Z = \{z_n \in [0,1]^3 \colon n = 1 \dots W \times H\}$ of size $W \times H$. Moreover, the user is expected to determine a trimap $T = \{t_n \in \{F, B, U\} \colon n = 1 \dots W \times H\}$. The value t_n determines for the n-th pixel whether the user considers it being a part of the foreground ($t_n = F$), background ($t_n = B$) or whether the pixel should be classified automatically ($t_n = U$). The trimap T is usually defined via some interactive tool enabling to draw a stroke over foreground pixels, another stroke over background pixels and leave the other pixels undecided.

In the proposed method, the input trimap is created automatically using the learned background GMM probabilistic model from Eq. 1 and two predetermined probability thresholds P_B and P_F:

$$t_n = \begin{cases} F, & p(z_n) < P_F \\ U, & P_F \leq p(z_n) \leq P_B \\ B, & P_B < p(z_n) \end{cases} \quad (3)$$

The thresholds P_B and P_F are set based on the training data so that 3% training pixels have the probability lower than P_F and 80% training pixels have probability higher than P_B in the learned background model. The foreground component of the trimap is thus initialized by lowly probable pixels while the background component by highly probable pixels.

The core part of the grabcut [13] algorithm is an iterative energy minimization. It repeatedly goes through two phases. First, GMM models for the foreground and the background color are reestimated. And second, the individual pixels are relabeled based on finding the minimum cut in a special graph. To estimate the GMM color models we utilize the binary tree algorithm [10] described in Sec. 2.1 followed by the maximum likelihood estimation introduced in Eq. 2. We use three components both for background and foreground GMM



(a) Input image (b) Segmentation (c) Approximation

Fig. 2: (a) Input is formed by a single RGB image. (b) The input trimap for grabcut algorithm is automatically initialized with foreground (plotted in yellow), background (blue) and unknown (red) pixels. The resulting segmentation gives a garment contour (green). (c) The contour is simplified by approximating it by a polygon (magenta).

which is sufficient in our case of not so varying table and garment. The grabcut algorithm iterates until convergence which usually takes 5–15 cycles. However, the segmentation mask is being changed only slightly in the later cycles. Since we need to get the segmentation as fast as possible, we stop the optimization after three cycles.

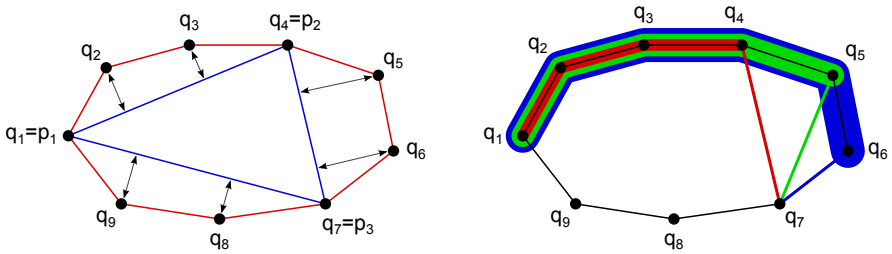
2.3 Contour simplification

The segmentation algorithm proposed in the previous section is followed by Moore's algorithm [2] for contour tracing. The result is a closed contour in the image plane, i.e. a list $(q_1 \dots q_L)$ of 2D coordinates $q_i = (x_i, y_i)$. The number of distinct points L depends on the image resolution as well as on the piece of garment size. Typically, L has an order of hundreds or thousands.

To be able to fit out polygonal model to the contour effectively, we need to simplify the contour by approximating it with a polygon having N vertices where $N \ll L$. More precisely, we want to select a subsequence of N points $(p_1 \dots p_N) \subseteq (q_1 \dots q_L)$. Additionally, we want to minimize the sum of Euclidean distances of the original points $(q_1 \dots q_L)$ to edges of the approximating polygon $(p_1 \dots p_N)$ as seen in Fig. 3a.

The simplification procedure is based on the dynamic programming algorithm for the optimal approximation of an open curve by a polyline [11], [7]. It iteratively constructs the optimal approximation of points $(q_1 \dots q_i)$ by n vertices from previously found approximations of $(q_1 \dots q_j)$ where $j \in \{n-1 \dots i-1\}$ by $n-1$ points. The construction is demonstrated in Fig. 3b.

Time complexity of the algorithm is $O(L^2 N)$. Since the algorithm works with an open curve, it would have to be called L times for every possible cycle breaking point q_i to obtain optimal approximation of the closed curve. However, we only call it constantly many times to get a suboptimal approximation which is sufficient for our purpose.



(a) Contour approximation problem (b) Dynamic programming solution

Fig. 3: (a) The original contour $(q_1 \dots q_L)$ (plotted in red) is simplified by approximating it with a polygon $(p_1 \dots p_N)$ (blue) while minimizing distances of the original points q_i to polygon edges. (b) Dynamic programming algorithm for polygonal approximation utilizes previously constructed approximations of points $(q_1 \dots q_4, q_5, q_6)$ by $n-1$ vertices (plotted in various colors) to obtain approximation of points $(q_1 \dots q_7)$ by n vertices.

3 Polygonal models

3.1 Models definition and learning

To be able to recognize the configuration of a piece of garment, we describe contours of various types of clothing by simple polygonal models. The models are determined by their vertices. Inner angles incident to the vertices are learned from training data. Additional conditions are defined in some cases to deal with inner symmetries or similarities of distinct models. We use the following models:

1. *Towel* is determined by 4 corner vertices as shown in Fig. 4. All inner angles incident to the vertices share the same probability distribution. There is an additional condition that the height of the towel (distance between the top edge and the bottom edge) is required to be longer that its width (distance between the left edge and the right edge).
2. *Pants* are determined by 7 vertices. There are 3 various shared distributions of inner angles as shown in Fig. 4.
3. *Short-sleeved shirt* is determined by 10 vertices and 4 shared distributions of inner angles as shown in Fig. 4. There is an additional condition that the distance between the armpit and the inner corner of the sleeve is required to be maximally 50% of the distance between the armpit and the bottom corner of the shirt.
4. *Long-sleeved shirt* is similar to the short-sleeved model. The distance between the armpit and the inner corner of the sleeve should be minimally 50% of the distance between the armpit and the bottom corner of the shirt.

The probability distributions for inner angles incident to vertices of polygonal models are learned from manually annotated data. We assume that the angles have normal distributions. This seems as a reasonable assumption, since e.g.

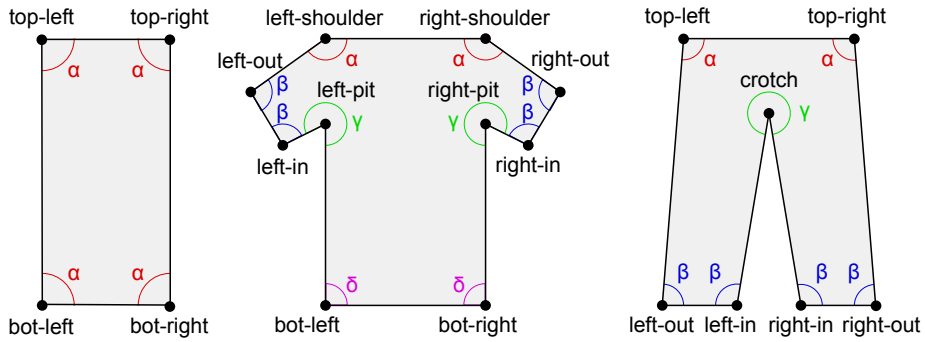
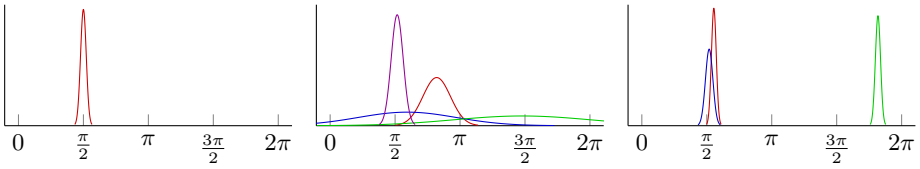


Fig. 4: Polygonal model for towel, short-sleeved shirt and pants. Angles sharing one distribution are denoted by the same letter and plotted with the same color.



(a) Towel (b) Short shirt (c) Pants

Fig. 5: Angle distributions learned for various types of clothes models. Colors of the plotted distributions correspond to angles in Fig. 4.

a corner angle of a towel should be approximately 90° with a certain variance caused by deformations of the contour. The mean and the variance of each normal distribution is estimated using a maximum-likelihood principle similarly to Eq. 2. Various vertices in a model can share the same angles distribution because of obvious symmetries, e.g. all 4 corners of a towel should be statistically identical.

3.2 Problem of model matching

We described in Sec. 2.3 how to approximate a contour by N points $(p_1 \dots p_N)$. Each polygonal model defined in Sec. 3.1 is determined by M vertices $(v_1 \dots v_M)$ where M is specific for the particular model. See examples of models in Fig. 4. We show how to match an unknown simplified contour onto a given model.

We assume that $N \geq M$, i.e. the simplified contour contains more points than is the number of vertices of the model to be matched. The problem of matching can be then defined as a problem of finding a mapping of simplified contour points to model vertices $f\colon \{p_1 \dots p_N\} \to \{v_1 \dots v_M\} \cup \{s\}$. Symbol s represents a dummy vertex which corresponds to a segment of the polygonal model. It makes it possible to leave some of the contour points unmapped to a real vertex. Additionally, a proper mapping f has to satisfy several conditions:

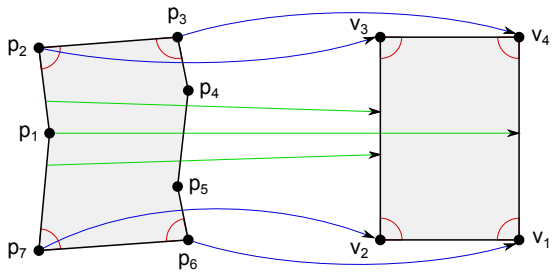


Fig. 6: Points of the simplified contour ($p_1 \dots p_7$) are matched (plotted in blue) to vertices of the polygonal model ($v_1 \dots v_4$). Some of them remain unmatched (green), i.e. they are mapped to a dummy vertex s representing a segment. The energy of the particular matching is based on a similarity of corresponding inner angles (red).

1. There exists a point p_i mapped to it for each vertex v_m. More formally, $\forall v_m \exists p_i \colon f(p_i) = v_m$.
2. No two points p_i and p_j are mapped to the same vertex v_m. However, many points can be mapped to segments represented by a dummy vertex s. Formally, $\forall p_i \neq p_j \colon f(p_i) = f(p_j) \Rightarrow f(p_i) = f(p_j) = s$.
3. The mapping preserves the ordering of points on the polygonal contour and the ordering of vertices of the polygonal model in the clockwise direction. For example of such a proper mapping see Fig. 6.

The number of mappings f satisfying the aforementioned conditions for N contour points and M model vertices is given by the combinatorial formula:

$$N\binom{N-1}{M-1} \geq N\left(\frac{N-1}{M-1}\right)^{M-1} \tag{4}$$

The interpretation is that we can choose 1 of N points to be mapped to the first vertex v_1. From the remaining $N-1$ points, we select a subset of $M-1$ points which are mapped to vertices $v_2 \dots v_M$. All other points are mapped to the dummy vertex s representing all segments of the polygonal model.

We introduce an energy function $E(f)$ associated with a matching f. Let us denote ϕ_i the inner angle adjacent to the point p_i of the simplified contour. Let us also denote μ_m the mean value and σ_m^2 the variance of the normal distribution of inner angles $\mathcal{N}(\phi; \mu_m, \sigma_m^2)$ learned for the vertex v_m of a particular polygonal model. We recall that the same distribution can be shared by several vertices of one polygonal model as seen in Fig. 4. The energy function is then given by:

$$E(f) = -\sum_{f(p_i)=v_m} \log\, \mathcal{N}\left(\phi_i; \mu_m, \sigma_m^2\right) - \sum_{f(p_i)=s} \log\, \mathcal{N}\left(\phi_i; \pi, \frac{\pi^2}{16}\right) \tag{5}$$

It can be seen that we force angles of unmatched points (p_i such that $f(p_i) = s$) to be close to π, i.e. we want the unmatched parts of the contour to resemble

straight segments. We set the variance $\pi^2/16$ for unmatched points empirically. Since the energy is inversely proportional to a probability, the optimal mapping f^* is obtained as $f^* = \arg\min_f E(f)$.

3.3 Matching algorithm

Eq. 4 shows that the count of all admissible mappings is exponential in the number of vertices M. Thus it would be inefficient to evaluate the energy function for each mapping. We have rather developed an algorithm employing a dynamic programming approach which has a polynomial time complexity.

The dynamic programming optimization procedure seen in Alg. 1 is called for every shifted simplified contour $(p'_1 \dots p'_N) = (p_d \dots p_N, p_1 \dots p_{d-1})$, where the shift is $d \in \{1 \dots N\}$. The reason is that Alg. 1 finds a mapping f such that $f(p_{i_m}) = v_m$ for $m \in \{1 \dots M\}$ and $1 \le i_1 \le i_2 \le \dots \le i_M \le N$, i.e. one of the first points is mapped to the vertex v_1, some of its successors along the contour to the vertex v_2 and so on. Thus we have to try various shifts in order to be able to map any point to vertex v_1 as seen in Fig. 6.

Alg. 1 does not work with points and vertices directly. It expects a precomputed matrix $V \in \mathbb{R}^{N \times M}$ and a vector $S \in \mathbb{R}^N$ instead. The value $V_{i,m}$ is a cost of matching the inner angle ϕ_i associated with the point p_i to the learned angle distribution for vertex v_m, i.e. $V_{i,m} = -\log \mathcal{N}(\phi_i; \mu_m, \sigma_m^2)$ as in Eq. 5. The value S_i is a cost of matching ϕ_i to the angle of a dummy vertex s, i.e. $S_i = -\log \mathcal{N}(\phi_i; \pi, \pi^2/16)$ as in Eq. 5.

Both minimizations in Alg. 1 can be performed incrementally in $O(N)$ time by remembering the summation value for previous j. The first minimization is performed N times, the second one $O(NM)$ times. Thus the time complexity of Alg. 1 is $O(N^2M)$. The Alg. 1 is called N times for variously shifted contour, i.e. for $d \in \{1 \dots N\}$. Thus the overall complexity of contour matching is $O(N^3M)$.

Algorithm 1 Contour matching algorithm

Input: $V_{i,m}$ = cost of mapping point p_i to vertex v_m
S_i = cost of mapping point p_i to segment s
Output: $T_{i,m}$ = cost of mapping sub-contour $(p_1 \dots p_i)$ to vertices $(v_1 \dots v_m)$
for all $i \in \{1 \dots N\}$ **do**

$$T_{i,1} \leftarrow \min_{j \in \{1 \dots i\}} \left(\sum_{k=1}^{j-1} S_k + V_{j,1} + \sum_{k=j+1}^{i} S_k \right)$$

end for
for all $m \in \{2 \dots M\}$ **do**
  for all $i \in \{m \dots N\}$ **do**

$$T_{i,m} \leftarrow \min_{j \in \{m \dots i\}} \left(T_{j-1,m-1} + V_{j,m} + \sum_{k=j+1}^{i} S_k \right)$$

  end for
end for
return $T_{N,M}$

4 Experiments

The proposed methods were tested on a dataset of spread garments collected at the Czech Technical University [14]. The dataset contains color images (as in Fig. 2) and depth maps taken by Kinect-like device from a bird's eye perspective. All images were manually annotated by specifying positions of landmark points which correspond to vertices of the proposed polygonal models in Fig. 4. The resolution of images is 1280×1024. The edge of 1 pixel approximately corresponds to 0.09 cm in real world coordinates. We used 158 testing images (29 towels, 45 pants, 45 short-sleeved shirts and 39 long-sleeved shirts). The algorithms were implemented mainly in Matlab. Some of the most time-critical functions were reimplemented in C++. The performance was evaluated on a notebook with 2.5 GHz processor and 4 GB memory.

The input images were downsampled to the resolution 320×256 for the purpose of segmentation. The smaller resolution preserves all desired details and significantly improves the time performance of the segmentation algorithm. Totally 153 of 158 input images were correctly segmented which gives 97% success ratio. The incorrectly segmented images were excluded from the further evaluation. The time spent by segmenting one image is on average 0.87 seconds.

The contour simplification algorithm is the most time consuming operation of the proposed pipeline. The running times can be seen in Tab. 1. They highly depend on the length of the contour which is induced mainly by the shape complexity of the particular category of clothing. The subsequent model matching procedure is working with the already simplified contour and thus it is very fast as seen in Tab. 1. The whole pipeline including also segmentation and contour simplification runs around 5 seconds in the worst case. This is a significant improvement compared to 30–150 seconds required just for model fitting which is reported by Miller et al. [9].

Table 1: Time performance (in seconds) of contour simplification phase and polygonal model matching phase for various categories of clothing.

Phase	Towel	Pants	Short-sleeved	Long-sleeved
Contour	1.33	3.95	0.64	1.88
Matching	0.01	0.01	0.03	0.03

Table 2: Displacements (in centimeters) of the identified vertices to ground-truth vertices found by polygonal model matching for various categories of clothing.

Error	Towel	Pants	Short-sleeved	Long-sleeved
Median	0.41	0.52	0.53	0.59
Mean	0.43	0.69	1.07	1.26
Std. dev.	0.23	1.00	1.40	1.79

Fig. 7: Displacements of the vertices found by model matching (plotted in green) and the manually annotated landmarks (red). The displacements were computed for various configurations of garments and then they were projected to the canonical image.

Tab. 2 summarizes displacements of vertices found by the proposed algorithm compared to the manually annotated landmark points. These errors are similar to those reported by Miller et al. [9]. They are small enough to determine the configuration of a piece of garment reliably and then use this information to manipulate the garment with robotic arms. Fig. 7 visualizes the displacements for the selected representatives of clothing. The errors were computed for various configurations of the same piece of garment and then they were projected to a canonical image. The biggest source of displacements are shoulders as seen in Fig. 7 for the green long-sleeved sweater. However, estimation of their position can be ambiguous even for a human. Moreover, their exact position is rather unimportant for automated manipulation. A few other significant errors were made while estimating armpits of a shirt with very short sleeves as seen in Fig. 7 for the white shirt. They are caused by indistinguishable shape of the sleeves on the contour. The contour resembles a straight line around the armpits.

The proposed algorithms were deployed to a real robot and successfully tested in several folding sequences of various garments, as seen in Fig. 1. The folding procedure succeeds approximately in 70% attempts. However, observed folding failures were almost never caused by the described vision pipeline. Main source of these failures lies in an unreliable grasping mechanism and in occasional inability to plan move of robotic arms.

5 Conclusion

We have fulfilled our goal and proposed a fast method allowing to recognize the configuration of a piece of garment. We have achieved a good accuracy, comparable to those of known approaches, despite the usage of a more challenging nonuniform background. The presented model has proved to be sufficient for the studied situation. The recognition procedure was deployed to a real robot and successfully tested in fully automated folding.

In the future, we would like to strengthen power of the model by introducing more global constraints. Our intention is to generalize the method to folded pieces of garment. We would also like to learn the robot how to detect folding failures and how to recover from them.

6 Acknowledgment

The authors were supported by the European Commission under the project FP7-ICT-288553 CloPeMa (J. Stria), by the Grant Agency of the Czech Technical University in Prague under the project SGS13/205/OHK3/3T/13 (J. Stria, D. Průša) and by the Technology Agency of the Czech Republic under the project TE01020197 Center Applied Cybernetics (V. Hlaváč).

References

1. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd ed. Wiley (2000)
2. Gonzalez, R.C., Woods, R.E., Eddins, S.L.: Digital Image Processing Using MATLAB, 2nd ed. Gatesmark (2009)
3. Hata, S., Hiroyasu, T., Hayashi, J., Hojoh, H., Hamada, T.: Robot system for cloth handling. In: Proc. Annual Conf. of IEEE Industrial Electronics Society (IECON). pp. 3449–3454 (2008)
4. Hu, X., Bai, Y., Cui, S., Du, X., Deng, Z.: Review of cloth modeling. In: Proc. SECS Int. Colloquium on Computing, Communication, Control and Management (CCCM). pp. 338–341 (2009)
5. Kita, Y., Kita, N.: A model-driven method of estimating the state of clothes for manipulating it. In: Proc. IEEE Workshop on Applications of Computer Vision (WACV). pp. 63–69 (2002)
6. Kita, Y., Ueshiba, T., Neo, E.S., Kita, N.: Clothes state recognition using 3d observed data. In: Proc. IEEE Int. Conf. on Robotics and Automation (ICRA). pp. 1220–1225 (2009)
7. Kolesnikov, A., Fränti, P.: Polygonal approximation of closed discrete curves. Pattern Recognition 40(4), 1282–1293 (2007)
8. Maitin-Shepard, J., Cusumano-Towner, M., Lei, J., Abbeel, P.: Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In: Proc. IEEE Int. Conf. on Robotics and Automation (ICRA). pp. 2308–2315 (2010)
9. Miller, S., Fritz, M., Darrell, T., Abbeel, P.: Parametrized shape models for clothing. In: Proc. IEEE Int. Conf. on Robotics and Automation (ICRA). pp. 4861–4868 (2011)
10. Orchard, M., Bouman, C.: Color quantization of images. IEEE Trans. on Signal Processing 39(12), 2677–2690 (1991)
11. Perez, J.C., Vidal, E.: Optimum polygonal approximation of digitized curves. Pattern Recognition Letters 15(8), 743–750 (1994)
12. Ramisa, A., Alenyà, G., Moreno-Noguer, F., Torras, C.: Using depth and appearance features for informed robot grasping of highly wrinkled clothes. In: Proc. IEEE Int. Conf. on Robotics and Automation (ICRA). pp. 1703–1708 (2012)
13. Rother, C., Kolmogorov, V., Blake, A.: Grabcut – interactive foreground extraction using iterated graph cuts. ACM Trans. on Graphics 23(3), 309–314 (2004)
14. Wagner, L., Krejčová, D., Smutný, V.: CTU color and depth image dataset of spread garments. Tech. Rep. CTU-CMP-2013-25, Center for Machine Perception, Czech Technical University (2013)
15. Wang, P.C., Miller, S., Fritz, M., Darrell, T., Abbeel, P.: Perception for the manipulation of socks. In: Proc. IEEE Int. Conf. on Intelligent Robots and Systems (IROS). pp. 4877–4884 (2011)
16. CloPeMa project – clothes perception and manipulation. `http://www.clopema.eu/`