Feature Extraction of Garments Based on Gaussian Mixture for Autonomous Robotic Manipulation

Tahir Rasheed[1], Abdul Attayyab Khan[2], Jamshed Iqbal[3,4,*]
[1]Ecole Centrale de Nantes, Laboratoire des Sciences du Numerique de Nantes (LS2N), Nantes, France
[2]Department of Electrical Engineering, Bahria University, Karachi, Pakistan
[3]Electrical and Computer Engineering Department, University of Jeddah, Jeddah, Saudi Arabia
[4]Department of Electrical Engineering, FAST National University of Computer and Emerging Sciences, Islamabad, Pakistan
[*]Corresponding author: jamshed.iqbal@nu.edu.pk

***Abstract*— Recent advancements in the domain of robotics have offered support to humans in their everyday activities with the aim of offloading workers from performing repetitive tasks. The present work highlights one such task in garment industry where the robot may find potential to manipulate different garments including developing a number of robotic skills like laundry pile sorting, garment stacking and garment folding/ unfolding. This paper is aimed to study the integration of hardware and software developed for ClopeMa Project on a human-friendly robotic platform, i.e. a Baxter robot that can safely operate side by side with humans. In particular, the paper discusses integration of RGB-D sensor with the ROS environment and studies utility of garment manipulation. The goal is to present a working platform which can autonomously recognize the configuration of a piece of garment spread out on a flat surface. The algorithm for recognizing the garment consists of first applying Gaussian mixture model (MoG) for background subtraction and then using polygonal approximation to acquire feature points for the foreground of the garment. The proposed algorithm is tested online through series of experiments on towel, pants and t-shirts of various colors and materials. Results under varying lightening conditions witness robustness of the proposed scheme.**

Keywords*— Garment manipulation; Gaussian mixture model; Industrial automation; Robotics applications

I. INTRODUCTION

Present research in the domain of robotics is broadly focused in developing robotic skills, which are similar or superior to humans with the ultimate aim of offloading humans from performing repetitive tasks [1]. Among the possible applications, this work highlights requirements for garment autonomous manipulation. This involves analysis of the given garment to detect feasible grasping points which can be used to manipulate different garments and develop different robotic skills such as laundry pile sorting according to color and material, garment stacking and garment folding/ unfolding [2]. There are number of methods for the analysis of a given piece of garment but this paper is based on the method developed by clothes perception and manipulation (ClopeMa) project [3] which involves geometric approach for the given piece of garment. This approach includes a complete pipeline for clothes configuration recognition by estimating positions of the most important grasping points (e.g. all four corners of a towel). This geometric strategy involves various fast and dynamic programming based methods to obtain efficient results during analysis of the garment.

The setup used for experiments consist of a Baxter robot [4] having an RGB-D sensor mounted on the top of the robot. In front of the robot and within the range of the sensor, a white table has been used as a support for analyzing different garments laying on it. The whole pipeline for the adopted method is given below.

A. Input Acquisition

The input is a single color image of a piece of garment lying on the table. The original model of the cloth (e.g. towel, pant and shirt) is assumed to be known in advance. The image is taken from a RGB-D sensor attached to the top of the Baxter robot. As the position of table and camera is fixed, the analysis is done only on the cropped image of the table while discarding the area outside the table given by the image.

B. Background Subtraction

This is relatively an important step which requires robustness in the algorithm. The goal of this step is to segment the foreground (garment) from the background (table) by assuming significantly different color of the garment in contrast to table. The color properties of the table can be learned from the data (different images of the background) which is called as background learning [5]. The background learning makes this step more robust that can solve many noise issues originated during image acquisition the camera. As an outcome of this step, a gray scale image containing only the garment (as maximum pixel intensity) with a black background is obtained.

C. Contour Detection

The gray scale image is then processed through Canny edge detection algorithm to get the contours of the garment. An alternate option is to apply Moore's algorithm [6] for tracing connected boundary of the region. The difference between both methods in that Canny gives edges which can be broken from several places if the result from the background subtraction is not efficient. In this case, these edges cannot be considered as a contour. So, performance of Canny algorithm is adequate provided background subtraction results are reasonably efficient [7]. Moore's algorithm gives a single connected contour having the largest connected area.

D. Contour Simplification and Approximation:

The contour obtained in the last step is very dense and contains many unnecessary points which are of no use to us. This contour is simplified based on polygonal approximation and can get the simplified contour points which can be considered as the vertices of the given garment. The approximation calculates the slope of the boundary (contour) and marks a point as a vertex when it detects a significant change in the slope.

E. Matching:

We assumed that the model of the garment [8] is initially known. The approximated model obtained is matched with the known model defined for the corresponding type of the garment. The matching procedure involves finding the correspondence between the approximated vertices and landmark points defining the specific garment model. The matching considers mainly the local features of the approximated model. If the number of vertices are more than the landmark points of the model, unmatched vertices are discarded.

II. Background Subtraction

A robust background subtraction algorithm should handle lightening changes, repetitive motions and long-term scene changes. These problems are usually addressed by making the background model adaptive so that its parameters can track illumination changes. Also, increasing the complexity of the model results in accurate representation of multimodal backgrounds. The algorithm reported in [9] is an adaptive method which uses a mixture of Gaussian distributions to model a multimodal background image sequence. For each pixel, each normal distribution in its background mixture corresponds to the probability of observing a particular intensity or color in the pixel. The background is generated by multiple surfaces appearing in the pixel view. Each surface is represented by a normal distribution having a mean equal to the surface intensity or color and a variance due to surface texture, illumination fluctuations or camera noise.

The first step in the background subtraction is to learn the background color which is a probabilistic distribution of RGB values of the background pixels. The distribution is represented as a mixture of K Gaussians i.e.

$$p(X) = \sum_{k=1}^{K} w_{k,t}\, N(z\,;\, \mu_k\,, E_k\ ) \tag{1}$$

$$N(z\,;\, \mu_k\,, E_k\ ) = \frac{e^{-\frac{1}{2}(x-\mu)^T E_k^{-1}(x-\mu)}}{\sqrt{(2\pi)^3 E_k}} \tag{2}$$

where N is a normal distribution having mean vector $w_{k,t}$ and a covariance matrix E_k. $w_{k,t}$ are weights of the Gaussians for kth component. Number of components (k) for Gaussian Mixture Model (GMM) is determined empirically based on the number of visible clusters in RGB. e.g. in one of the cases considered in the present work, background is completely white (table) while the foreground is completely black (pant). So, only two GMM components are required. GMM forms Gaussians taking into account all the colors in the image. From the Gaussians, the background color and foreground color can be easily judged in the present instance. The background is modeled as a mixture of adaptive Gaussians. Mixture and adaptive natures are due to multiple background colors and lightening conditions respectively.

The Gaussian's parameters are evaluated using a sequence of images of the background. At each iteration, Gaussians are evaluated using a simple heuristic rule to determine which ones are most likely to correspond to the background. At the end of this procedure, foreground is constructed by the image pixels which are not matched with the background Gaussians.

The background model is obtained as follows: In the first step, the Gaussian distribution for all the pixels $\{X_1, X_2 \ldots\ldots\ldots X_n\}$ is calculated on the first image of the background. These initial Gaussian parameters are then updated using other images in a learning process based on K-means approximations detailed in the following procedure.

If a new pixel X_{t+1} has been matching with the already existing Gaussian's (within the limit of 2.5σ) then Gaussian $\mu_{i,t+1}$ and $\sigma^2{}_{i.t+1}$ are updated as follows

$$\mu_{i,t+1} = (1\text{-}\rho)\mu_{i,t} + \rho X_{t+1} \tag{3}$$

$$\sigma^2{}_{i.t+1} = (1\text{-}\rho)\sigma^2{}_{i.t} + \rho(X_{t+1} - \mu_{i,t+1})^2 \tag{4}$$

where $\rho = \alpha N(X_{t+1}\,;\, \mu_{i,t}\,, \sigma^2{}_{i.t}\ )$ and α is a learning rate. The weights of the Gaussians are then updated by

$$w_{i,t+1} = (1\text{-}\alpha)w_{i,t} + \alpha M_{i,t+1} \tag{5}$$

where $M_{i,t+1}$ = 1 for matching Gaussian and 0 for all the others. If the new pixel X_{t+1} is not matched with the already existing Gaussian, then the Gaussian for that specific pixel is not updated. After this step, we are proceeding heuristically to choose the Gaussian which has most supporting evidence and less variance. So the Gaussians are ordered by the value of w/σ. Then the first B distributions are chosen as a background model i.e.

$$\mathrm{B} = argmin_b\left(\sum_{i=1}^{b} w_i > T\ \right) \tag{6}$$

where T is the minimum portion of the image that has to be a background (threshold). Fig. 1 is the initial background which is the empty table while Fig. 2a is the captured image having a pant on the table at time t (These images belong to ClopeMa dataset. In this case we only took one background because these are simple images having no problem with the noise). By applying the background subtraction the results obtained are shown in Fig. 2b. It is a gray scale image which contains only foreground (pant).

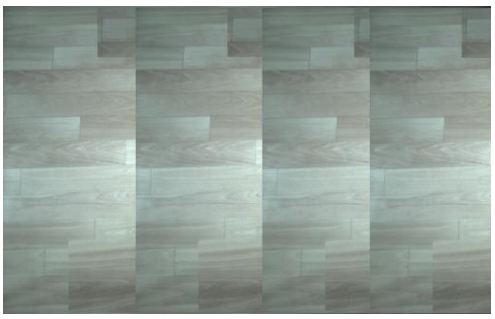


Fig. 1. Background table.

(a) (b)

Fig. 2. Background subtraction: (a) Image acquired through camera (b) Result.

III. Contour Detection

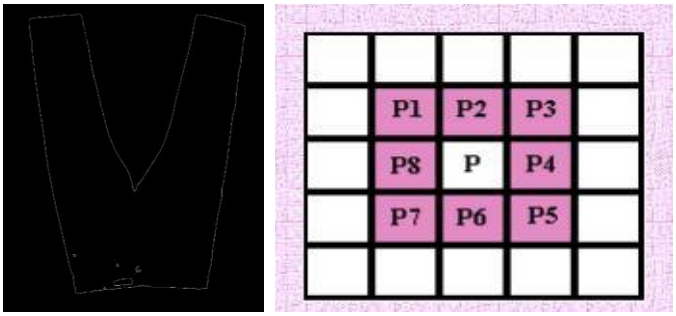
The background subtraction resulted in a gray scale image that contains the foreground only. This gray scale image is then processed to find the connected boundary of the foreground, which can be achieved by two different methods:

A. Edge Detection

One way to get the connected boundary of the foreground image is to process the image based on Canny edge detection algorithm. The result can be seen in the given Fig. 3a.

B. Moore's Approach

The idea behind Moore-Neighbor tracing [10] is simple. The Moore neighborhood of a pixel P is the set of 8 pixels which share a vertex or edge with that pixel. These pixels are named as {P1, P2, …, P8} in Fig. 3b. The Moore's neighborhood is also known as the 8-neighbors or indirect neighbors.



(a) (b)

Fig. 3. Contour detection: (a) Result of edge detection (b) Moore neighborhood of pixel P.

Given a digital pattern i.e. a group of white pixels on a background of black pixels (a grid), a white pixel is located and is declared as 'start' pixel. This can be done by starting from the bottom left corner of the grid, scanning each column of pixels from bottom to upwards and from leftmost column to the right until a black pixel is encountered. Being on the start pixel, the contour is extracted by going around the pattern in a clockwise direction without loss of generality. The traversal direction can be arbitrary provided it is consistent throughout the algorithm.

The general idea is that every time a white pixel P is hit, we need to backtrack i.e. go back to the previous black pixel and then go around P in a clockwise direction, visiting each pixel in its Moore neighborhood until a white pixel is encountered. The algorithm terminates when the start pixel is re-visited. The visited white pixels form the contour of the pattern. The results of Moore's algorithm on Fig. 2b is shown in Fig. 4.

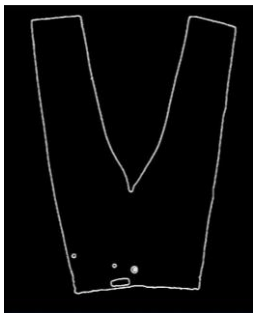


Fig. 4. Result of Moore's algorithm.

The results from Moore's algorithm are much better than the Canny because Moore deals with the connected boundary while Canny deals only with the edges which may or may not be connected.

IV. Contour Simplification And Approximation

After tracing the contour in Section III, next step is to simplify this contour and to get only the significant points on the contour. The number of distinct points L (where L contains $(q_1, q_2, \dots q_L)$ points) on the contour depends upon the image resolution as well as on size of the garment piece. Typically, L has an order of hundreds or thousands. We need to simplify the contour by approximating it with a polygon having N vertices with N<<L. The objective is to select a subsequence of N points $(p_1, p_2, \dots p_N)$ that is a subset of $(q_1, q_2, \dots q_L)$. Additionally we want to minimize the sum of Euclidean distances of the original points $(q_1, q_2, \dots q_L)$ to edges of the approximated polygon $(p_1, p_2, \dots p_N)$ as seen in Fig. 5. Simplification procedure for the garment contour is based on the dynamic programming algorithm for the optimal approximation of a close curve [11]. This algorithm iteratively computes the optimal approximation of points $(q_1, q_2, \dots q_i)$ by n vertices from previously found approximation of $(q_1, q_2, \dots q_j)$ where $j \in \{n-1 \dots i-1\}$ by *n-1* points as demonstrated in Fig. 6.

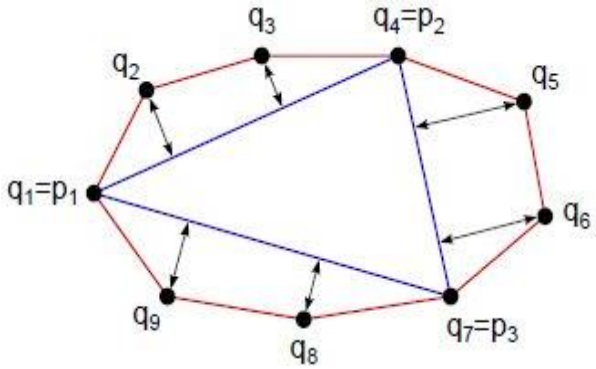


Fig. 5. Contour approximation problem.

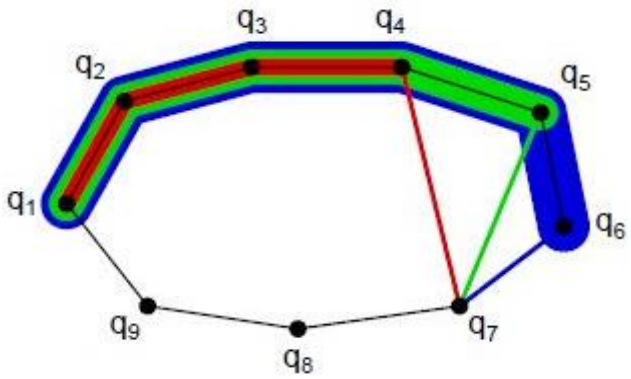


Fig. 6. Dynamic problem solution.

In Fig. 5, the original contour $(q_1, q_2, \dots q_L)$ plotted in red is simplified with a polygon in blue. The points $(p_1, p_{2,} \dots p_N)$ are the simplified points approximated by a polygon (in blue) while minimizing the distances of the original points q_i to polygon edges. In Fig. 6, the dynamic programming algorithm for polygonal approximation utilizes previously constructed approximation of points $(q_1, q_2, \dots q_6)$ by *n-1* vertices to obtain approximation of next points $(q_1, q_2, \dots q_6,\ q_7)$.

Here, the N-vertex polygonal contour P obtained from Moore's algorithm is approximated into another polygonal contour Q with the minimum number of line segments M such that the approximation error $E(p)$ is less than error bound (can be around 2%) [12]. $E(p)$ is defined as a Euclidean distance from the vertices of the curve P to the approximated line segment of curve Q.

$$E(P) = max_{1\le m\le M} d(q_m, q_{m+1}) \tag{7}$$

where

$$d(p_i, p_j) = max_{i\le k\le j} k; d(p_i, p_j)$$

and

$$q_m = p_i,\ q_{m+1} = p_j$$

The approximation [11] for the closed curve P with the fixed starting point can be found by first converting into a feasibility graph which is constructed on vertices of the curve P for the given error tolerance. Nodes $V = \{v_1, v_2, \dots\ v_n\}$ of the graph G1 are vertices $\{p_1, p_2, \dots\ p_n\}$ of the curve P. A pair of nodes v_i and v_j is connected by an edge if the approximation error for the curve segment $\{p_i, p_{i+1}, p_j\}$ by the line segment (p_i, p_j) is less than a given error tolerance i.e. $d(p_i, p_j) \le \varepsilon$.

The solution to this problem lies in determining the shortest path in the feasibility graph G1. To find this path in a directed acyclic graph, we introduce 1D discrete state space $\Omega_1 = w_n : n = 1, .. N$. Every point in this state space represents the sub-problem of the node in the graph G1. The cost function $C(n)$ is given as the minimum number of edges in the shortest path and is calculated by dynamic programming for all *n=1,2,....N.*

$$C(n) = min^{1\le j\le n}_{(v_j, v_n)\in G_1} C(j) + 1$$

A. *Bellman and Ford Algorithm*

The approximation algorithm computes final contour points which are the cost of the cheapest paths from a starting node to all other nodes in the graph. Thus, the paths afterwards can also be constructed. The algorithm considers each q points as nodes and tries to find the shortest path by satisfying the condition that the approximation error $E(p)$ should be less than error bound. The first estimate is:

- The starting node has cost 0, as its distance to itself is obviously 0.
- All other nodes have cost as infinity, which is the worst estimate possible.

From Fig. 7, the subset of total contour points (in the box) has been chosen. Based on which, the algorithm finds the shortest path which leads to polygonal approximation. Afterwards, the algorithm checks every edge for the condition if cost of the source of the edge plus the cost for using the edge (considered as unity) is smaller than the cost of the edge's target i.e.

$$if\ dist[v] > dist[u] + 1, \text{then}$$
$$Update\ dist[v]$$
$$dist[v] = dist[u] + 1$$

Finally, the updates for each edge predict the contour points from the set of q points that we have found from Moore's algorithm.

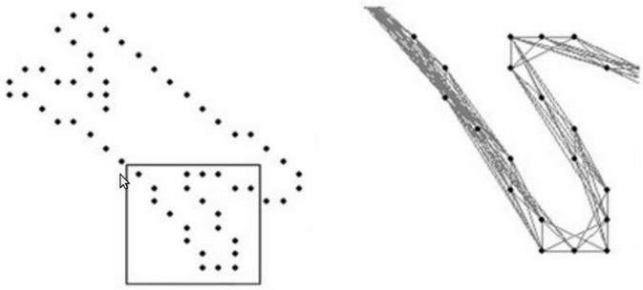
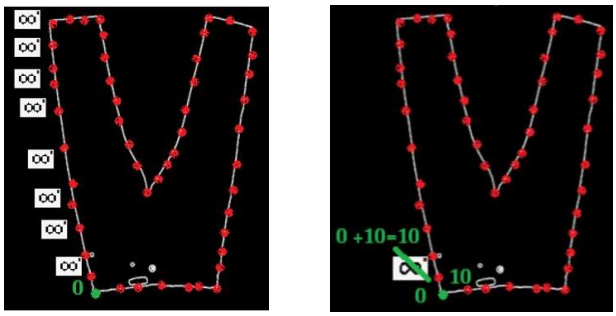


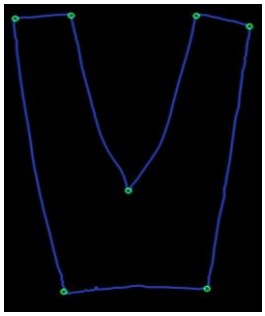
Fig. 7. Bellman and Ford algorithm.

B. *Working of Dynamic Programming*

The algorithm calculates shortest paths in a bottom-up fashion. It first calculates the shortest distances for the shortest paths which have at-most one edge in the path. Then, it calculates shortest paths with at-most 2 edges, and so on. After the i^{th} iteration of the outer loop, the shortest paths with at most i edges are calculated. A simple path can contain maximum of |V| – 1 edges, so the outer loop runs |v| – 1 times. Assuming that there is no negative weight cycle, if we have calculated shortest paths with at most i edges, then an iteration over all edges guarantees to give shortest path with at-most $i + 1$ edges. The determined shortest path is the required contour points. The working of the algorithm can be illustrated by Figs. 8 (a-c).



(a) (b)



(c)

Fig. 8. Algorithm: (a) Initialization (b) Update list. (c) Contour points determined by dynamic programming.

V. Feature Matching

Since the model of the garment on the table is assumed to be known, each model is determined by the vertices of the specific garment. Following the marking of vertices in the known model, we need to match these features (vertices) with the obtained simplified vertices for the specific garment on the table. So, we have two sets of feature points; one from the known model F and other from the contour approximation G. Mathematically,

$$F_{I1} = \{(x1_i, y1_i, N1_i)\}_{i=1}^{N}$$

$$G_{I2} = \{(x2_j, y2_j, N2_j)\}_{i=1}^{M}$$

where $X = [x, y]$ are positions of features in the image while $N1, N2$ are the image patches around feature points which are centered at (3*3). The neighborhood 3*3 image patch can be seen in Fig. 3b. The adjacency matrix is then calculated by,

$$E(i,j) = e^{-||X1_i - X2_i||^2/\sigma^2}$$

where E is N*M matrix measuring the degree of closeness between the feature F_{I1} in image I_1 and feature G_{I2} in image I_2. σ is the standard deviation. The matrix E has values in the range [0, 1] and is used to calculate affinity matrix by

$$A(i,j) = E(i,j) * \frac{1}{2}(\emptyset_{NCC}(N1_i, N2_j) + 1)$$

where $\emptyset_{NCC}$ is termed as Normalized Cross Correlation (NCC) which is frequently used in computer vision. It is used to get rid of the big differences between pixel positions that affect the overall efficiency of the algorithm. NCC is calculated for each pixel. For pixel f, NCC $(\hat{f})$ is calculated as

$$\hat{f} = \frac{f - \bar{f}}{\sqrt{\sum(f - \bar{f})^2}}$$

where, $\bar{f}$ is the mean of overall image and $\sqrt{\sum(f - \bar{f})^2}$ is standard deviation. The matrix A also has values in the range [0, 1]. Each row in A is examined to find the maximum element. Then it is evaluated if the same element is also maximum in the corresponding column of A. Holding this condition true implies that we the match is found.

VI. Results

The algorithm has been applied on images from ClopeMa dataset using RGB-D sensor and on-line results have been acquired. Initially, several background models have been formulated with an empty table for 'learning background'. Then the experiments are done on-line on the towels, pants and t-shirts of various colors and materials. In these results, green circles show the vertices of the given garment which are actually the feature points of the specific garment.

Setup for the experiments is same as mentioned in Section I. The images acquired from RGB-D senor are at the rate of 30 frames/second while the machine used for processing of the images is Intel core i7 CPU 930 @ 2.80 GHz (64-bit) with 8 GB RAM. For each garment category but with different materials, 2-3 experiments have been done. For the sake of brevity, results corresponding to one material for each garment is presented. Since we have not performed model matching, so the algorithm is detecting the vertices even if the garment is not completely flat on the table. Fig. 9 shows results on a t-shirt orientated on the table. 8-10 points have been detected depending on the flatness of the shirt on the table.

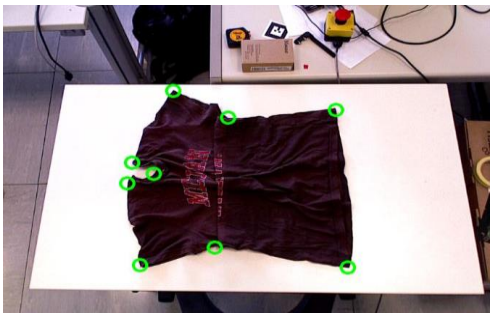
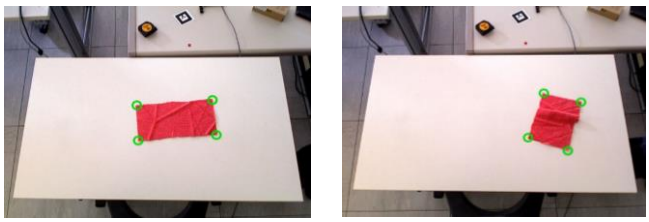


Fig. 9. Contour point for t-shirt.

Fig. 10 presents results of application of the algorithm on black and yellow towel (Knitwear material) orientated at various angles with four feature points. Fig. 11 shows results on the red towel (Cotton toweling material) orientated at different angles while a blue jean spread in multiple orientations is illustrated in Fig. 12. As can be seen that the pant contains seven feature points.

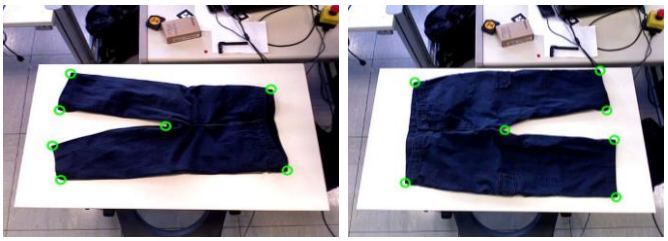
(a) (b)

Fig. 10. Contour points for knitwear cloth:
(a) Orientation 1 (b) Orientation 2



(a) (b)

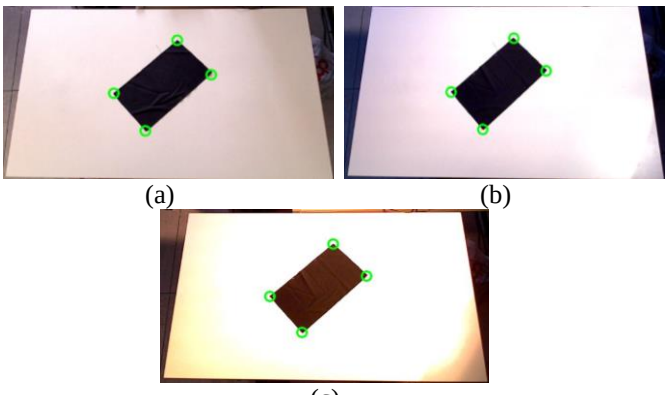
Fig. 11. Contour points for knitwear cotton towel:
(a) Orientation 1 (b) Orientation 2



(a) (b)

Fig. 12. Contour points for knitwear jeans
(a) Orientation 1 (b) Orientation 2

Thanks to GMM, the presented algorithm is also capable of recognizing correct vertices even by changing the lightening conditions. Varying these conditions, Fig. 13 (1-c) presents the results of contour points corresponding to low, medium and high lightening. One can see that consistent results are obtained even though the conditions are different. This implies that the proposed algorithm is robust enough to take care of noise, shadows and lightening issues.



(a) (b)

(c)

Fig. 13. Feature extraction under various lightening conditions: (a) Low (b) Medium (c) High

VII. Conclusion

We have presented an algorithm based on the ClopeMa project which can provide the feature points for nearly all type of garments. The robustness of the algorithm is evident by its performance in variable lightening condition. The primary limitation offered by this approach is constituted by the fact that we need a background significantly different from the garment. In case the portion of background i.e. color of table is similar to that of color of the garment, the performance of the presented approach highly depends on the size of the matching portion. In worst cases, the algorithm may give feature points within the cloth by considering the boundary of the matching portion as the boundary of the whole cloth. This may lead to failure of the method due to wrong detection of vertices. Thus, the key assumption of proper working of the approach is that the background color should be different that garment color.

In future, this algorithm can be used for calculating the transformation from the camera to the pixel location to get the 3D point of the features which can be passed to Baxter for the development of numerous robotic skills.

References

[1] J. Iqbal, R. U. Islam, S. Z. Abbas, A. A. Khan and S. A. Ajwad, "Automating industrial tasks through mechatronic systems – A review of robotics in industrial perspective", Tehnicki Vjesnik-Technical Gazette, vol. 23, no. 3, pp. 917-924, 2016

[2] J. Stria, D. Prusa, V. Hlavac, L. Wagner, V. Petrik, P. Krsek and V. Smutny, "Garment perception and its folding using a dual-arm robot", IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 61-67, 2014

[3] "CloPeMa (Clothes Perception and Manipulation) research project," http://www.clopema.eu

[4] E. McLeod, "The factory robot of the future: Baxter is the world's first low-cost, user-friendly factory robot", Short article, SAGE Business Researcher, pp. 1-4, 2017

[5] T. Bouwmans, L. Maddalena and A. Petrosino, "Scene background initialization: A taxonomy", Pattern Recognition Letters, In Press, DOI: 10.1016/j.patrec.2016.12.024, 2017

[6] R. C. Gonzalez, R. E. Woods and S. L. Eddins, Digital Image Processing Using MATLAB, 2nd ed, ISBN: 0982085400, Gatesmark Publishing, 2009

[7] P. Jiménez, "Visual grasp point localization, classification and state recognition in robotic manipulation of cloth: An overview", Robotics and Autonomous Systems, vol. 92, pp. 107-125, 2017

[8] S. Lu, P.Y. Mok and X. Jin, "A new design concept: 3D to 2D textile pattern design for garments", Computer-Aided Design, vol. 89, pp. 35-49, 2017

[9] Orchard, M., Bouman, C.: Color quantization of images. IEEE Transactions on Signal Processing, vol. 39, no. 12, pp. 2677-2690, 1991

[10] T. Liu, A. W. Moore, A. Gray and K. Yang, "An investigation of practical approximate nearest neighbor algorithms", Advances in Neural Information Processing Systems, 2004

[11] A. Kolesnikov and P. Franti, "Polygonal approximation of closed discrete curves", Pattern Recognition, vol. 40, no. 4, pp. 1282-1293, 2007

[12] J. C. Perez and E. Vidal, "Optimum polygonal approximation of digitized curves", Pattern Recognition Letters, vol. 15, no. 8, pp. 743-750, 1994