Graph-Based Robust Shape Matching for Robotic Application

Hanbyul Joo, Yekeun Jeong, Olivier Duchenne, Seong-Young Ko, In-So Kweon

***Abstract*— Shape is one of the useful information for object detection. The human visual system can often recognize objects based on the 2-D outline shape alone. In this paper, we address the challenging problem of shape matching in the presence of complex background clutter and occlusion. To this end, we propose a graph-based approach for shape matching. Unlike prior methods which measure the shape similarity without considering the relation among edge pixels, our approach uses the connectivity of edge pixels by generating a graph. A group of connected edge pixels, which is represented by an "edge" of the graph, is considered together and their similarity cost is defined for the "edge" weight by explicit comparison with the corresponding template part. This approach provides the key advantage of reducing ambiguity even in the presence of background clutter and occlusion. The optimization is performed by means of a graph-based dynamic algorithm. The robustness of our method is demonstrated for several examples including long video sequences. Finally, we applied our algorithm to our grasping robot system by providing the object information in the form of prompt hand-drawn templates.**

I. INTRODUCTION

Object detection is fundamentally important to find a target object in an input scene. Shape information plays an important role in object detection. The human visual system can often recognize an object on the basis of the object's 2-D outline shape alone. Objects belonging to the same category have similar shapes even if their textures or colors might be completely different. Because of this property, shape information has been used in computer vision area for object detection and categorization [1], [2], [3], [4], [5]. With regard to an interface for robotic applications, shape-based object detection is particularly useful because the input information could be in a simple template form such as Fig. 1(a). In this framework, without giving the preprocessed object image by removing the background clutter, a user can inform the robot about the target objects by means of prompt hand-drawn templates.

Shape-based object detection methods compare each region of the target edge map to the prior shape information (template) of the object. In order to compare the shape similarities of different regions, the edge pixels corresponding to the template are selected from all the edge pixels in each region. Subsequently, the cost is calculated by measuring the similarity between the prior shape information and the selected corresponding edge pixels. Finally, the area having the least cost is treated as the target object position.

According to the form of prior shape information, two approaches for shape-based object detection can be considered as follows. In one approach, the whole outline of an object is used for prior information as a rigid template form [3], [4], [5]. Because the template is a rigid form, multiple templates are required to handle shape variance or scale change. The mai n advantage of this approach is the simplicity of the prior model. Only a simple template without learning is required for detection. Our algorithm falls into this approach using whole object outline as a rigid template. In another approach, [1] and [2] use a group of contour fragments as prior information and detect the object with the best combination and arrangement of fragments. Each contour fragment has some degree of freedom under constraints according to the object model. This approach has an advantage in that it can detect articulated bodies such as humans and horses because each part can be found independently. However, this approach requires prior learning stage to generate reasonable contour fragments and object models.

In both the whole-outline-based approach and the contour-fragment-based approach, the correspondences between the contour (whole or fragment) and target edge pixels are important to measure the shape difference. Depending on the correspondence, the shape similarity cost can be low in the background clutter, and it can be high even in the true object position. Fig. 1 shows a typical example of this problem.

If we consider a pixel-by-pixel correspondence between the template and the edge edge map, it become extremely difficult to distinguish the true object from the background clutter; for example the red in Fig. 1 (c). Note that the

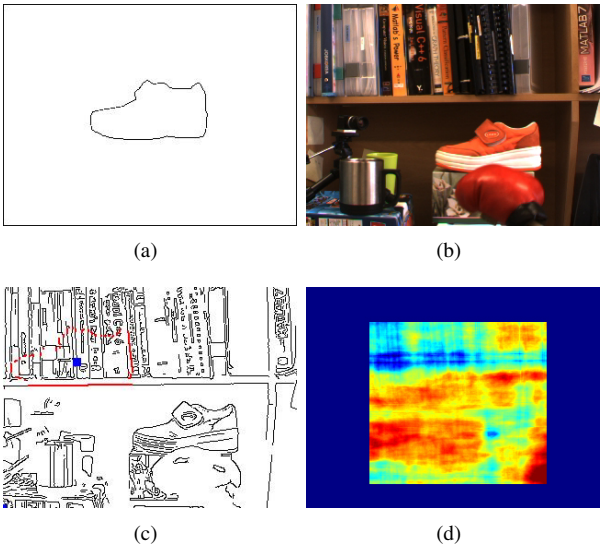


Fig. 1. Challenging example of shape matching (a) template (b) input image (c) lowest cost position(a blue rectangle) using Chamfer measurement and the edge pixels compared to template (red pixels) (d) dissimilarity cost map for every position using Chamfer measurement (outer regions are omitted because some parts of the template are located in the outside of the image boundary)

shape formed by the set of these pixels looks like a shoe, even if there is no actual shoe . This problem is caused by the combination of unrelated small pixels. The red pixels are actually the small parts of several vertical lines on a bookshelf. Therefore, if this pixel set is selected for shape comparison, the shape similarity cost might be low, and mismatch might occur. Fig. 1 (d) shows the cost map using Chamfer measurement([6], [7]) representing similarity cost at every position. The red regions are the areas with high cost, which means that the shapes are completely different from the template at those positions. The blue regions are the areas with low cost, which means that the shapes in these areas are very similar to the template. Note that the costs in the cluttered regions are even lower than the true object position.

In the human visual system, the edge connectivity is a very important factor. Thus, the connected pixels, not the unrelated small parts, can be considered together for comparison. However, if we assume the correspondence with the template in pixel level, it becomes difficult to consider the related pixels together. To deal with this problem, we propose an algorithm considering the connectivity of the edges. The proposed algorithm prevent unrelated combination of edge fragments. In our approach, we group edge pixels according to their connectivity. All edge pixels in the same group are included or excluded together when calculating the cost. This grouping is done from bottom-up a segmentation result. Therefore, the cost at each position is calculated by finding a combination of edge pixel groups with the lowest cost. In this work, we determine the optimal combination of edge pixel groups using the graph-based approach and dynamic programming. Our algorithm increases the gap between the true object matching cost and the false matching cost. Our algorithm is robust for occlusion and template variation. The experiments for several examples including long video sequence in section IV demonstrated the robustness of our algorithm. We also applied our algorithm to the grasping system explained in section II. Using the property of shape matching, a user can inform the robot about a target object by means of immediate hand-drawn templates.

II. System Configuration and Motivation

This work is a part of out research on an intelligent grasping system shown in Fig. 2. Our system consists of a robot arm (Fig. 2 (b)) and a stereo camera (Fig. 2 (c)), both of which are installed on a mobile robot as shown in Fig. 2. Our robotic arm has a 6-DOF arm and a 3-DOF gripper. To detect an object for grasping, we use the shape-based approach for the following reasons. Firstly, objects with similar shapes can be detected although their colors or textures are completely different. Secondly, we can use a hand-drawn outline to select the object for grasping. Instead of making an image database for each object in advance, we can choose the target object by a prompt hand-drawn outline shape.

The detailed scenario of our grasping system is as follows. First, the template image, which is the silhouette image of

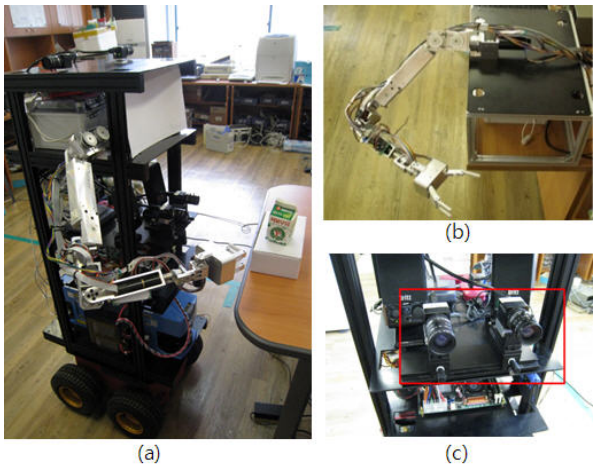


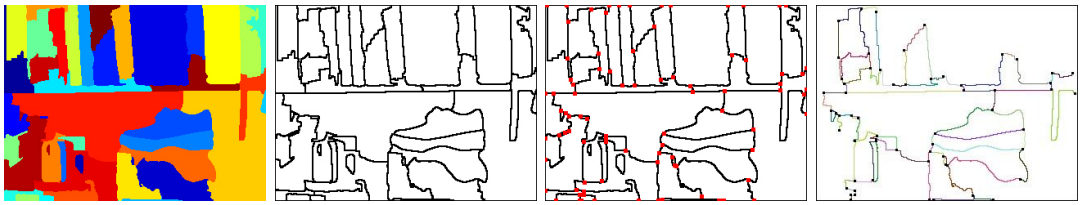
Fig. 2. Intelligent grasping system using shape-based object detection (a) overall system (b) 6-DOF robot arm (c) stereo camera

the desired object, is chosen. Using this template image, the positions of the desired object in both the left and right images from the stereo camera can be extracted; here the proposed shape matching algorithm is used. The position values are utilized to calculate the 3D object position in terms of the camera coordinate through a triangulation method. Then, the position is calculated in terms of the world coordinate. Finally, the robot calculates the proper grasping position by using prior knowledge about the relation between the measured center point and the grasping point of the object. After determining the grasping position, the robot moves and grasps the object. In our experiment, we shows the experimental results of our grasping system by using the proposed shape matching algorithm.

However, our previous research [8] using oriented shape matching algorithms [2] tends to fail if the template shape is even slightly different from the that of object. Therefore, in case such as our application, which uses a hand-drawn template, the matching accuracy might be worse because the cost of background clutter is often lower than that of the true object position.

III. Proposed Algorithm

Given an input image, we first divide the image into small homogeneous regions using an over-segmentation algorithm. From the a over-segmented result, we generate an undirected graph. Each edge of the graph represents a group of edge map pixels (we use the term ”edge” to denote the edges of a graph, and the term ”edge pixel” to denote the edge map pixels). Each edge of the graph,not each edge map pixel, is the unit for shape comparison with the template subpart. The weight of an edge means the shape similarity cost according to its corresponding template part, and the cost is calculated using the average orientation difference and Euclidian distance. A closed loop in the graph represents a region in the input image, thus the weighted sum of the edge weights belong to the loop means the shape similarity between the region and the template. Finally, the most similar shape is found among



(a) (b) (c) (d)

Fig. 3. Generating a graph from an over-segmented image. (a) an over-segmented image (b) an edge map extracted from the over-segmented image (c) extracted nodes (red rectangles) (d) final graph (the edges passing the image boundary are omitted).

all possible closed loops in the graph using the dynamic programming technique based on a shortest path algorithm. In this section, we explain the proposed graph-based shape matching algorithm in detail.

A. Graph Generation from an Over-segmented Image

In this section we explain how to generate a undirected graph $G(E, V)$ from an over-segmented result. In section III-D, we will transform $G(E, V)$ into a directed graph by taking the direction of each edge according to its correspondence to the template.

For the first step. we divide an input image into small segments by using an over-segmentation algorithm. Because the over-segmentation algorithm merges homogeneous region into one segment, we can assume that there is a discontinuity at the boundary between two segments. Therefore, we generate an edge map by extracting the pixels at the boundary between two segments. Fig.3(a) shows an over-segmented result generated from Fig.1(b), and Fig.3(b) is an edge map extracted from Fig.3(a).

An undirected graph G generated from the over-segmentation result is composed of a node set V and an undirected edge set E. We define the nodes in V as the points at which three or more segments meet. These points are the same as the junction points in the edge map. The extracted nodes are shown in Fig.3 (c). The edges in E are defined between two connected nodes. Conceptually, the edge E_{pq} which connects two adjacent nodes V_p and V_q represents the group of edge pixels that connect the two nodes in the edge map. Further, each group of edge pixel belong to each edge is the smallest unit for shape comparison. Fig.3 (d) shows the final graph with each edge drawn different color.

This graph definition has a meaningful property. In an over-segmented image, we can find the object by piecing together the segments. In the view of a graph, we can consider the same object region by finding a closed loop in the generated graph. Therefore, the problem of finding an object by piecing together the segments is equivalent to the problem of finding the closed loop in the graph.

The weight of a closed loop in the graph is an approximation of the similarity cost between the template and all edge map pixels belong to the edges composing the loop. Because we grouped connected edge pixels into an edge in E, this approach can prevent the use of unrelated combination of small fragments for the similarity cost calculation. In the proposed approach, only those edge pixels considered as a part of the object outline are used for the similarity cost calculation.

B. Edge Weight using the Average of the Euclidean Distance and the Orientation Difference

The weight of each edge is calculated by comparing the similarity to the corresponding subpart of the template T. Because the correspondence is changed according to the template position, the cost of each edge is also changed according to the template position. We refer to the template translated by τ as $\tau(T)$ (The translation τ can also be a transformation if we want to cover the rotation or the skew of the the object shape). For each positioned template $\tau(T)$, the weight of each edge is calculated and the optimal similarity cost, which is the lowest cost, is calculated using the optimization algorithm explained in III-D. The final object position is the lowest cost position.

To calculate each edge weight, we determine the template subpart that corresponds to the edge by using its two end nodes. For an edge $E_{pq} \in E$ and its end nodes V_p and V_q, we find the nearest template points $\tau(T_p)$ and $\tau(T_q)$. We assume that E_{pq} corresponds to the template subpart between $\tau(T_p)$ and $\tau(T_q)$, $\tau(\overline{T_{pq}})$. The weight of edge E_{pq} is determined using the average Euclidean distance and the orientation difference as follows.

$$W_{E_{pq}} = max\left(d(E_{pq}, \tau(\overline{T_{pq}})), d(\tau(\overline{T_{pq}}), E_{pq})\right) \quad (1)$$

where,

$$d(X, Y) = \frac{1}{N_X} \sum_{x_i \in X} \parallel x_i - y^{x_i} \parallel + \lambda |o(x_i) - o(y^{x_i})|. \quad (2)$$

Equation 2 is oriented Chamfer measurement [2]. The x_i and y are the pixel elements of X,Y and y^{x_i} is the nearest pixel in Y from x_i. $\parallel \cdot \parallel$ is the Euclidian distance. And, $o(\cdot)$ is orientation of the tangent line; therefore, the seconde term of (2) means the orientation difference between an element of X and its nearest point in Y. λ is the constant weight for the orientation difference, and N_x is the number of pixels in X. Conceptually, the cost $d(\cdot)$ means the average Euclidean distance and orientation difference between each pixel of X and its nearest pixel of Y. Note that $d(\cdot)$ is not a commutative function, that is, $d(X, Y) \neq d(Y, X)$. $d(X, Y)$ is the cost calculated by finding the nearest edge pixel from X to Y, while $d(Y, X)$ is the cost calculated in the opposite way. We

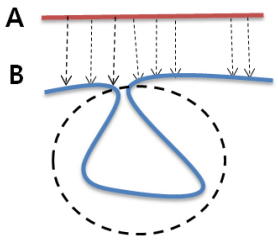


Fig. 4. Calculation for d(A,B). The dotted arrows indicate the nearest pixels from A to B. The part of B inside the dotted region might be omitted when calculating d(A,B). To avoid this problem, we use maximum between d(A,B), d(B,A)

define edge weight as the maximum value, as given in (1). Fig. 4 shows why our weight is reasonable.

In the calculation of d(A,B), nearest points of all A pixels can be found in the part on B outside of the dotted circle. So, the shape difference inside the dotted region is omitted for the calculation of d(A,B). Thus, d(A,B) could be low even if the shapes of two edges look totally different. However, d(B,A) is still high because the cost from the points inside the dotted circle is still used for average cost calculation. Therefore, because we take their maxima value, we can be consider both way and we can sure that two edges are similar when $W_{E_{pq}}$ is low.

C. Similarity Cost for the Closed Loop

Since the edge weight is defined as the average cost, we have to use a weighted summation to obtain the cost of connected edges. We define the new weighted summation $\bigoplus$ as follows.

$$W_{E_{a-b}} = W_{E_a} \bigoplus W_{E_b} = \frac{l(\tau(\overline{T_a}))}{l(\tau(\overline{T_a})) + l(\tau(\overline{T_b}))} \cdot W_{E_a} + \frac{l(\tau(\overline{T_b}))}{l(\tau(\overline{T_a})) + l(\tau(\overline{T_b}))} \cdot W_{E_b} \quad (3)$$

where E_{a-b} denote the connected edge of E_a,and E_b. T_a, T_b are the corresponding template subparts of each E_a, E_b. $l(\cdot)$ is the length of the template subpart. The corresponding template subpart of E_{a-b} is exactly same as the connection of T_a and T_b because we defined the correspondence using two end nodes of edges. Therefore,

$$l(\overline{T_{a-b}}) = l(\overline{T_a}) + l(\overline{T_b}). \quad (4)$$

The new definition for weighted summation is the key using the dynamic programming technique for optimization.

Finally, the similarity cost of a closed loop $L \subset E$ is defined using the new summation as follows.

$$\bigoplus_{e \in L} W_e = ((W_{e_1} \bigoplus W_{e_2}) \bigoplus W_{e_2}) \ldots . \quad (5)$$

And the most similar loop L^* is defined as

$$L^* = \arg\min_{L} \bigoplus_{e \in L} W_e. \quad (6)$$

D. Optimization to find optimal loop L^*

We design the optimization algorithm to find L^* based on dynamic programming. Our optimization algorithm is performed by the iterative shortest path algorithm [9]. Our new summation (3) satisfies the optimal substructure property of the shortest path algorithms. The sub-paths of a shortest path are also the shortest paths because the weight for summation of that sub-path is statically determined by the two end nodes of the sub-paths. Therefore, we can use the optimal weight of a sub-path for calculating the cost of extended paths passing through this sub-path.

To use the shortest path algorithm, we generate a directed graph from an undirected graph $G(E, V)$ by taking the direction for each edge. First, we give a direction for the template, for example, clockwise. The direction of the edge is determined according to the direction of the corresponding template part. For example, if $T_a - T_b$ is the corresponding template part of $E_a - E_b$ and the direction of template is from T_a to T_b, then the direction of $E_a - E_b$ is also from E_a to E_a. However, if T_a and T_b are too close, that is, smaller than a threshold, then we take both directions to avoid an unexpected break int the closed path. Fig. 5 shows an example of how to give an edge direction. In this example, we give both directions, indicated by orange arrows, because this edge is too short to give a direction.

The shortest cycle passing through an edge E_{pq} which connects V_p to V_q can be found using a shortest path algorithm. In this case, we find the shortest path from V_q to V_p after erasing edge E_{pq}. Then, we can find the shortest cycle by adding E_{pq} to the end of the path. When we use a shortest path algorithm, we use $\bigoplus$ instead of conventional summation. Finally, our optimization algorithm can be performed by iteratively finding the shortest cycle for each edge as Algorithm 1.

As mentioned above, L^* is changed according to the translation τ because we define the edge weight using $\tau(\overline{T_{pq}})$. Therefore, the final object detection is done by comparing L^* of each τ. To reduce the calculation time spent on finding L^*

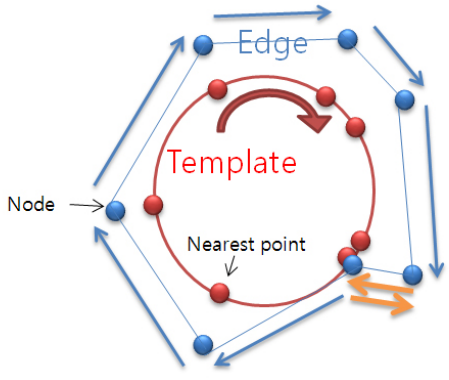


Fig. 5. Taking the edge direction for directed graph. The red line is a template and the blue one is a graph. Given the direction of template, we determine the direction of each edge according to the corresponding template direction. Notice the orange arrows that we take both direction.

Algorithm 1 Optimization Algorithm to Find L^*

for each edge E_{uv} **do**
  w = $W_{E_{uv}}$ (Save the current edge weight)
  $W_{E_{uv}}$ = ∞ (Erase the current edge)
  tempCost = the cost of shortest path from v to u (using Bellman-Ford shortest path algorithm with $\bigoplus$)
  if tempCost $\bigoplus$ w $<$ shortestCost **then**
    W_{L^*} = tempCost $\bigoplus$ w
    L^* = (shortest path from v to u) $\cup$ E_{uv}
  end if
  $W_{E_{uv}}$ = w (Recover the current edge)
end for

for each τ, the graph is generated using the segments around $\tau(T)$ instead of using the full segments in our experiments.

IV. Experimental Result

We have tested our algorithm on several examples including long video sequences. The input images have a resolution of 320x240 pixels. In all of our experiments, the initial over-segmentation was computed using publicly available code for Multiscale Normalized Cuts [10]. (available at http://www.seas.upenn.edu/$\sim$timothee/software/ncut_multiscale/ncut_multiscale.html).

Fig. 6 shows the similarity cost map generated by the proposed algorithm for the example shown in Fig. 1. Compared with Fig. 1 (c), this result demonstrates that our algorithm is more discriminative than the previous method. Because the cost gap between the object and background clutter is relatively high, and our algorithm tolerates shape variation caused by occlusions or scale changes.

We also tested our algorithm for the long video sequences. In this video, the target object- a shoe - is located in the complex cluttered background and often occluded by the other object. Our video includes 350 image frames, and a shoe is occluded in 205 images. To demonstrate the tolerance for shape variation, we tested with different size of the templates which resizing ratios are from 0.8 to 1.2. Although we did not consider rotation in this experiment, it can be covered by rotating the template in every position for some angles. To find the optimal matching position, we translated the template for each grid point with a 20-pixel interval.

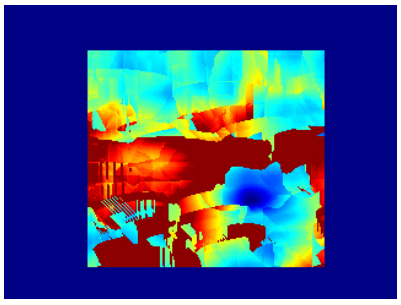


Fig. 6. A cost map generated by the proposed algorithm for the example in Fig. 1(b). We gave hight cost(dark red area) for the position which doesn't have any possible loop.

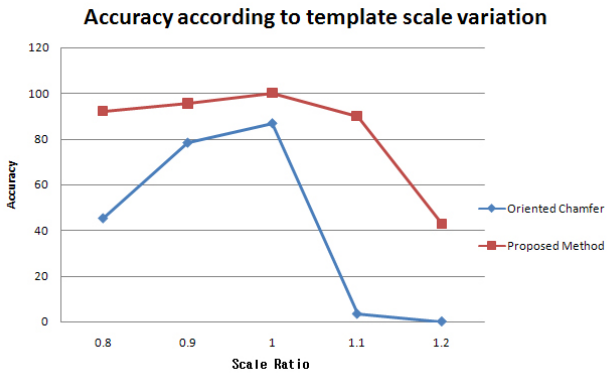


Fig. 7. Comparison of accuracy of the proposed algorithm and accuracy of the oriented chamfer matching according to the scale variation of the template

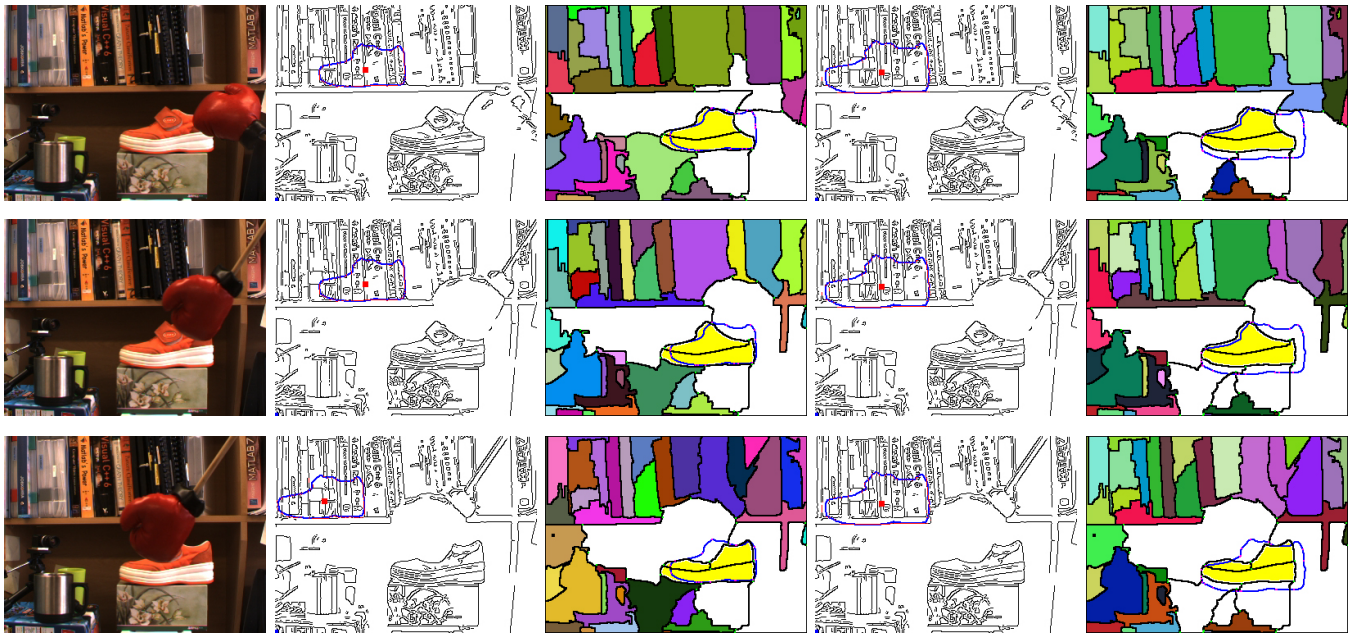
The result of our algorithm is shown in Fig. 8 (c) and (e). The blue lines are the optimal template positions, which have the lowest cost. At this template position, the graph is generated from the white regions. And the yellow regions are the regions contained by the optimal closed loops which has lowest cost. To evaluate the accuracy, we counted the correctly detected objects among all the frames. We evaluated that the detection is correct if the detected regions overlap over more than half of the true object region. Fig. 7 shows the graph comparing the result between the proposed algorithm and the oriented Chamfer matching. In the graph, the x-axis represents the scale variance of the template and y-axis represents the accuracy. The result demonstrates that our algorithm is more robust against scale variances of the shape.

We also tested our algorithm using a hand-drawn template. Fig. 9 shows the results using a same template. While oriented Chamfer matching failed to detect the objects because of shape difference between the hand-drawn template and the objects, our algorithm successfully detected the objects.

Finally, we applied the proposed algorithm to our grasping system. Fig. 10 shows the detection results at both the left and right images from the stereo camera. The red dots inside the yellow regions indicate the calculated center of mass. From these center positions, the object position in terms of the world coordinate was calculated using triangulation method with the camera calibration result. Some images of our grasping system demonstration video using the shape matching result of Fig. 10 are shown in the Fig. 11.

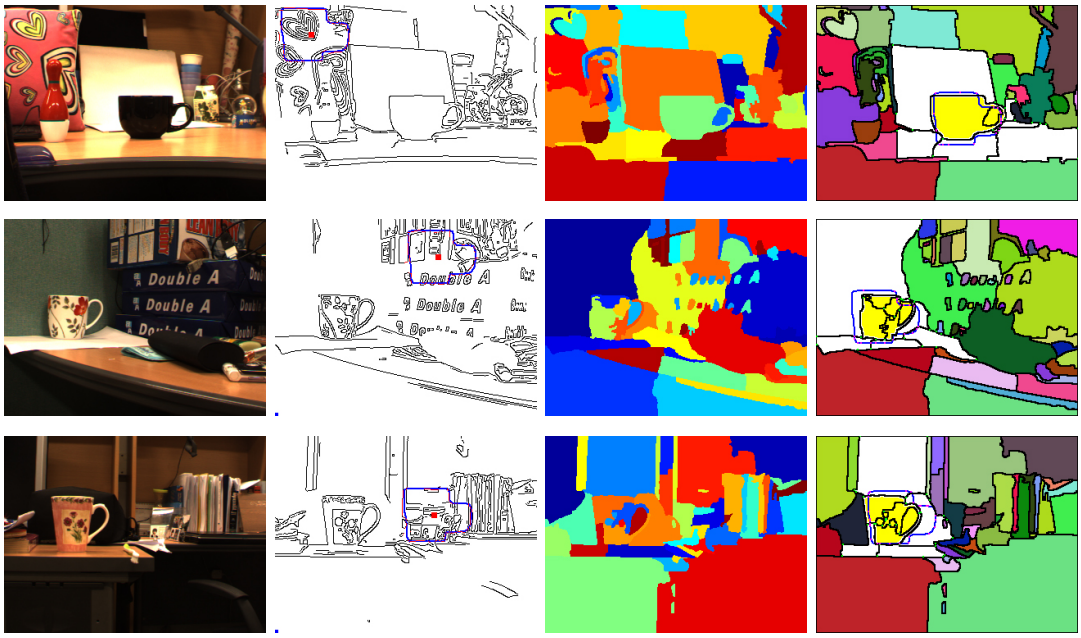
V. Conclusion and Future Works

We have presented a new approach for shape-based object detection. Unlike the previous methods, our approach uses the connectivity of edge pixels by generating a graph. The related edge pixels, which belong to an edge, are considered together and their similarity cost is defined by means of an explicit comparison with the corresponding template part. Conceptually, our approach is equivalent to finding the optimal combination of the segments that form the object using shape-based measurement. We solved this problem



(a) (b) (c) (d) (e)

Fig. 8. Experimental result according to occlusions and template scale variation. (a) Input images (b) oriented Chamfer matching (template scale ratio is 0.9) (c) proposed algorithm (template scale ratio is 0.9) (d) oriented Chamfer matching (template scale ratio is 1.1) (e) proposed algorithm (template scale ratio is 1.1).

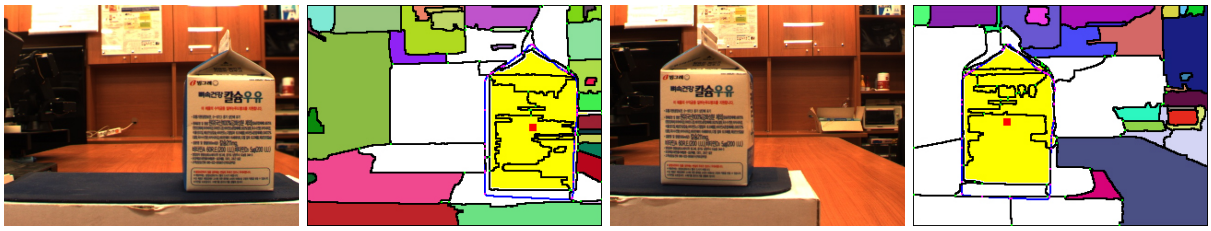


(a) (b) (c) (d)

Fig. 9. Experimental result for several objects with same template (a) input image (b) oriented Chamfer matching (c) over-segmentation result (d) proposed algorithm

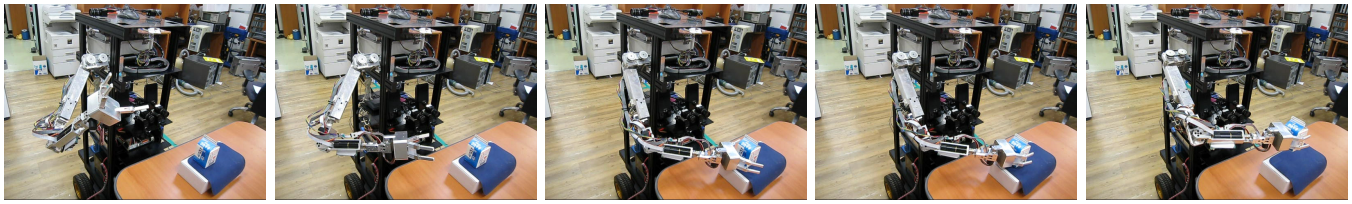
using a graph-based approach. This approach provides the key advantage of reducing ambiguity even in the presence of complex background clutter and occlusions, which is hard to solve in pixel level. The optimization is performed by a graph-based dynamic algorithm based on our new weighted summation. Our approach yields better results for the complex and occluded scenes. In our experiments, we demonstrated the robustness of our algorithm by testing with the several examples.

In this work, we only concentrated on the single-template-based approach. However, this work can be expanded to multiple-template-based approach to cover variations in shape.



(a) (b) (c) (d)

Fig. 10. Shape matching result for stereo camera images of our robot grasping system. (a,b) the input image of the left camera and the result. (c,b) the input image of the right camera and the result.



(a) (b) (c) (d) (e)

Fig. 11. The Robot grasping system using proposed shape matching

VI. ACKNOWLEDGMENTS

This work was supported by the IT R&D program of MKE/IITA [2008-F-030-01, Development of Full 3D Reconstruction Technology for Broadcasting Communication Fusion] and Ministry of Knowledge Economy, Korea [Monocular vision based natural feature extraction for cognitive model].

REFERENCES

[1] A. Opelt, A. Pinz, and A. Zisserman, "A boundary fragment model for object detection," *ECCV*, vol. 2, pp. 575–588, 2006.

[2] J. Shotton, A. Blake, and R. Cipolla, "Contour-based learning for object detection," *ICCV*, 2005.

[3] A. Thayananthan, B. Stenger, P. Torr, and R. Cipolla, "Shape context and chamfer matching in cluttered scenes," *CVPR*, pp. 127–134, 2003.

[4] D. M. Gavrila, "Pedestrian detection from a moving vehicle," *6th European Conf. on Computer Vision*, vol. 2, pp. 37–49, 6 2000.

[5] T. Cour and J. Shi, "Recognizing objects by piecing together the segmentation puzzle," *CVPR*, 2008.

[6] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf, "Parametric correspondence and chamfer matching:two new techniques for image matching," *5th Int. Joint Conf. Artificial Intelligence*, pp. 659–663, 1977.

[7] G. Borgefors, "Hierarchical chamfer matching: a parametric edge matching algorithm," *IEEE trans. Pattern Analysis Machine Intelligence*, vol. 10, no. 6, pp. 849–865, 1988.

[8] S.-Y. Ko, H. Joo, D. G. Choi, H. Kim, and I.-S. Kweon, "Intelligent grasping by learning from observation," *The 5th International Conference on Ubiquitous Robots and Ambient Intelligence*, 2008.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, "Introduction to algorithms," *The MIT Press*, 1990.

[10] T. Cour, F. Benezit, and J. Shi, "Spectral segmentation with multiscale graph decomposition," *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.