3D Object Retrieval and Pose Estimation for a Single-view Query Image in a Mobile Environment

Yoon-Sik Tak
Department of Electrical Engineering
Korea University
Seoul, Korea
Life993@korea.ac.kr

Eenjun Hwang
Department of Electrical Engineering
Korea University
Seoul, Korea
ehwang04@korea.ac.kr

***Abstract*—3D object retrieval and its pose estimation for a single view query image are essential operations in many specialized applications. With the recent deployment of various mobile devices, such operations require near real-time performance. However, most of the existing methods are not appropriate for mobile devices, due to their massive resource requirements. In this paper, we propose new 3D object retrieval and pose estimation schemes that can be used on a client-server platform. In order to accomplish this, we first construct both a sparse and a full index on the shape feature of the objects for the client and the server, respectively. Then, the client (the mobile device) retrieves the candidate camera view images that are similar to the query image by using the sparse index. The server refines the results by using the full index and then computes the exact pose by using the SIFT (Scale Invariant Feature Transform) features. In the experiment, we show that our prototype system based on the proposed scheme can achieve an excellent performance.**

Keywords- 3D object retrieval, pose estimation, shape-based retrieval, distance curve, SIFT.

I. INTRODUCTION

3D object retrieval and pose estimation are popular operations in various applications, such as robotic vision, medical image analysis, unmanned aerial vehicles (UAVs), and manufacturing automation. For instance, such operations can be used by robots to recognize diverse objects and change them to some specific pose for further actions such as assembly [1].

Depending on the hardware requirements, existing studies on the problem can be classified into three groups: The first group uses specialized equipment, such as the CMU high speed VLSI range sensor found in [2] and the CCD camera and laser scanner found in [3]. The second group uses multiple cameras. For example, in [4], the pose estimation was done using a pair of ground and onboard cameras for an autonomous unmanned aerial vehicle. In [5], a linear algorithm for computing the 3D points and the camera positions from multiple perspective views was proposed. The third group uses a single camera. For instance, in [6], a fully automatic solution using the Contracting Curve Density algorithm with speedup factors and SIFT features was proposed for a 3D object pose tracking. In [7], a pose tracking scheme based on the SIFT features and the Ferns was proposed for the classification of objects on mobile phones. The SIFT and Ferns were simplified for mobile phones at the cost of accuracy, due to their resource requirements. In our previous works [8][9], we introduced a simple object type classification scheme based on the shape symmetry and presented a time-consuming but accurate client-server collaboration scheme for 3D object retrieval in a mobile environment.

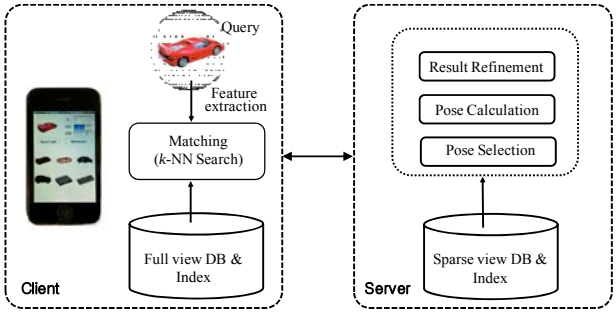


Figure 1. An illustration of system flow

In this study, we ameliorate our previous work in the following directions: (i) We present a new object type classification scheme, which can determine the type of an object automatically using the shape difference curve. ii) We present another client-server collaboration scheme which takes an heuristic approach to determine the object pose faster. We compare the performance of those two collaboration schemes through extensive experiments

In order to achieve good object recognition, we construct two indexes with different granularities based on the shape of the objects: The sparse index is constructed for the client in order to perform an approximate matching using large angle view images. Therefore its size is small compared to a full index. Conversely, the full index is constructed for the server using the small angle view images; these can be used for a more accurate matching. We propose two different client-server collaboration schemes in order to achieve load balancing. Basically, the client performs an approximate matching using the sparse index. The server refines it by using the full index. Since different view images of an object could give the same shape, we use their SIFT features for an accurate pose estimation. Figure 1 shows the overall architecture of our scheme.

 ISBN: 978-1-61208-129-8

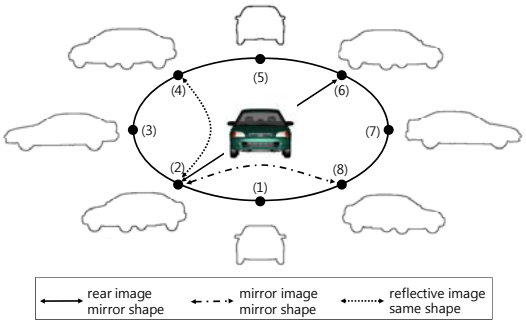


Figure 2. An illustration of shape patterns

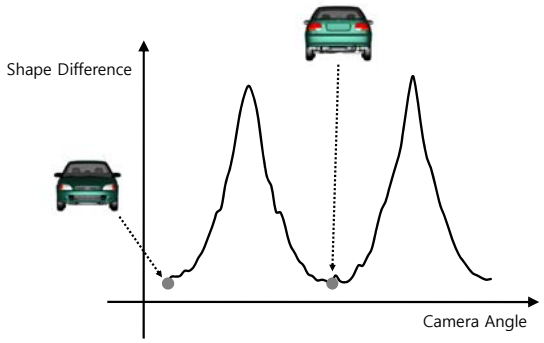


Figure 3. The shape difference curve of a car object

II. THE VIEW INDEX ORGANIZATION

A. The Shape Represenstation

One straightforward method for shape based 3D object recognition is to consider all of the distinct camera views of the objects. For each camera view image, we first extract its shape contour and then calculate its distance curve by connecting the distances between the center point and all of the points along the shape contour. Considering the distance curve as sequence data, we can use well-known sequence matching techniques for retrieval purposes. In addition, we can construct a multidimensional index based on their DFT (Discrete Fourier Transform) values.

B. Camera View Skimming

Most real-world objects have bisymmetry in the front and/or on the side. Depending on the object symmetry, different camera view images of an object can have a same or a mirrored shape. Formally, for an image at an arbitrary angle, we can define three related images according to the camera viewpoint, as seen in Figure 2: the rear image, the mirror image, and the reflective image. Based on these images, the following properties are exhibited depending on the object symmetry: (i) for a bisymmetrical object, the mirror image has the mirror shape of the object; (ii) for any object, the rear image also has the mirror shape of the object; and (iii) the reflective image has the same shape of the given image. Based on these facts, we can remove the redundant camera view images that have the same or mirrored shapes from the index without sacrificing any matching accuracy. This facilitates the reduction of the index size and improves the matching speed. More specifically, our camera view skimming scheme consists of two parts: i) Mirror image pairing, and ii) Camera view pruning.

Mirror image pairing: For any type of object, an image and its rear image have mirrored shapes. Distance curves of mirrored shapes are simply the reverse of each other, and their DFT values are the same. Therefore, we can pair these mirror shaped view images via a set of DFT values. By pairing the mirror-shaped views during indexing and restoring the reversed curve during matching, the index size can be reduced by 50%.

Camera view pruning: Since all of the viewpoints of 3D objects can be generated by a combination of horizontal and vertical camera movements, we can consider the object symmetry in two planes: the horizontal and the vertical. Depending on the front and side symmetries, we can define four object types per plane. For instance, the horizon plane has four object types: H1- H4:

H1: Represents objects that have the same shape with respect to all horizontal camera views (e.g., a sphere).
H2: Represents objects that have front symmetry. Their distance curves repeat every 90 degrees (e.g., cars).
H3: Represents objects that have front symmetry and the same front and side views. Their distance curve repeats more frequently than that of H2. (e.g., dice).
H4: Represents objects that have no repeating shape pattern. We can define the vertical object types in the same manner.

We define vertical object types in the same manner, except that we consider the front and top views of the objects in the vertical plane. By combining the horizontal and vertical types, we can define 16 different object types.

Object type classification: Depending on the object type, the index entries for the redundant camera view images can be removed from the index without sacrificing any matching accuracy. Even though most real-world objects have certain level of symmetry, it is not easy to determine the exact object type automatically. In order to perform this efficiently, we developed a new object type classification method based on the shape difference curve. Informally speaking, the shape difference curve indicates how similar each camera view image is to the base view image. The shape difference of two different view objects can be defined by the Euclidean distance of their distance curves. For any symmetric object, its base view represents the camera view where the object is exactly bisymmetric. Based on the base view image, we can calculate its shape difference curve using all the camera view images. Depending on the object type, its shape difference curve has different repeating patterns. By analyzing these repeating patterns, we can determine the object type. For instance, Figure 3 shows the shape difference curve of a sample car object. Since the car has bisymmetry, the repeating pattern appears twice in the curve. The detailed

 ISBN: 978-1-61208-129-8

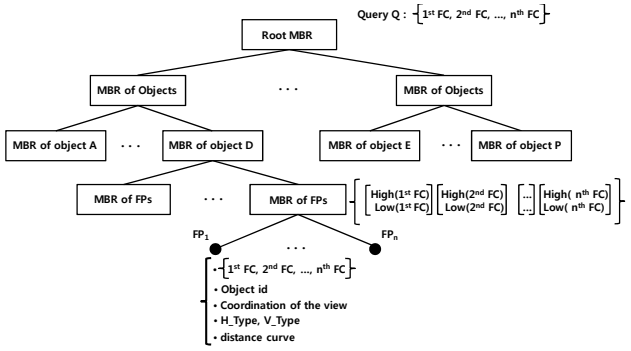


Figure 4. The overall index structure

steps for automatic object type classification are found in Algorithm 1. By considering both the horizontal & the vertical view sequences, we can automatically define object types.

Algorithm 1: *Object Classification (image sequence IS)*

```
Variable queue : Queue;
Period P = {0, 0} // {a, b} denotes view image period from a° to b°

1.  find a base view BV from the images in IS
2.  if there is no BV, exit the algorithm // Type 4.
3.  else
4.    calculate shape distance distances of BV image for IS
5.    insert every image I whose shape distance is less than
      some threshold σ into the queue
6.    repeat until queue is empty do
7.      retrieve one from the queue into next
        set EndP as ⌜next.position/2⌝
        initialize P to {0, EndP} and D to 0
8.      for i is 2 to (180/ EndP)
9.        for j is 0 to EndP
10.         if i is odd
11.           D += shape distance of I_j and I_(i-1)* EndP+j
12.         else
13.           D += shape distance of I_j and I_i*EndP-j
14.     if D ≤ σ
15.       return P
```

C. The Index Construction

Even though the number of camera view images that need to be indexed is considerably reduced via our camera view skimming method, we still have to consider a large number of view images of the 3D objects. Therefore, an effective index structure is essential to achieve fast searching. For this purpose, we have constructed a R-tree based indexing structure based on the set of DFT coefficients obtained from the distance curves. The detailed steps for index construction are: (1) For the distance curve of each view image, we calculate a set of Fourier Coefficients (FCs) and define its Fourier Point (FP) which includes the *object id*, the *coordinate*, the *H_type*, the *V_type*, and the distance curve of the origin view. (2) For each object, we construct a subtree which contains all of the camera views of the object done by grouping the adjacent FPs using a minimum bounding rectangle (MBR). For each MBR, the lower and upper endpoints of the FCs are defined by its lower and upper bounds. (3) All of the MBRs representing the subtree in (2) are grouped again into larger MBRs until all of the MBRs of the objects are contained in the Root MBR. Figure 4 illustrates the overall structure of our index structure.

III. MATCHING

In this section, we describe how the server and client work together to retrieve similar 3D objects and estimate their exact poses using a single view image. We consider two different collaboration schemes: CS1 and CS2. The former guarantees no false dismissal by considering all of the camera view images at the server. The latter uses some heuristics to speed up the matching process at the cost of accuracy.

A. The K-NN Search in the Mobile Client

Since we represent the shape of the objects by using the distance curve, any of the existing matching frameworks proposed for the sequence data can be used. In this paper, we revise the priority queue based k-NN search algorithm [10] to find the k most similar to the objects based on our index structure. In order to prevent an unnecessary and time-consuming matching process, we hierarchically used several low bound functions, such as *Fourier_Dist*, *MINDIST* and *LB_Keogh* [10]. The revised k-NN search algorithm is described in Algorithm 2.

Algorithm 2: *k-NN Search (Q, k)*

```
Variable queue : MinPriorityQueue;
1.  queue.push(root);
2.  result = {};
3.  while not queue.IsEmpty() do
4.    top = queue.Pop();
5.    if top.id is in the result
6.      continue;
7.    else
8.      if top is a sequence with DTW Dist.
9.        add top to result;
10.       if | result | = k
11.         return result;
12.     else if top is a leaf node
13.       for each Fourier Point P in top do
14.         queue.push(P, Fourier_Dist(Q,P));
15.     else if top is a Fourier Point P
16.       retrieve its full sequence S from DB;
17.       queue.push(S, LB_Keogh(Q,S));
18.       calculate reverse sequence S' of S //mirror shape
19.       queue.push(S', LB_Keogh(Q,S'));
20.     else if top is a sequence S with LB_Keogh Dist
21.       queue.push(S, DTW(Q, S));
22.     else
23.       for each child node C in top
24.         queue.push(C, MINDIST(Q,C));
```

 ISBN: 978-1-61208-129-8

B. Collaboration Scheme I (CS1)

After retrieving *k* similar objects for the given query Q, the client sends the result R to the server for refinement. The server refines it by using its full view index. In order to speed up the refinement, we use the maximum distance of R as an upper bound for the search. In order to guarantee no false dismissals, the CS1 considers all of the views of the objects during the refinement. Figure 5 shows the flow of the CS1; a brief sketch for the CS1 is shown in Algorithm 3.

Algorithm 3: *CS1 (Q, R)*

```
Variable queue : MinPriorityQueue;
1. retrieve MBR of Object Os whose distance from Q ≤
   maximum distance of R.
2. for each O
3.     if O.id is in R
4.         set the UB of O as the dist. in R.
5.     else
6.         set the UB of O as maximum distance in R.
7.     insert O into the minimum priority queue.
8. k-NN Search_CS1(Q, k)
```

The k-NN Search_CS1(Q, k) at line 6 is a modified version of Algorithm 2. The difference is that we don't need to push the root of the index (line 1) into the queue and the following code segments need to be inserted before every push operation.

```
1. if top.UB ≤ D(Q, top)
2.     continue;
3. set top to one of the nodes {P, S, S', C}
4. set D() one of the distances {Fourier_Dist(),LB_Keogh(),
   DTW(), MINDIST()}
```

C. Collaboration Scheme II (CS2)

In some applications, a quick response time could be more important than the guarantee of no false dismissal. Moreover, with the huge database of 3D objects, supporting a fast search can be prioritized at the cost of accuracy. CS2 speeds up the matching process by using a heuristic algorithm at the cost of matching accuracy. This scheme is based on the assumption that if, for two camera views V1 and V2 of an object and query Q, if *angle*(V1, Q) < *angle*(V2, Q), then *dist* (V1, Q) < *dist*(V2, Q). Even though there could be some exceptions, this assumption is still valid in most cases. Based on this assumption, CS2 can refine R very quickly. Figure 5 shows the CS2 flow; the major steps are shown in Algorithm 4. Before adding the top into the result in line 5 in Algorithm 2, Algorithm 4 is called in order to refine the results of the mobile device.

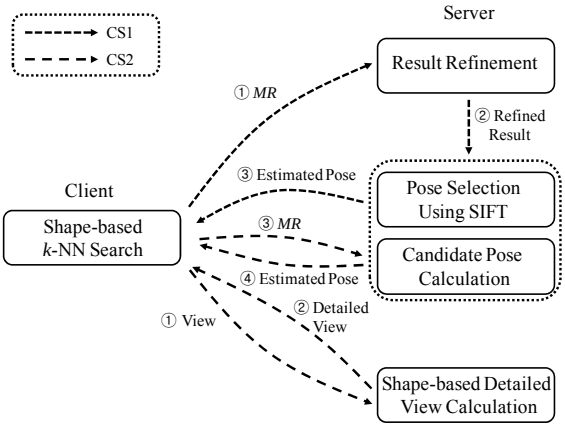


Figure 5. An overall flow for collaboration schemes

Algorithm 4: *CS2 (Q, image I, client view gap Gc)*

```
Variable queue : MinPriorityQueue;
1. set x as x coordination of I, y as y coordination, p as Gc.
2. extract 8 neighbor views N of I with the combination of
   V_{x±p, y±p}
3. insert every N into the queue with LB_Keogh dist. and p
4. repeat until |Result| = k do
5.   if top contains DTW distance
6.     if server view gap Gs ≤ top.p, set top.p to top.p /2.
7.       extract neighbor views N of top.p
8.       calculate LB_Keogh(N, Q)
9.       insert N into the queue with the distance and p.
10.    else
11.      insert top into Result
12.  else
13.    queue.push(top, DTW(Q, top));
```

D. The Candidate Pose Extraction

Since different views of an object might give the same shapes, we have to consider all of the same shaped views to give an accurate pose estimation. However, we have removed the redundant view images with the same shape obtained during the index construction. Therefore, at the pose estimation stage, we need to retrieve these images from the database or generate them dynamically from the 3D object using software tools such as CAD. Equations (1) and (2) explain the way to calculate the coordination of candidate pose views when the coordination of the base view is (*k*, *l*). HR_{period} and VR_{period} denote the period of horizontal and vertical shape pattern, respectively. For instance, HR_{period} of typical cars is 90. By combining all of the x and y coordination, we can get all of the candidate poses.

$$x = \begin{cases} HR_{period} * (i-1) + k & \text{if } i \text{ is odd} \\ HR_{period} * i - k & \text{if } i \text{ is even} \end{cases} \quad (1)$$

$$y = \begin{cases} VR_{period} * (j-1) + l & \text{if } i \text{ is odd} \\ VR_{period} * j - l & \text{if } i \text{ is even} \end{cases} \quad (2)$$

 ISBN: 978-1-61208-129-8

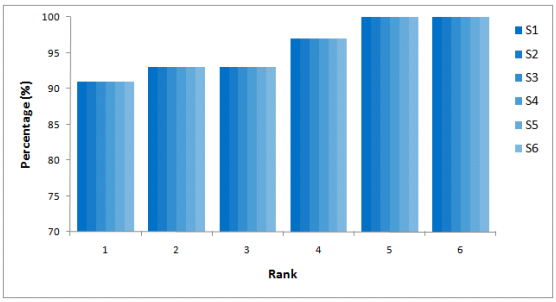


Figure 6. An accuracy comparison

where, $1 \le i \le 180/HR_{period}$ and $1 \le j \le 180/VR_{period}$

E. *The Pose Selection using SIFT Features*

Among the candidate poses, the best matching pose can be determined based on the actual visual features. This can be done by using a well-known image matching method such as SIFT [11] or SURF[12].

SURF is known to take relatively shorter time in matching than SIFT. On the other hand, SIFT shows better accuracy than SURF. In this work, we just need to consider a small number of images for pose estimation. Hence, we use the standard SIFT [11] method for matching for better accuracy even though it will take slightly longer time than the SURF method.

IV. THE EXPERIMENTS

A. *The Systems and Datasets*

In order to evaluate the performance of our proposed scheme, we implemented a prototype system. The server was equipped with an Intel Core2Duo CPU with 4 GB of RAM. iPhone 3Gs was used as the mobile client. Most of the applications at the client and server were implemented using C#. For the dataset in the experiments, we generated 259,200 views from 200 objects collected via the Internet [8]. The dataset contains diverse types of objects such as vehicles, kitchen appliances and furniture, to name a few.

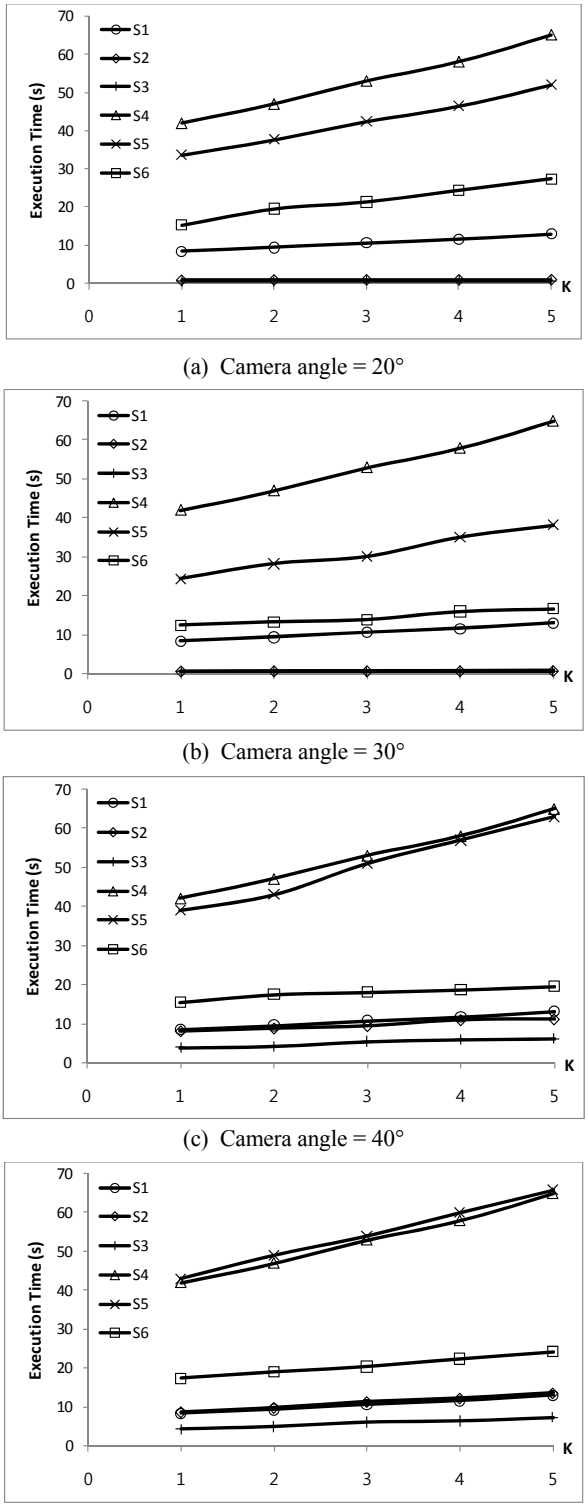
For the comparison, we considered six different system configurations that depended on the platform and the use of view skimming, as shown in Table 1.

Table 1 The System Configuration

Type	Description
S1	Server alone with camera view skimming
S2	CS1 with camera view skimming
S3	CS2 with camera view skimming
S4	Server alone without camera view skimming
S5	CS1 without camera view skimming
S6	CS2 without camera view skimming

B. *The Accuracy Comparison*

In this experiment, we show that our camera view skimming scheme does not impair the retrieval accuracy under any platform. The query input was a randomly selected view image stored in a database as a 3D model. Figure 6 shows the cumulative match curves (CMC) of the six different system configurations. For the test, we



(a) Camera angle = 20°

(b) Camera angle = 30°

(c) Camera angle = 40°

(d) Camera angle = 50°

Figure 7. The effect of camera angle on execution time

constructed indexes for 10° view images at the server and 30° view images at the client. From the graph, we observe the following facts:

1) For any platform, the camera view skimming does not have any effect on the accuracy.
2) Regardless of the view skimming, the three different platforms show the same accuracy. Theoretically, the server-alone and the CS1 guarantees the same accuracy. However, unlike CS1, CS2 cannot guarantee the same accuracy because CS2 refines the results using a heuristic approach. Therefore, CS2 shows a lower accuracy than CS1.

C. The Execution Time Comparison

In this experiment, we compare the total execution time, which includes the approximate estimation at the client and the result refinement at the server. In order to see the effect of the camera view gap size on the execution time, we considered four different camera angles for the client ranging from 20° to 50°, inclusively. In any case, the server used 10° of the camera view gap for the index construction. Since our scheme basically searches for similar objects based on the K -NN search, we measured the execution time by varying the size of the K as 1 to 5. Figure 7 shows the results. From the figure, we can observe the following facts:

1) Our camera view skimming scheme dramatically reduced the execution time.
2) A wider camera angle for view images with CS1 at the client helped to reduce the execution time since the wider camera view angle results in a smaller index at the client. However, an excessive camera angle gap can increase the execution time due to the overhead at the server for the refinement of the client result.
3) CS2 could reduce the execution time compared to CS1. From the experiment, we observe that setting the view extraction gap at the mobile client at 30° achieves a minimal searching time.

V. CONCLUSION

In this paper, we proposed a new shape-based client-server collaboration scheme for 3D object retrieval and pose estimation in a mobile environment. In particular, we proposed a camera view skimming scheme that reduces the index size and improves the search time using the bisymmetric property of most objects. For the pose estimation, we used the SIFT method to compare the same-shaped view images. Via various experiments on the prototype system, we demonstrated the effectiveness of our scheme. In addition, we proposed two collaboration schemes and compared their performance. Conclusively, larger camera angles used for the index at the client can reduce the index size and improve the search time. However, excessive camera angles might increase the search time at the server.

ACKNOWLEDGEMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2010-0025395)

REFERENCES

[1] A. Collet and S.S Srinivasa, "Efficient multi-view object recognition and full pose estimation," IEEE Conf. on Robotics and Automation, pp.2050-2055, 2010.

[2] D.A. Simon, M. Hebert and T Kanade, "Real-time 3-D Pose Estimation Using a High-Speed Range Sensor," IEEE Conf. on Robotics and Automation, pp.2235-2241, 1994.

[3] L. Haoxiang, W. Ying and C.W. de Silva, "Mobile Robot Localization and Object Pose Estimation Using Optical Encoder, Vision and Laser Sensors," IEEE Conf. on Automation and Logistics, pp.617-622, 2008.

[4] E. Altug and C. Taylor, "Vision-based pose estimation and control of a model helicopter," IEEE Conf. on Mechatronics, pp.316 - 321, 2004.

[5] C. Chen, D. Schonfeld and M. Mohamed, "Robust Pose Estimation Based on Sylverster's Equation: Single and Multiple Collaborative Cameras," IEEE Conf. on Acoustics, Speech and Signal Processing, 1085-1088, 2008.

[6] G. Panin and A. Knoll, "Fully Automatic Real-Time 3D Object Tracking using Active Contour and Appearance Models," Journal of Multimedia, vol. 1 no. 7, 2006.

[7] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond and D. Schmalstieg, "Pose Tracking from Natural Features on Mobile Phones," IEEE/ACM Symposium on Mixed and Augmented Reality, pp.125-134, 2008.

[8] H. Kim, Y. Tak and E. Hwang, "Shape-based indexing scheme for camera view invariant 3-D object retrieval," Multimedia Tools and Applications, Vol. 47, No. 1, pp.7-29, 2010.

[9] Y. Tak and E. Hwang, "Indexing and Matching Scheme for Recognizing 3D Objects from Single 2D Image," Internet and Multimedia Systems and Applications, 2009.

[10] E. Keogh and C. Ratanamahatana, "Exact indexing of dynamic time warping," Knowledge and Information Systems, Vol.7, pp. 358-386, 2005.

[11] D.G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," International Journal of Computer Vision, Vol. 60, no. 2, pp. 91 - 110, 2004.

[12] H. Bay, A. Ess, T. Tuytelaars, L. V. Gool, "SURF: Speeded Up Robust Features", Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346- 359, 2008.

 ISBN: 978-1-61208-129-8