GET-based map icon Identification for Interaction with Map and Kiosks

Huiqiong Chen, Derek Reilly
Faculty of Computer Science, Dalhousie University, Halifax, Canada
Paper ID ****< replace **** here and in header with paperID>

Abstract

This paper presents a GET (Generic Edge Token) based approach of detecting and recognizing objects by their shapes, and applies it to improve our ongoing work that considers ways of interacting with paper maps using a handheld. In our work, the GET-based technique aims to help user better locate points of interest on map by recognizing these icons from images/videos captured by handheld camera. In this method, video/image content can be described using a set of perceptual shape features called GETs. Perceptible object can be extracted from a GET map, and then be compared against pre-defined icon models based on GET shape features in recognition. This method provides a simple, efficient way to locate points of interest on the map, determining handheld location, orientation when combined with RFID (senor-based) technique. The tests show that the GET-based object identification can be executed in reasonable time for the real-time interaction system. Meanwhile, the detections and recognitions are robust under different lighting conditions, camera focus, camera rotation, and distance from the map.

Keywords: *Generic Edge Token, object recognition, ubiquitous interaction*

1. Introduction

Visio-based techniques allow tracking and recognizing visual features from video sequences. These techniques can be widely used in many applications [1] [2]. In this paper we present a GET (Generic Edge Token) based approach of detecting and recognizing objects in video/image by their shapes, and apply it to our ongoing work that considers ways of interaction with paper maps and kiosks using the handheld. In our work, the integration of GET-based object identification and Radio Frequency Identification (RFID) sensor-based techniques permits direct interaction with the visual features of the resource using handheld.

The key idea in the "interaction with map and kiosks" research is to link the virtual and real worlds by hovering over, pointing at, or gesturing toward physical maps or kiosks with a handheld device. The handheld device acts as a window on the virtual world, allowing the user to send queries and gather feedbacks. For example, the user can select an icon on a map by centering it in the visual field of a handheld camera [3]. The system will detect and recognize an icon based on its GET shape features, determine the handheld location, orientation to the map by using the combination of video recognition results and RFID data, and then send back information associated with the icon to user. In the icon detection, a perceptible object can be extracted from a GET map by grouping the approximate GETs into object contour closures. The detected object is then compared against pre-defined icon models based on GET shape features for icon recognition.

Most object identification techniques require detect object first from a video/image and then recognize the object by its features. The features used commonly in identification include color, edge, optical flow and texture, etc. The most common object detection techniques either segment objects from a single frame/image (e.g., region segmentation) or employ frame difference information during detection (e.g., background subtraction,). Some object detection methods are reviewed in [4][5]. An extensive survey for object detection, representation and tracking is presented in [6]. Comparing to other object identification techniques, our GET-based method uses the unique perceptually stable edge features to describe objects and represent video/image content. It provides a simple and efficient way to identify the icons in real-time under different conditions such as lighting, camera distance, rotation and shooting angle.

The rest of this paper is organized as follows. Section 1 introduces backgrounds and motivation of the technique first and then gives system architecture. Section 2 presents the GET-based object detection. Section 3 provides object feature descriptors that can describe the detected objects and pre-defined icon models based on their shape features. Section 4 discusses the details of the icon recognition. In Section 5, experimental results and analysis are provided. Finally, Section 6 discusses conclusions and suggests possible future works.

1.1. Motivation

The GET-based technique presented here extends the previous interaction prototype by increasing selection precision and permitting lenslike interaction with the underlying resource. In previous work, Reilly et al. have explored the use of sensor-based techniques in the interaction. RFID tags are embedded within maps as position indicators [5]. Tags have been placed in a regular grid, or beneath large landmarks. However, this approach has some limitations. First, the accuracy of interaction is limited due to the resolution of RFID (normally larger than 20mm in diameter). So the granularity afforded by an RFID-only solution is limited. If there are multiple points of interest in the range of one tag, a list of resources linked to that tag will be returned to user instead of the specific point of interest that user wants. For instance, a user moves the handheld over the area associated with a RFID tag D3 to locate the bank icon in that area, where D3 represents the coordinate of the tag ID. Fig. 1(a) shows the relevant visual feedback: the grid area on the paper map associated to this tag. However, what interests the user in this area is the bank, which the user can center to the handheld camera and capture directly through the camera, as Fig. 1(b) shows. Therefore more, more precise queries might be achieved if a point of interest captured from camera can be detected and recognized by the system.

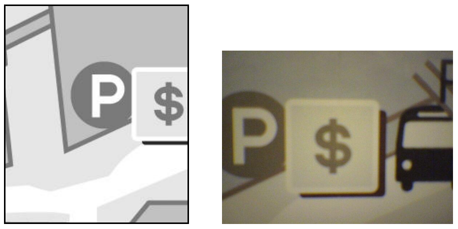


Fig. 1: (a) the visual feedback from RFID tag D3 (b) A point of interest captured by the camera

Second, user can not select the icons across the RFID boundaries if there is only a simple mapping between each RFID tag and an information resource. For example, the bus icon in Fig. 2(a) lies on the boundaries of four individual RFID tags in Fig. 2(b). In this case, the user enquiry can not be processed on any individual tag.

At last, it is challenging to generate virtual overlays in a continuous fashion on a handheld screen since RFID reads provide no direct indication of angle or rotation of the handheld. We need to decide the handheld orientation by using the visual features of the captured icons. As shown in Fig. 2(c), camera orientation can be determined if the estimation of icon rotation angle is available.

The limitations above can be solved if the visual information can be incorporated with the location information provided by RFID tag. The hybrid method will make the retrieve more feasible and accurate in the prototype. Edge features (GETs) are used instead of color in the icon identification, as the color of icons will vary under different lighting conditions.

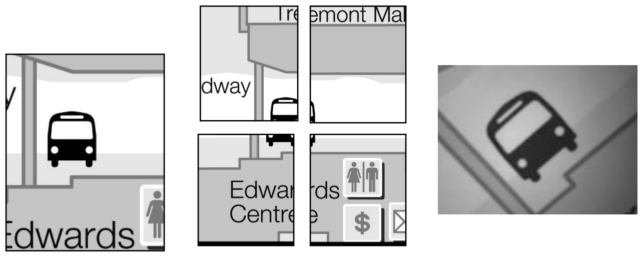


Fig. 2: (a) a bus icon on the map; (b) the map grids for RFID tags associating with that icon; (c) icon captured by handheld camera with rotation

1.2. Generic Edge Tokens

Generic Edge Tokens (GETs) are perceptually significant image primitives which represent classes of qualitatively equivalent structure elements. A set of GETs includes Generic Segments (GS) and curve partition points (CPP). A GS is perceptually distinguishable edge segment with linear or nonlinear features while A CPP is a typical junction of GSs. Gao and Wang proposed a fast curve partition method which performs partition following the similar objective in human visual perception. The partition is performed based on monotonic discontinuation of both direction information and geometry data. Detail explanation of GET detection can be found in [8]. Fig. 3 (a) shows curve partition examples.

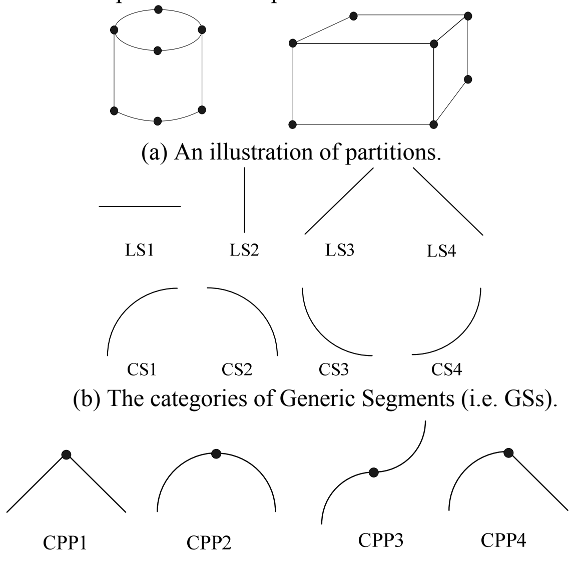

(a) An illustration of partitions.
LS1
LS2
LS3
LS4
CS1
CS2
CS3
CS4
(b) The categories of Generic Segments (i.e. GSs).
CPP1
CPP2
CPP3
CPP4
(c) The typical types of CPPs.

Fig. 3. Curve partition examples

Each GS has its own unique descriptive characteristics and represents a general class of curve segments. GSs can be perceptually classified into eight categories as Fig. 3 (b) shows. A CPP is a perceptually significant point where adjacent GSs meet and curve turning takes place. The are 4 types of typical CPPs as illustrated in Fig. 3 (c). The formal definition of GET and CPP categories can be found in [9]. CPP can group GSs into perceptual structures. The intrinsic perceptual features of GETs will facilitate further detection of perceptual shapes formed by GETs and ease the icon extraction and recognition in this application. A GET map sample is shown in Fig. 4.

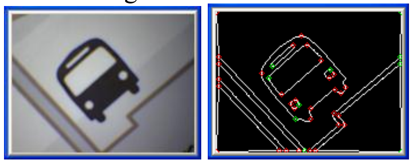


Fig. 4. The illustrations of GET map extraction

1.3. System Architecture

Fig. 5 gives the system architecture. The process involved three steps: (1) GET extraction by edge tracker; (2) icon/object detection by GET grouping and (3) icon recognition by its shape.

Once any image/video is captured by camera, video content can be described based on GETs. The edge tracker produces GET maps from individual video frames. Each segment in the map is classified into on of the eight categories in Fig. 3(b).

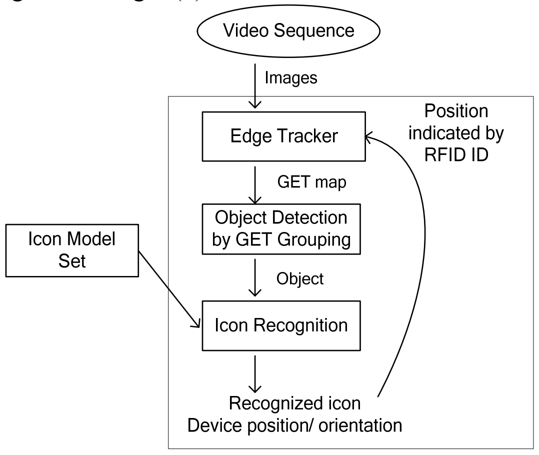


Fig. 5. The system architecture

In the stage of object detection, appropriate GETs in a GET map can be perceptually grouped into object represented by GET-based closures and their relationships. In order to be identified as an icon, a GET based object is compared against each element in a set of pre-defined icon models based on recognition rules. Each element in the model set represents one type of perceptible icons (and/or valuable features), and is described by a set of recognition rules. The recognition rules are generated by object feature descriptors according to the generic description of that icon. Once an icon is recognized, the mobile device's position and orientation relative to the map/icon are determined. This information is then used to update the feedback on the device.

Ideally, the RFID tag read by the mobile device can assist to locate which 'grid square' the device's camera is centered on so that the system can track the device position and restrict the focus of location to a certain Region in subsequent frames.

2. Object Detection

In the first step of GET map extraction, we use the edge tracker provided by Deep Vision Inc. which was developed based on the concept in Section 1.2 [8].

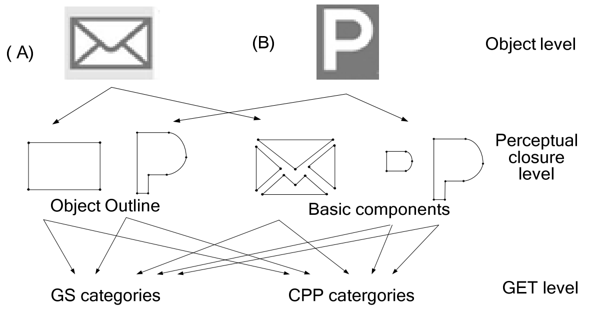


Fig. 6: object representation hierarchy

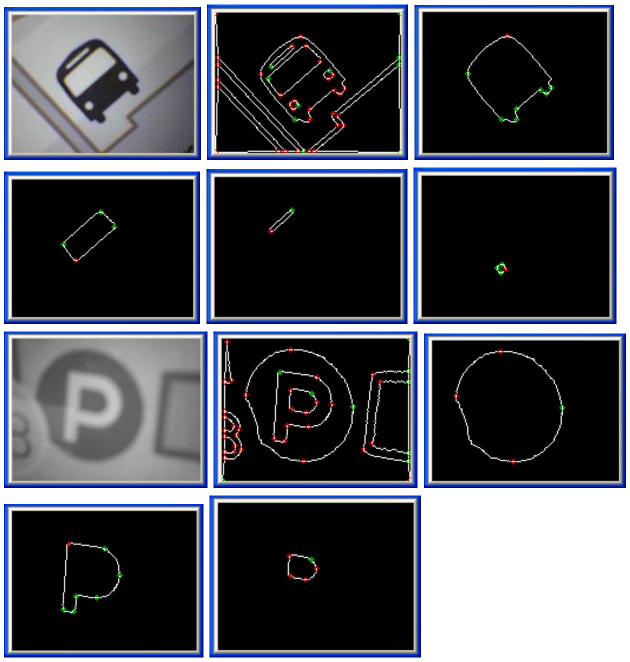


Fig. 7. Object extraction examples. For each example: camera image; GET Map; object outline; some basic components.

Perceptible object can be extracted from a GET map by grouping the GETs into approximate contour closures. We can apply the algorithm in [9] in this application to detect possible objects (i.e., map icons) from GET map. In this algorithm, both the outline of an object and its basic components are extracted so that a hierarchical model is set up to represent the object. Fig. 6 illustrates the hierarchy. Any object can be represented by the shape of its outline, the shapes of its basic components, each of which is a closure described by GETs, as well as the relationships between its outline and each basic component. Because of the descriptive nature of GET representation, object shapes can be estimated quickly based the contour GET types, which will support object recognition tasks in the next step. Fig. 7 gives some examples of object extraction.

3. Object Feature Descriptor

In order to identify icons, each object got from the former steps should be compared against all types of the pre-defined icon models using shape structure attributes. Each of the generic models, such as the polygon, rectangle, bank, parking, post offices, bus terminals, etc, can be described by a set of recognition rules. To set up proper recognition rules for each model, we define object feature descriptors, based on which a set of recognition rules can be generated for an icon model.

The potential features that can be used to identify any icon model mainly include (1) the shape of icon outline; (2) the shapes of all basic components of the icon; (3) icon color attributes, and (4) the relations between the each basic component and the object outline. Appropriate features should be chosen for the object description so that the recognition can be accurate and robust. In this application, the features selected for object recognition must satisfy the following criterions:

- invariant to object scaling and rotation (robust to camera rotation, and distance from the map).
- tolerant to different lighting conditions on map.
- tolerant to shape deformation caused by the camera shooting angle.

Considering these factors, we select two types of descriptors for our recognition: (1) GET-based shape descriptor, which can be used to describe the closure shapes, and (2) relationship descriptor that describes how the basic components build up an object/icon.

(1) GET-based shape descriptors

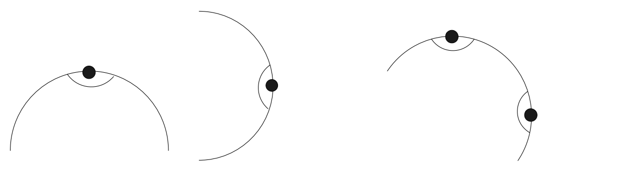
Shape descriptors can describe the shape of a GET-based closure. Normally it is used to describe an object outline and each basic component constituting this object. The features used in the description include:

- CPP; The CPPs depicts the connections between GSs. Not all CPPs have equivalent value for recognition. Only the non-smooth CPPs (see definition below) with critical perceptual features will be described in recognition rules. A non-smooth CPP can be described by its type, the internal angle and the edge length ratios between the two GSs connecting to it.

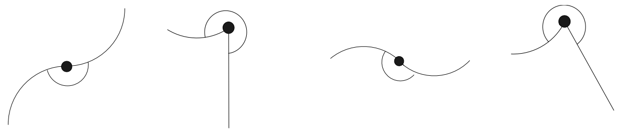
The CPPs are perceptually stable features. All types of CPPs are invariant to shape scaling and shape deformation. Various types of CPPs can be classified into 2 categories according to their rotation attributes:

Smooth CPPs: a smooth CPP connects two GSs which have continuously perceptual meaning. The direction of the two GSs turns smoothly in perceptual view as Fig. 8 (a) shows. Although smooth CPPs are not stable partition points when object rotation occurs, (see Fig. 8(a)), they are not critical factors in recognition, since the GSs they connect have similar perception. A smooth CPP is removed in the descriptor so that the curves it connects are integrated as one smooth curve.

Non-smooth CPPs: a non-smooth CPP connect GSs with different perceptual meaning as Fig. 8 (b) shows. They are invariant to rotation as well as scale so it is reliable CPPs for recognition.



(a) Left: two smooth CPPs; right: smooth CPP rotation illustration



(b) Left: two non-smooth CPP; right: non-smooth CPP rotation

Fig. 8: The rotation of CPPs.

- GS; each GS can be described by its type (line/curve) and curvature.

The calculation of the curvature is shown in Fig. 9. For a GS with endpoint v_x and v_y, let p_i be the middle point of GS, p_l be the middle point of the curve between v_x and p_i, p_r be the middle point of the curve between p_i and v_y. The curvature of this GS can be approximately characterized by a sequence e $[v_x, p_l, p_i, p_r, v_y]$. The curvature of this GS is estimated by the normalized curvature:

$$Cuv = |(\theta_1 + 2*\theta_2 + \theta_3 - 720)/3| \qquad (1)$$

Where θ_1 is the angle between $p_r v_y$ and $p_r p_i$; θ_2 is the angle between $p_i p_r$ and $p_i p_l$; θ_3 is the angle between $p_l p_r$ and $p_l v_x$.

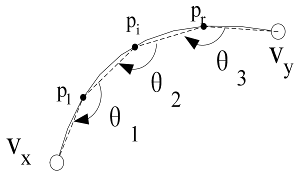


Fig. 9: Curvature evaluation of a GS

The normalized curvature for a curve arranges between (0, 180). The curvature of a straight line is 0. The larger the value of *Cuv* is, the more curving this GS is.

- Closure; it describes structure attributes belonging to the whole closure.

Each closure is described by N_{GS} (the number of GSs in the closure), N_{CPP} (the number of non-smooth CPPs), N_{LS} (number of straight lines), and N_{CS} (the number of curves), N_{Para} (the number of parallel pairs); each pair of parallels G_i and G_j in the closure is featured by the length ratio and the distance ratio between the G_i and G_j:

$$Ratio_r = L_l/L_s \qquad (2)$$
$$Ratio_d = L_d/L_s$$

where L_d is the distance between the parallel pairs, L_l is the length of the longer GS in G_i and G_j while L_s is the length of the shorter GS.

TABLE 1: Illustration of general shape description

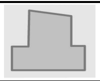
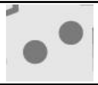
Shape	Shape Sample	Shape descriptors			
		CPP	GS	Closure	
				Parallel	other
Polygon		Y	Y		Y
Circle		Y	Y		Y
Triangle		Y	Y		Y
Rectangle		Y	Y	Y	Y
Trapezoid		Y	Y	Y	Y

Table 1 illustrates some shapes in the map and the shape features applicable to them.

(2) Relationship descriptor; the relationship descriptor describes how several basic components build up an object. It considers the number of singular closures (i.e., basic components) in an object, the relationships between each basic component and the object outline, including size ration and position relationship.

For each interior singular closure C_i in an object, the size ratio of C_i is defined as:

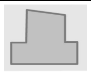
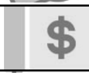
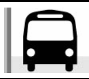
$$Ratio_s = S_i/S_o \qquad (3)$$

where S_i is the size of C_i, S_o is the size of the object outline. The position relationship between each basic component and the icon outline can be "inside" or "adjacent" by sharing with some common GSs.

Table 2 illustrates icon models and the descriptors that applicable to them.

For one specific icon model, a set of recognition rules can be generated by apply the shape and/or relationship descriptor to it. Let $M=\{M_1,\dots M_k\}$ be set of the icon models, and $S=\{S_1,\dots S_n\}$ be set of the shapes described by the shape descriptor, $\forall M_i \in M$ can be described by a tuple $<c, s, r>$, where $c \in S$ is the object outline shape, and $s = \{s_1,\dots s_m\}$ is a subset of S. Each element in s is one basic component of M_i. $r=\{r_1,\dots r_m\}$ is the set of relationships associating c and s where $\forall r_j=r(c, s_j)$ can be described by the relationship descriptor. Particularly, if M_i has only one basic component, $s=\{c\}$ and $r=\Phi$ where the relationship descriptor is not applicable to M_i.

TABLE 2: Illustration of object description

Object	Object sample	Shape descriptors				Relationship descriptor
		CPP	GS	Closure		
				Parallel	other	
building		Y	Y		Y	
Parking		Y	Y	Y	Y	Y
Bank		Y			Y	Y
Post Office		Y	Y	Y	Y	Y
Bus Stop		Y		Y	Y	Y

4. Object (icon) Recognition

Any unknown object extracted from the former steps can be matched to known icons/shapes by comparing it with each of the object shape models. The unknown object that satisfied the recognition rules of some specific model will be identified as an entity of that model.

For any detected closure, the shape descriptor will be applied to generate a number of features. The recognition process will measure the difference between these features and the recognition rules of a shape model as the difference between the unknown shape and the shape model. Therefore the shape distance between an unknown closure c and a shape model s can be estimated as follows:

$$d(c,s)=\sqrt{\sum_i w_i(f_i'-f_i)^2}$$

where f_i' and f_i are the values of the ith feature of c and s respectively, w_i is the weight of the ith feature of s.

If the distance d is small, the shape of c is more likely to be categorized into the type of s. Some range of distance tolerance is allowed in the recognition to make the match more robust.

The shape recognition procedure is given below:

Step 1: For each unknown closure C_i, perform step 2 and 3 for recognition.

Step 2: for each type of shape $S_j \in S$ described by the shape descriptor, rank the shape match between C_i and S_j by the value of $d(C_i, S_j)$.

Step 3: among all the match ranks, select the shape model with the highest rank as the shape of this closure.

For an unknown object constituted by multiple closures, both the shape descriptor and the relationship descriptor will be applied to generate shape features for each closure as well as the relationships among the shapes. The unknown object should be compared against each element in M.

The object recognition procedure is given below:

Step 1: For each unknown object O_i constituted by multiple shapes, let O_i be described by a tuple $\{c_i', s_i', r_i'\}$, perform step 2 and 3 for the recognition.

Step 2: for each object models $M_j=\{c_j, s_j, r_j\}\in M$, perform the shape recognition for each element s' in s_i' against s_j, and rank the shape matches by $d_s=\sum d(s', s_j)+d(c_i', c_j)$.

Step 3: perform relationship match between r_i' and r_j: $d_r=0$; for $\forall r' \in r_i'$ and $\forall r\in r_j$, if $r'=r$, $d_r=d_r+1$.

Step 4: Among all the object models, select the one with the highest rank of (d_s+d_r) as the object of this recognition.

Camera rotation can be estimated easily during recognition by calculating the angle between the matched pairs.

5. Experiments and Evaluation

Currently we are working on a prototype of Marked up map which links to a paper map of Halifax, Canada. GET-based recognition support was developed for a set of icons used in a commonly used map of Halifax. The icon detection and recognition steps were tested on four sets of images and videos. The first set contains images/videos taken in normal condition. The second set contains images/videos taken by the handheld's camera at different rotations and shooting angles relative to the map. The third set contains images/videos taken from various distances from the map. The fourth set contains images/videos taken under two different lighting conditions (low light and high reflection). The size of each image or video frame captured by handheld camera ranges between 160*120 and 176*144. The frame rate of the videos is 10 frame/s. The detection and recognition accuracy is estimated by precision and recall. For all testing images/videos, let N_{all} be total number of icons needed to be recognized; let N_{recog} be the total number of icons recognized by our approach; let $N_{correct}$ be the number of icons correctly extracted and recognized; Recall = $N_{correct}$ / N_{all}; Precision = $N_{correct}$ / N_{recog}. Experiment results are listed in Table 3.

The testing shows that our methods are robust under different lighting conditions, camera focus, camera rotation and shooting angle,, and distance from the map. The average execution time for the detection and recognition is around 0.04~0.08 sec in total for one video frame or image. Therefore, GET-based object detection and recognition can be executed in reasonable time for real-time system.

The reasons of false identification include: (1) several types of icons share similar shape properties in the map, (2) the icon contour cannot be detected completely from the image or (3) user moves handheld so fast that the video frames captured from the handheld camera are too blur to be processed. In practice, some icons may not be identified in one frame during video recognition but this is not a serious problem because the icon may occur over a period of time so one miss recognition will not cause the loss of icon information.

Table 3: Icon detection and recognition accuracy

Image Set	$N_{correct}$	N_{recog}	N_{all}	*recall*	*precision*
Normal	23	25	25	92%	92%
Rotation/shooting	19	19	20	95%	100%
Distance	13	14	15	86.7%	92.9%
Lighting	38	39	42	90.5%	97.2%

6. Conclusion and Future work

This paper presents an object detection and recognition technique that can be applied to our map interaction prototype to recognize icons from camera images/videos. In the proposed system, perceptual shape features (GETs) are used for describing video content. Perceptible object can be extracted from GET map, and then be compared against pre-defined icon models based on GET shape features for recognition.

The hybrid of RFID and GET-based object identification approach facilitates the use of mobile device for interaction with static maps and other similar visual resources by giving rotation, distance, and precise position. The experiment shows the robustness and efficiency of the recognition. We also apply this approach to other applications [10][11].

Currently only the edge based shape features is used in object description and recognition. In the future other basic features, such as color, may be integrated in the system to give more accurate icon description for recognition.

References

[1] C. A. Rahman, W. Badawy and A. Radmanesh, A Real Time Vehicle's license Plate Recognition System, in *Proc. IEEE* Conf. on Advanced Video and Signal Based Surveillance: 163- 166, 2003.

[2] H. Yan, Z.Y. Wang, S. Guo, A Method for 2D Bar Code Recognition by Using Rectangle Features to Allocate Vertexes, in Proc. Of the 6th International Workshop, GREC 2005: 99-108, 2005

[3] D. Reilly, H. Chen, Mobile lenses: a hybrid approach to direct interaction with maps and kiosks, Proc. of the Permid at Pervasive 2006, 2006.

[4] A. Narendra, On Detection and Structure Representation of Multiscale Low-Level Image, ACM Computmg Surveys, 27(3), pp. 304~306, 1995

[5] J. Freixenet, X. Munoz, D. Raba, J. Mart, and X. Cuf, Yet another Survey on Image Segmentation: Region and Boundary Information Integration, in Proc. of the 7th European Conference on Computer Vision: 408-422, 2002

[6] A. Yilmaz, O. Javed, M. Shah: Object tracking: A survey, ACM Computing Surveys, 34(4): 1-45, 2006

[7] D. Reilly, M. Rodgers, R. Argue, M. Nunes, et al., Marked-up Maps: Combining Paper Maps and Electronic Information Resources, Journal of Personal and Ubiquitous Computing, 10(4): 215-226, 2006

[8] Q. Gao, A. Wong, Curve detection based on perceptual organization, Pattern Recognition, 26(1): 1039-1046, 1993.

[9] H. Chen, Q. Gao, Efficient Image region and shape detection by perceptual contour grouping, Proc. the 2005 IEEE Int. Conf. on Mechatronics and Automation: 793-798, 2005.

[10] H. Chen, D. Rivait, Q. Gao, Real-Time License Plate Identification by Perceptual Shape Grouping and Tracking, Proc. of the 9th International IEEE Conf. on Intelligent Transportation Systems: 1352-1357, 2006.

[11] D. Reilly, H. Chen, Toward fluid, mobile and ubiquitous interaction with paper using recursive 2D barcodes, accepted by the Pervasive Mobile Interaction Devices workshop at Pervasive 2007, 2007