社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－CVIM－155 (29)
2006／9／9

# Study of Real-time Marker-based PC Detection for Visual Call Center Applications

Qi Zhang†, Jun Ohya†, Shunichi Yonemura‡

† *Graduate School of Global Information and Telecommunication Studies, Waseda University*

‡ *Nippon Telegraph and Telephone Corporation*

## ABSTRACT

***Nowadays, majority of PC producers provide their product support services via visual call centers. At times, users cannot describe their problems correctly which causes waste of time. We propose a new solution for PC based visual call center applications. The video images taken by the user's portable camera are used for smooth communications between the user and the operator at the visual call center. To protect the user's privacy, only the PC in the video images should be visible to the operator. This paper proposes a marker based method for detecting the PC in real-time. The design of the markers that are attached to the PC are discussed and studied. A modified Chamfer Matching Algorithm is used for detecting the markers. Experimental results demonstrate the effectiveness and efficiency of the proposed method.***

## 1. Introduction

Visual call centers are wildly used as a solution of PC troubleshooting. Such service is indeed useful for users, but the serious problem is that majority of novice users cannot describe their problems correctly when they are talking to the operators. To solve this problem, we propose a new solution that uses video data obtained by a handy camera at the user's hand. The operator can know what is happening by watching the video images sent from the camera. However, this solution brings us a new issue. If video information is used, operators are able to see not only the computers, but everything within the image including the user's privacy.

A method is proposed to protect users' privacy; more specifically, images sent to a call center are processed to make everything invisible except the PC. To achieve this objective, a method which can detect target objects is needed. In previous literatures, numerous object detection approaches have been proposed. However, most of those works only focus on the effectiveness, but not efficiency. Moreover, most experiments mentioned in those papers were carried out under controlled conditions, so they cannot be used practically. To conquer this difficulty, a marker-based region extraction method is discussed in this paper.

Marker recognition is frequently applied to various issues. But for our application, markers should be well-designed in order to achieve desirable computation efficiency. Special colors are used in our application so that color filters can be easily built. Here, a combined color filter which consists of three simple filters is used to make the processing faster.

Only special colors are not enough to recognize the markers. A special marker pattern which can be detected rapidly is painted on the marker. An image matching algorithm is needed as well to decide whether those colors belong to the markers or something else. In the

application, a modified version of Chamfer Matching Algorithm (CMA) [1] is used to detect the pattern of the markers. After the matching, the variance values of matched regions are calculated to eliminate the effect of noise.

The method introduced in this paper has been proved effective by experimental results. The algorithm is efficient and robust enough to be applied under practical conditions.

The rest of this paper is organized as follows. The entire processing flow of our application is shown in section 2. In section 3, we discuss the design of the markers. Section 4 describes the chamfer matching algorithm. Experimental results are presented in section 5, followed by conclusions in section 6.

## 2. Processing Flow

The supremely important target of our method is to achieve real-time image processing. In previous literatures, image matching has been proved to be a complicated process, which costs much more time than any other processing. To get rid of this problem, a coarse-to-fine processing is applied in our application. The image matching process does not need to be very accurate. Moreover, the matching only takes place in several small areas within an original image in order to obtain desirable effectiveness. Some other processes are applied to lessen the computation of the image matching. Fig. 1 shows the processing flow of our method.

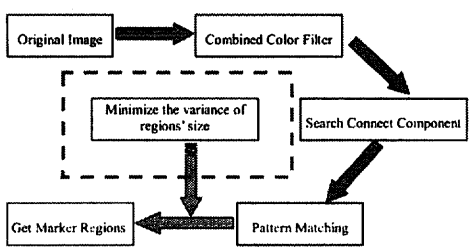


Fig. 1 Processing Flow

A filtering processing is first conducted when an original image is sent to the application. Here, the color filter is called "Combined Color Filter", because it contains two simple filters. There is a description about this combined filter in section 3. The system tries to find out all the areas whose color is similar to that of the markers; we call them the "candidate regions". Then, within each area, image matching is performed to detect the pattern of the markers. Some candidate regions in which the system cannot find the pattern are discarded. Notice, the marker pattern is not matched at a high accuracy. Since it costs a lot of time, other subsequent processes are used to improve the accuracy later. Usually, some non-marker regions are still left even after the matching. Since the size of every marker is the same, by minimizing the variance of the regions' size, the marker regions can be obtained.

## 3. Marker Design

### 3.1. Background Color

Well-designed markers can make our processing fast, accurate and smooth. The markers should be suitable for filtering to ensure the rapidity of the following computations.

Some rather bright colors are selected to be the background color of the markers. They are able to be detected easily because of the high brightness. In addition, people do not lay many things with bright colors around because such colors make their eyes uncomfortable. That means, commonly, the noise in the original images is not too much. In practice, the markers used in our application are painted with fluorescence pigment to make the color more distinctive. In our application, yellow is used as the background color. The RGB constitution of this color can be expressed as (255, 255, 0), in which there are two very strong color components (red and green) and a rather weak component (blue). A filter can be easily built to detect this color. In fact, the color of pink (255, 0, 255) and cyan (0, 255, 255) can also be used as background.

### 3.2. Marker Pattern

As for the pattern of our markers, a circle is drawn at the center of each marker. This pattern can be detected even if markers are rotated. Since the distance between user's camera and the computer is unknown, the circle should not be drawn by a thin line. Thin lines sometimes disappear in the images which are taken far away from the PC. Therefore, the pattern is designed to be a round area at the center of a marker. The area is painted with dark color to enhance the contrast between this area and the background, so that the circle pattern can be gotten by searching edge pixels within the image. The marker used in our application is shown in Fig. 2.

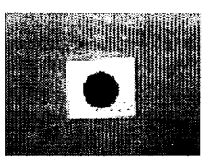


Fig. 2 The marker

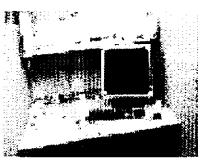
### 3.3. Color Filtering

Color filtering is used to decrease unnecessary computation of image matching. That means the more noise is filtered, the faster our algorithm becomes. Several filters are used together to improve the accuracy of color filtering. We call them "combined filter".

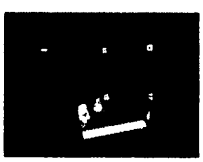
Totally, the combined filter is composed of three common filters. As we discussed before, the background color used in our application has a high brightness value. This peculiarity is firstly used for the filtering. Areas with high brightness values can easily be detected by using a threshold. Then, another filter which distinguishes RGB constitution is used. For our application, in which yellow is used as the background color, the second filter determines whether both the red and green components are much stronger than the blue component. After the second-time filtering, we get areas whose colors are similar to the background color. The result by the second filter is shown in Fig. 3 (b).

In fact, we can start the image matching immediately after the second-time filtering, but to make the algorithm more efficient, one more filter is used. The round area, which is defined as the marker pattern, is painted in a different color at the center of the marker. The third filter determines whether the color at the center of an area is similar to the center area color. In our application, the area in painted with black color and this filter determine whether the center of an area is dark enough.

After the combined filtering, we get several areas which resemble the markers. Those areas are called "candidate regions". The image matching described in Section 4 is performed for every candidate region. Fig. 3 (c) shows the candidate regions obtained after the combined filtering.



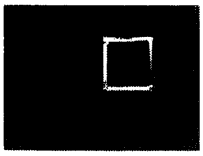
(a) Original image



(b) After the 2nd filter

(c) After the 3rd filter

(d) Extracted edge



(e) Result of 3-4DT (f) Final result

Fig. 3 Processing results

## 4. Chamfer Matching Algorithm

Matching methods can be loosely divided into three classes: algorithms that use the image pixel values directly, e.g., correlation methods; algorithms that use low-level features such as edges and corners; and algorithms that use high-level features such as identified objects, or relations between features, e.g., graph-theoretic methods.

Methods that use the image pixel values directly are very sensitive to any change between images, e.g., a modest shift in illumination may make matching between otherwise equal scenes impossible. The same structures in images from different sensors cannot be identified. High-level matching methods are very insensitive to these disturbances. The drawback is that high-level features must first be extracted and identified and that is, in most cases, a difficult matching problem in itself.

An image matching algorithm named "Chamfer Matching Algorithm" (CMA) [1] is used in our application to match the marker pattern. This method is carried out at an intermediate level. It matches edge points or other low-level feature points, extracted from the digital images.

Chamfer matching was first proposed in 1977 [1]. It is a technique for finding the best fit of edge points from two different images, by minimizing a generalized distance between them. The edge points of one image are transformed by a set of parametric transformation equations, which describe how the images can be geometrically distorted in relation to one another. The original version of the chamfer matching algorithm is useful only in a limited number of applications. It is a fine-matching method, which needs a good start hypothesis of the transformation that brings the edges into correspondence.

The original chamfer matching method has now been modified in several versions. The most famous one is called "Hierarchical Chamfer Matching Algorithm" (HCMA) [2]. Recently, the chamfer matching method is

used for human recognition by some researchers as well [3].

Since CMA is a time consuming method, a modified version is used in our application in order to achieve desirable computation efficiency. Templates of the marker pattern are automatically generated and only a part of feature points are used for matching.

### 4.1. Edge Extraction

An edge extraction method is needed since edge points are used as feature points in CMA. The research on edge extraction has a long history and various methods have been developed. Commonly, edge extraction uses a high pass filter (called operator), which is usually expressed as a small window, to scan the gray-level images. There are some rather complex methods as well, e.g. wavelet analysis etc. By using complicating methods, we can obtain very good results. However, those methods cost more time and since the marker pattern is only a circle, a simple operator is accurate enough.

In our application, Prewitt Operator is used for edge extraction. The result of the edge extraction processing is called predistance image. See Fig. 3 (d) for an example.

### 4.2. Distance Transformation

The process converting a binary image to an approximate distance image is called distance transformation (DT). In the predistance image, each non-edge pixel is given a value that is a measure of the distance to the nearest edge pixel. The edge pixels get the value zero. The true Euclidean distance is resource demanding to compute; therefore an approximation is used here. Good integer approximations of Euclidean distance can be computed by a process known as chamfer 3-4 distance (3-4DT) [4], [5].

In a 3 by 3 pixel neighborhood, the distance between horizontal/vertical neighbors is defined as 3, and the distance between diagonal neighbors is defined as 4. With these values the maximum difference from the Euclidean distance is about 8 percent while the difference becomes 13 percent when a more common "2-3DT" is used.

In the binary edge image, each pixel is first set to zero and each non-edge pixel is set to infinity. If the DT is computed by parallel propagation of local distances, the processing can be expressed as follows:

$$v_{i,j}^{k} = \min(v_{i-1,j-1}^{k-1} + 4, v_{i-1,j}^{k-1} + 3, v_{i-1,j+1}^{k-1} + 4, v_{i,j-1}^{k-1} + 3,$$
$$v_{i,j}^{k-1}, v_{i,j+1}^{k-1} + 3, v_{i+1,j-1}^{k-1} + 4,$$
$$v_{i+1,j}^{k-1} + 3, v_{i+1,j+1}^{k-1} + 4)$$

where $v_{i,j}^{k}$ is the value of the pixel in position (i, j) at iteration k. The iteration continues until no value changes. The number of iterations is proportional to the longest distance presented in the image. Fig. 4 is a simple example of 3-4DT. In fact, the 3-4DT can also be calculated sequentially, see [2]. The image processed by distance transformation is called distance image. See Fig. 3(c) for an example.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

(a) Original image

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 13 | 10 | 7 | 4 | 3 | 4 | 7 | 8 | 11 |
| 12 | 9 | 6 | 3 | 0 | 3 | 4 | 7 | 10 |
| 11 | 8 | 7 | 4 | 3 | 0 | 3 | 6 | 9 |
| 8 | 7 | 4 | 3 | 0 | 3 | 4 | 7 | 10 |
| 7 | 4 | 3 | 0 | 3 | 4 | 7 | 8 | 11 |
| 6 | 3 | 0 | 3 | 4 | 6 | 7 | 8 | 11 |
| 6 | 3 | 0 | 3 | 4 | 3 | 4 | 7 | 10 |
| 7 | 4 | 3 | 0 | 3 | 0 | 3 | 6 | 9 |
| 8 | 7 | 4 | 3 | 0 | 3 | 4 | 7 | 10 |
| 11 | 8 | 7 | 4 | 3 | 4 | 7 | 8 | 11 |

(b) 3-4DT result

Fig. 4 3-4DT

### 4.3. Matching Measure

The template composed of feature points is superimposed on the distance image. An average of the pixel values that all the feature points hit is the measure of correspondence between the edges, called the edge distance (See Fig. 5). A perfect fit between the two edges will result in edge distance zero, as each feature point in the template will then hit an edge pixel. The actual matching is to minimize the edge distance. To make this minimization as simple as possible, the edge distance function should be smooth and convex enough. There are many variants of matching measure averages, e.g. arithmetic, root mean square (abbreviated r.m.s.) and median. The root mean

square average is used in our application for matching measure (see [6] for the reason):

$$\frac{1}{3}\sqrt{\frac{1}{n}\sum_{i=1}^{n} v_i^2}$$

where $v_i$ are the distance values and n the number of points in the template. The average is divided by three to compensate the unit distance three in the 3-4DT. In fact, different measures can be used together as well (see [7]).

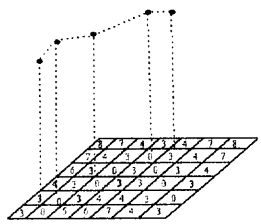


Fig. 5 Edge distance computation

### 4.4. Start Position and Template Generation

In the original version of CMA, the start position where the matching starts has to be decided first. When the matching starts, the template is transformed by equations in order to deal with the translation, rotation, scale etc. For our case, since we have already had several candidate regions after the color filtering, the start positions are set to be the center of each candidate region.

In stead of template transformation, template generation is used in our application. Since the pattern is a circle, it is not necessary to deal with the rotation of template. Moreover, circle can be generated easily. By generating circles of different size, the marker pattern can be found from images taken from different distances.

To make the algorithm faster, when the templates are generated, the size of them is changed in every 2 or 3 pixels. In practice, not all the feature points but only a part of them on the circle edge are generated. The number of feature points depends on the size of the circle. For example, when the radius of circle is 2 pixels, 9 feature points are enough for the matching (at this time, there are totally 12 points on the edge); but if the radius is 4 pixels, more feature points should be generated. It has been proved by the experiment that this way is accurate enough for our solution.

### 4.5. Noise Elimination

It is probable to result in false matching in non-marker regions even though we search the marker pattern in every candidate region. At times, some objects whose pattern is rather similar to the markers appear in the original image. To identify the markers from all the matched regions, the variance of region size is calculated. Marker regions can be found out by minimizing the variance because each marker has the same size.

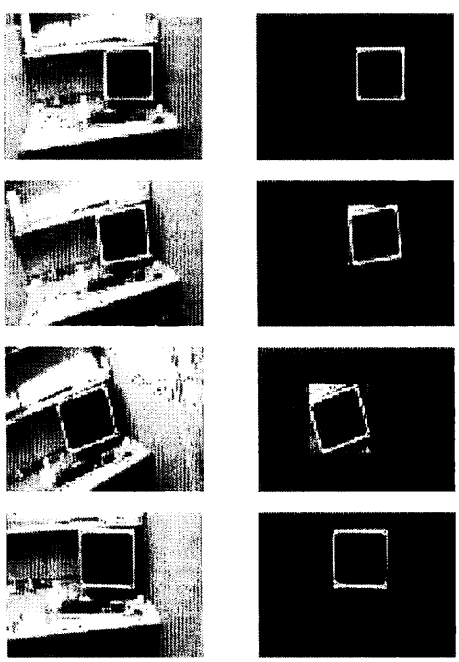
## 5. Experimental Results

Experiments were carried out to testify the efficiency and effectiveness of our algorithm. Video images whose size is 320 by 240 pixels are used to conduct the experiment. Note that the algorithm gets slower when the noise increases. But all the computation for each frame can be done within 32 milliseconds. For our case, it is fast enough.

As for the accuracy, it remains above 80 percent, even if there is a lot of noise within the image. More experimental results are shown in Fig. 6. The original images are shown on the left side and the results are on the right side.

## 6. Conclusion

This paper has proposed a new solution for PC troubleshooting by visual call centers, where the protection of the users' privacy is emphasized. Markers are used to help the recognition of computers. Filters and other processes are used to lessen the computation of image matching. The chamfer matching algorithm is modified to improve the computation efficiency. The efficiency and effectiveness are shown by the experimental results.



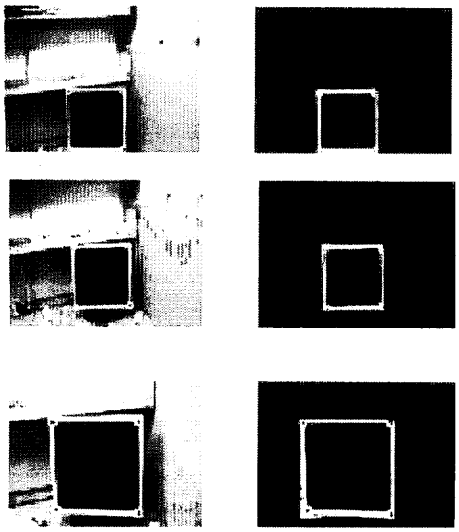


Fig. 6 Experimental results

## References:


[1] H.G. Barrow, J.M. Tenenbaum, R.C. Bolles and H.C. Wolf, "Parametric correspondence and chamfer matching: Two new techniques for image matching", in Proc 5th Int. Joint Conf. Artificial Intelligence, Cambridge, MA 1977, pp. 659-663

[2]Gunnila Borgefores, "Hierarchical chamfer matching: A parametric edge matching algorithm", IEEE transaction on Pattern Analysis and Machine Intelligence, Vol. 10, no. 6, pp. 849-856, 1988

[3] Liang Zhao, "Closely Coupled Object Detection and Segmentation", ICCV, pp. 454-461, Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, 2005.

[4] ----, "Distance transformations in arbitrary dimensions," Comput. Vision, Graphics, Image Processing, vol. 27, pp. 321-345, 1984.

[5] ----, "Distance transformations in digital images," Comput. Vision, Graphics, Image Processing, vol. 34, pp. 344-371, 1986.

[6] G. Borgefors, "An improved version of chamfer matching algorithm," in 7th Int. Conf. Pattern Recognition, Montreal, P.Q., Canada, 1984, pp. 1175-1177.

[7] Abdul Ghafoor, "Robust Image Matching Algorithm", EC-VIP-MC 2003, 4th EURASIP Conference focused on Video / Image Processing and Multimedia Communications, 2-5 July 2003, Zagreb, Croatia