2-D Curved Shape Recognition Using a Local Curve Descriptor and Projective Refinement

Kyoung Sig ROH and In So KWEON, Nonmembers

SUMMARY In this paper, we propose a descriptor as a shape signature and the projective refinement as a verification method for recognizing 2D curved objects with occlusions from their partial views. For an extracted curve segment, we compute a series of the geometric invariance of equally spaced five co-planar points on the curve. Thus the resulting descriptor is invariant only under rotation, translation, and scale, but sufficient similarity is preserved even under large distortions. It is more stable and robust since it does not need derivatives. We use this transformation-invariant descriptor to index a hash table. We show the efficiency of the method through experiments using seriously distorted images of 2-D curved objects with occlusions.

Key words: occlusion, curve segment descriptor, invariant, geometric hashing, projective refinement

1. Introduction

Feature indexing approaches for object recognition are attractive when the number of object is very large. Especially, invariance-based techniques use invariant features of objects to structure the model-base. When features, that remain invariant independent of viewing positions, are detected on an image, these features are used to compute indexing functions and in turn to reduce the search space. The importance of indexing approaches in object recognition has been emphasized and demonstrated in many papers [3], [8]–[10].

Recently, various invariant descriptors have been proposed for recognizing planar shapes[1], [2]. Among these, only a few descriptors are applicable for recognizing 2D curved objects with occlusions under affine or projective transformation. A descriptor must be similar under the change of viewing directions and must discriminate curve segments with different shapes. It must also possess a local property to recognize occluding shapes. Traditionally, smooth curves can be described using differential invariances with their local properties. But these descriptors require high order derivatives, which are very sensitive to pixel errors due to noise. Therefore, it is known as a numerically impractical tool.

For solving this problem, Weiss[7] provided differential invariants that only require taking the fourth derivative of a curve, and suggested methods for robustly finding the derivative of a curve in spite of image error. Van Gool et al.[4],[5] and Brill et al.[6] presented semi-differential invariant descriptors which require reference points to reduce the order of derivative. Van Gool’s descriptor is used to recognize more than forty curved objects. But the total time spent on object recognition was relatively high.

As another approach, Zisserman et al.[3] proposed an invariant descriptor using a canonical frame construction that relies on the measurement of bitangency and cast tangency points; four such points are measured for each canonical frame. Then, they also proposed some indexing functions for recognizing planar shapes with concavity.

In this paper, we propose a descriptor as a shape signature and the projective refinement as a verification method for recognizing 2D curved objects with occlusions from their partial views. The descriptor is similar to the canonical frame method in using five points invariant, but it is different in constructing the descriptor and in indexing for object recognition. For an extracted curve segment, we compute a series of the geometric invariance of equally spaced five co-planar points on the curve. Thus the resulting descriptor is invariant only under rotation, translation, and scale, but sufficient similarity is preserved even under large distortions. It is more stable and robust since it does not need derivatives and is less sensitive to the accuracy of reference points. Also it can be applied for a general shape including convex curve segments. We use this transformation-invariant descriptor to index a hash table.

We use a projective refinement method as a verification algorithm, which was proposed in [16]. The method repeatedly computes a projective transformation between model and scene contours using hypotheses generated in the previous stage. The resulting pixel error after prediction refinement is used to select a true hypothesis. An error model for five-point invariants is introduced to define a similarity between models and scene descriptors, to determine a searching range in a hashing table, and to determine weights in the iterative projective refinement.

This paper is organized as follows. In Sect.2, the
plane projective transformation is derived, the geometric invariance for five coplanar points is reviewed, and an error model for the invariant is proposed. In Sect. 3, a method of constructing a descriptor is presented, and the similarity and discrimination is tested. In Sect. 4, we explain how to construct a hash table with indexing, and how to generate hypotheses, and a projective refinement method is presented. In Sect. 5, we present experimental results using images of forty curved objects.

2. Descriptor and Similarity Function

In this section, we define a local transformation-invariant descriptor to recognize the occluded object, what is an extension of the closed contour descriptor [16]. Also a similarity measure between two descriptors is defined.

2.1 Five-Points Plane Invariant

Given five points on an object plane and the corresponding five points on the image plane, the five-points invariant is defined by:

\[ I = \frac{\det(X_1^2 X_2^2 X_3^2 X_4^2) \det(X_2^2 X_3^2 X_5^2)}{\det(X_2^2 X_3^2 X_4^2) \det(X_3^2 X_4^2 X_5^2)} \]
\[ = \frac{\det(X_1^2 X_2^2 X_3^2) \det(X_2^2 X_3^2 X_4^2) \det(X_3^2 X_4^2 X_5^2)}{\det(X_2^2 X_3^2 X_4^2) \det(X_3^2 X_4^2 X_5^2)} \]

(1)

where \( X_1^2 \) and \( X_2^2 \) represent the \( i \)-th image point at view 1 and 2, respectively.

Assuming a Gaussian noise model for pixel positioning error, the variance of the five-points invariant becomes

\[ E[(\tilde{I} - I)^2] = \sigma^2 \sum_{i=1}^{5} \left[ \left( \frac{\partial \tilde{I}}{\partial x_i} \right)^2 + \left( \frac{\partial \tilde{I}}{\partial y_i} \right)^2 \right]. \]

(2)

where \((\tilde{x}_i, \tilde{y}_i)\) represents the noisy observation of a true image point \((x_i, y_i)\), and \(\tilde{I}\) represents the noisy invariant propagated from pixel noise.

Then, the threshold for the similarity for a given invariant \( I \) is defined as:

\[ \Delta I = 3 \times \sqrt{E[(\tilde{I} - I)^2]} \]

(3)

2.2 The Local Contour Descriptor

In Fig. 1(a), a segment of a planar shape with \( L \) pixels is shown as a white curve. We divide the contour segment into the five sub-segments with equal length. For example, the shaded circles in white in Fig. 1(a) show such a case. Using the five-point invariant, a descriptor can be constructed along the curve segment as follows:

\[ I(k) = \frac{\det(X_2^2 X_3^2 X_4^2) \det(X_2^2 X_3^2 X_5^2) \det(X_3^2 X_4^2 X_5^2)}{\det(X_2^2 X_3^2 X_4^2) \det(X_2^2 X_3^2 X_5^2) \det(X_3^2 X_4^2 X_5^2)}, \]

\[ k = \text{START} \sim \text{END} \]

(4)

where

\[ X_1^2(k) = (X_1^2(a), Y_1^2(a), 1), a = k \]
\[ X_2^2(k) = (X_2^2(b), Y_2^2(b), 1), b = L/5 + k, \]
\[ \text{if } b > L, b = b - L \]
\[ X_3^2(k) = (X_3^2(c), Y_3^2(c), 1), c = 2L/5 + k, \]
\[ \text{if } c > L, c = c - L \]
\[ X_4^2(k) = (X_4^2(d), Y_4^2(d), 1), d = 3L/5 + k, \]
\[ \text{if } d > L, d = d - L \]
\[ X_5^2(k) = (X_5^2(e), Y_5^2(e), 1), e = 4L/5 + k, \]
\[ \text{if } e > L, e = e - L \]

Here, START and END represent the start and end point of the segment, respectively.

Figure 1(b) shows a resulting local descriptor, in which \( I(l) \) and \( I(k) \) are invariants computed from \((X_1(l), X_2(l), X_3(l), X_4(l), X_5(l))\) and \((X_1(k), X_2(k), X_3(k), X_4(k), X_5(k))\), respectively.

We define a similarity measure between the model and scene invariants, \( I_m \) and \( I_l \), as:

\[ \text{similarity} (I_m, I_l) = \frac{\sum_{k=1}^{L} T(k)}{L} \]

(5)

where \( L \) denotes the number of invariants, and

\[ T(k) = \begin{cases} 1 & |I_m(k) - I_l(k)| < \Delta I, \\ 0 & \text{otherwise} \end{cases} \]

(6)

and \( \Delta I \) is a preset threshold.

2.3 Similarity and Discrimination of the Descriptor

Figure 2 shows images of the same object taken from

Fig. 1 A segment of a 2-D object and the computed descriptor.

(a) A segment of an input image
(b) Corresponding Descriptor

Fig. 2 Selected segments at each view.
different view positions. Numbered segment indicates the corresponding curve segment in each image. Figures 3 (a), (b), (c) show the proposed descriptors for segments at each view and (d) show the descriptors of each segment for image 1. Table 1 shows the similarity measures between the descriptors at each view with a minimum similarity 0.8. Table 2 shows the similarity measure between the descriptors for two segments in image 1, with a maximum similarity 0.3.

3. Recognition Algorithm

3.1 Model-Base Construction by Hash Table

In this section, we present the structure of a hash table and an indexing method for it. For a segment $j$ of model $i$, we compute a boundary descriptor $MD_{ij}$ that has $L$ points on its curve segment. The $L$ invariants of the descriptor are used to form $L$ hash tables. Each bin of a hash table corresponds to a particular value of an invariant and is used for matching. $MD_{ij}$ can be represented by its compact form:

$$MD_{ij} = \{I_{ij}(1), I_{ij}(2), I_{ij}(3), \ldots, I_{ij}(L)\}$$

where each element is a five-points invariant and indicates a bin of a hash table.

Figure 4 shows an example of the model-base construction.

The time complexity of this stage is $O(M S_M L^2)$, where $M$ is the number of models, $L$ is the number of boundary data, and $S_M$ is the average number of segments for each model.
3.2 Recognition

3.2.1 Hypotheses Generation by Indexing and Grouping

For a curve segment \( k \) of an input image, we compute the descriptor \( SD_k \) and vote the segments of the model that are recorded in the address indicated by each element of the descriptor. Also we consider the searching bound, \( \Delta I \) for each element, which is defined in [16]. Thus, we vote all the segments of the model within the searching bound. Figure 5 shows an example of indexing and voting.

Hypotheses are generated if the value in the voting table is greater than a predetermined Threshold. Figure 6 shows an example of the generated hypothesis-table for all the input segments, where a shaded box denotes that there exists a generated hypothesis between a scene segment and the corresponding model segment.

Using this table, we group adjacent curve segments to form a longer segment, if the neighboring segments satisfy the condition on the similarity. For example, in Fig. 6, the segments 2 \( \sim \) 4 of the model 1 are combined because all the segments correctly correspond to the segments 1 \( \sim \) 3 of the input. In this case, these hypotheses are grouped to form a hypothesis. Table 3 shows the grouping result, where \( \textbf{Length} \) represents the number of the grouped curve segments.

3.3 Projective Refinement (Verification)

The verification method proposed in [16]. In this section, we review the method. Hypotheses generated in the previous prediction stage do not provide the exact correspondence because the length of equally divided boundary segments varies under the projective transformation. Therefore, hypotheses are verified by transforming and superimposing the model onto the scene. The verification is done by repeatedly extracting the projective parameters and modifying the length of boundary segments.

The data on the boundaries of an input scene and a model are represented by

\[
O_{k}^{ln} = \{X_{k}^{ln}, Y_{k}^{ln}\}, \quad k = 1 \sim n^{ln} \\
O_{k}^{Mo} = \{X_{k}^{Mo}, Y_{k}^{Mo}\}, \quad k = 1 \sim n^{Mo}
\]

where \( n^{ln} \) and \( n^{Mo} \) indicate the number of points on the boundary of the scene and the model.

For a correct matching between the model and scene with a different number of boundary points, the model and scene boundary are normalized or subsampled to form a descriptor of \( N \) boundary points. Let the normalized scene and model boundary data be

\[
q_i^{ln} = \{X_{\tau_1(i)}^{ln}, Y_{\tau_1(i)}^{ln}\}, \quad q_i^{Mo} = \{X_{\tau_2(i)}^{Mo}, Y_{\tau_2(i)}^{Mo}\}
\]

where

\[
\tau_1(i) = \frac{n_i^{ln}}{N} \times i, \quad \tau_2(i) = \frac{n_i^{Mo}}{N} \times i
\]

and \( N \) is the number of points of normalized contour.

The projective transformation between \( q_i^{ln} \) and \( q_i^{Mo} \) can be written to

\[
\begin{bmatrix}
q_1^{ln} \\
q_2^{ln}
\end{bmatrix} = \frac{1}{t_{31}u_i^{Mo} + t_{32}v_i^{Mo} + t_{33}} \begin{bmatrix}
t_{11} & t_{12} \\
t_{21} & t_{22}
\end{bmatrix} \begin{bmatrix}
u_i^{Mo} \\
v_i^{Mo}
\end{bmatrix} + \begin{bmatrix}
t_{13} \\
t_{23}
\end{bmatrix}
\]

or \( q_i^{ln} = (c \cdot q_i^{Mo} + t_{33})^{-1}(Aq_i^{Mo} + b) \).

Then, we compute the projective parameters by

<table>
<thead>
<tr>
<th>Model No.</th>
<th>Start Model No.</th>
<th>Start Input No.</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>:</td>
</tr>
<tr>
<td>26</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>37</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>38</td>
<td>6</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>
minimizing
\[ \varepsilon^2 = \sum_{i=1}^{N} w_i^2 (q_i^I - (c \cdot q_i^{Mo} + t_{33})^{-1}(A q_i^{Mo} + b))^T \cdot (q_i^I - (c \cdot q_i^{Mo} + t_{33})^{-1}(A q_i^{Mo} + b)). \] (10)

The variance defined by Eq. (2) is used to determine the weight \( w_i \) in Eq. (10).

\[ \frac{1}{w_i^2} = \sigma_0^2 = \sum_{j=1}^{5} \left( \frac{\partial \tilde{I}_i}{\partial x_j} \right)^2 \times E[(\tilde{I}_i - I_i)^2] \] (11)

From Eqs. (10) and (11), the projective parameters minimizing \( \varepsilon \) is given by

\[ P = (Q^TQ)^{-1}Q^TH \] (12)

where

\[ P = (t_{13}t_{23}t_{21}t_{22}t_{31})^T \]

\[ Q = \begin{bmatrix}
    w_iX_i^{Mo} & w_iY_i^{Mo} & w_i & 0 \\
    0 & 0 & 0 & w_iX_i^{Mo} \\
    \vdots & \vdots & \vdots & \vdots \\
    0 & w_iX_i^{Tn}X_i^{Mo} & w_iX_i^{Tn}Y_i^{Mo} & w_iX_i^{Tn} \\
    w_iY_i^{Mo} & w_iY_i^{Tn}X_i^{Mo} & w_iY_i^{Tn}Y_i^{Mo} & w_iY_i^{Tn} \\
    \vdots & \vdots & \vdots & \vdots 
\end{bmatrix} \]

\[ H = (\cdots - w_iX_i^{Tn} - w_iY_i^{Tn} \cdots)^T. \]

Using the projective parameters computed by Eq. (12), the model is transformed as

\[ q_i^{Mo} = (c \cdot q_i^{Mo} + t_{33})^{-1}(A q_i^{Mo} + b), \quad i = 1 \sim N. \] (13)

From the transformed boundary data, a normalized distance between two adjacent points on the boundary becomes

\[ \tau_2'(i) = \frac{|q_{i+1}^{Mo'} - q_i^{Mo'}|}{T}, \quad i = 1 \sim N \]

where \( T = \sum_{i=1}^{N} |q_{i+1}^{Mo'} - q_i^{Mo'}|. \) (14)

We resample the input boundary into segments whose length is given by \( \tau_2'(i) \). Equivalently,

\[ q_i^{In'} = O^{In'}_{\tau_2'(i)}, \quad i = 1 \sim N. \] (15)

The above procedures in Eqs. (10)–(15) are repeated until the projective parameters converge.
Table 5 Verification results.

<table>
<thead>
<tr>
<th>MODEL No.</th>
<th>Start Segment of MODEL</th>
<th>Start Segment of INPUT</th>
<th>Length</th>
<th>Pixel Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>8</td>
<td>6</td>
<td>5</td>
<td>0.6290</td>
</tr>
<tr>
<td>38(reversed)</td>
<td>7</td>
<td>13</td>
<td>5</td>
<td>0.7314</td>
</tr>
</tbody>
</table>

(a) Segmented contour for model 37
(b) Segmented contour for reversed model 38

Fig. 9 Segmented contour for matching models with the input.

4. Experiments

4.1 Model Database

As a model exercise, we have selected forty models with a general curved shape for 2-D object recognition, as shown in Fig. 7. The image size is 256 × 256 and the object recognition is implemented on an IBM 586 personal computer.

4.2 Experiments

Figure 8 shows the experimental results. In these figures, the bolded line represents the reconstructed models into the input images. Figure 8 (a) shows an input image consisted of Model 37 and reversed Model 38, and unknown object. Figure 8 (b) shows the segmented contour by inflection points, and (c) shows the reconstructed image for the recognized models. Figure 8 (d) shows the overlapping of the reconstructed models on the input image.

Table 4 represents the result of the grouping for generating hypotheses. Table 5 represents the verification results that are determined from the pixel error after the projective refinement.

Figure 9 shows the segments for the recognized models in the input image. In this figure, the term “reverse” means the reversed image of the model.

Figure 10 shows the various occluded images and is overlapped the reconstructed image for searched models. Figures 10 (a) and (b) show the complete matching results for the occluded input images. And Figs. 10 (c) and (d) show the recognition result for a single object.

5. Conclusion

In the paper, we introduced an approximated invariant descriptor based on the geometric invariance. Using this representation, we developed an object recognition algorithm for 2-D objects based on indexing and iterative projective refinement. And we analyze the error model for five points invariant that plays an important role. We use the error model to check the similarity between two invariants and to determine the searching range in an indexing table, and to determine the weight in obtaining iterative projective transformation parameters.

Since the invariant descriptor has a nice property that being invariant up to reparameterization and being well approximated for the perspective projection, our recognition algorithm was successfully able to recognize 2-D objects and to obtain the exact correspondence with the error below 1.0 pixel even though the images of the objects showed very large distortions.

It is numerically stable because of not including differential and not sensitive to the position error of the reference points and applicable of general curve including a convex curve. Since the number of invariance for
object recognition is at least equal to or more than the number of boundary data, the algorithm is more reliable than the conventional ones.

In experiments, the method has successfully recognized planar curved and occluded shapes, which show very large distortions due to the change in camera’s viewing directions obtaining exact correspondence. Our recognition algorithm, tested with forty models, has accomplished the recognition and pose estimation within 5sec CPU time on a 586 PC.

References