OBJECT RECOGNITION USING RADIUS PROFILES

Sándor I. JUHÁSZ
Advisors: Gábor HORVÁTH (BUTE-MIT), Atsushi SUGAHARA (Toshiba-RDC)

I. Introduction

Several different approaches can be used for image classification and object detection. In the first case the goal is only to decide whether a certain object is present on the image, in the latter the position and orientation of the objects should be found too.

Our goal was to build a real-time object detector. Real-time applications don't allow the use of complicated algorithms due to processing limitations. The detector should be able to detect objects with shiny metallic surfaces too. For such objects it is difficult to define a certain texture. The visible texture changes rapidly as the object, the camera or the lights change position or orientation. The objects on the images can be in any position and orientation. They can be in partial occlusion by other objects and they can cover each other too. The algorithm should be able to handle complex, concave objects too. It should find the objects on the images, calculate the centers and orientations.

A common way to find objects is to use image patches. Many methods use such features [1, 2, 3]. These can be combined with other types of features, for example contour information [4]. Unfortunately these models are not fast enough to be used real-time.

Psychology experiments show that contours alone can be used effectively for object identification even in noisy and partially occluded cases [5]. This fact led to several methods using only the contour of the object for recognition purposes. Some of these can learn object category features semi-automatically using only the boundary rectangle of the object as additional information [6, 7]. The model closest to our algorithm is described in [8].

Contour fragments are good choices for features as they can be computed easily on the fly and they work well in cases where no texture is available too. They can represent inner structures of the object as well. The similarity of these fragments and the image contour at a certain position can be defined by the Chamfer-distance. The common way to speed up this calculation is to generate the distance transformed image. This can be still too slow for real-time applications if the distance transformation is used many times.

To avoid speed problems and to achieve rotation-invariance we used a different feature: local radius profiles (see section II.). Distance transformation is no longer needed with the new feature and global searching can be avoided too by fitting the features only to feature points where the contour curvature have extremum.

II. Local Radius Profiles

Local radius profiles can be calculated at a high speed and only the contour of the object is needed for these calculations thus making them ideal features for this problem. Canny edge detector with some Gaussian smoothing is a good choice for robust contour detection. The short contour fragments should be deleted to speed up the search algorithm and to reduce noise in the results.

Features are extracted at curvature extrema points of the contour. These positions are scale- and rotation invariant. This property makes multi-scale matching easier. Feature points with curvatures above a high limit and under a low limit should be deleted as these represent very high curvature points which are sensitive to noise and very straight parts of lines with no stable maximum position.

A Definition of Local Radius Profiles

Figure 1 shows an example of how to calculate radius profiles. P is curvature extremum point of the contour. C is the center of the osculating circle at P with the radius *r*. It is used as the center for the calculations. We measure the distance between C and the analyzed contour at certain directions. A possible choice is to use a 60° interval on both sides of the CP axis. If the angle between the sampling directions is 2° then a 61-dimension vector will be the result. A bounding circle with a radius of *3r* can be used to limit the results. It makes the calculation of the radius possible at directions where the line starting from C cannot cross the contour. Radii are normalized so that CP distance equals 1. This choice makes the feature vectors scale-invariant. Rotational invariance is already guaranteed by the definition of the sampling directions.

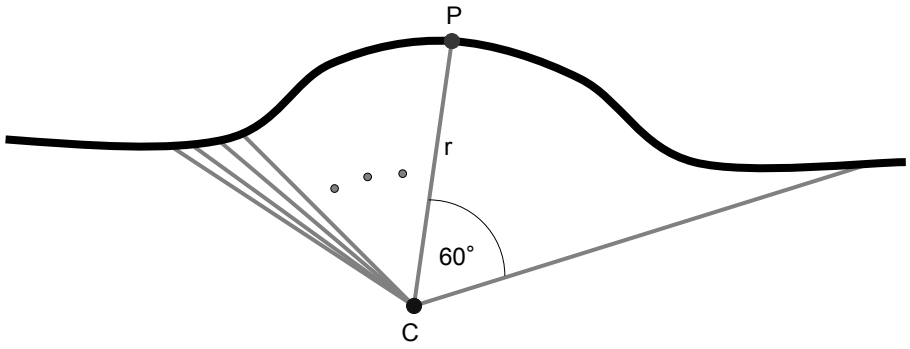

Figure 1: Structure of a radius profile

B Distance of radius profiles

With the above choice radius profiles are vectors of the 61 dimension space. We use the usual Euclidean distance to measure the distance between two such vectors γ_1 and γ_2:

$$D(\gamma_1, \gamma_2) = \sqrt{\sum_{i=1}^{61} (\gamma_{1,i} - \gamma_{2,i})^2} \,. \qquad (1)$$

III. Training

A Choosing Features

The radius profiles or features used for recognition are acquired from the training images. The object boundaries should be clearly visible and no occlusion should be present. Boundary boxes, rotation values and the centers of the objects are also stored to help future identification of the orientation.

In the first phase all the features are acquired. We use agglomerative clustering to reduce the number of features. Radius profiles that are similar enough and point to the same object center are merged into a cluster. Each cluster is represented by only one feature. ANN method [9] is used to speed up distance calculations.

The validation images can contain other objects too and occlusion is allowed. There should be images with only background objects (negative validation images) and images with the object to be recognized too (positive validation images).

B Building Weak Detectors

Radius profiles alone are features not good enough to reliably detect objects. We combine them to create better detectors. We combine *k* of them to form a weak detector. *k=2* and *k=3* are typically used. Every possible combination is generated except the ones where the features are on the same curve and are very close to each other as these represent the same feature place. The classification output of a weak detector h_i on image I is defined as

$$h_i(I) = \max_{p_1, p_2 \dots p_k} (h_i(I_{p_1, p_2 \dots p_k})). \quad (2)$$

It is the maximum value of the classifications considering every possible $p_1, p_2 \dots p_k$ position. The possible locations are the curvature extrema of the contours here too. The classification at the position $p_1, p_2 \dots p_k$ is defined as

$$h_i(I_{p_1, p_2 \dots p_k}) = \begin{cases} 1 & if \quad D_{h_i}(I_{p_1, p_2 \dots p_k}) < th_{h_i} \\ 0 & otherwise \end{cases}. \quad (3)$$

th_{h_i} is the learned threshold for weak detector h_i. If the result is 1 then the weak detector detected the object, otherwise it found nothing at that position. The distance inside the formula is a combination of the feature distances at the appropriate positions:

$$D_{h_i}(I_{p_1, p_2 \dots p_k}) = \frac{1}{m_s^2} \sum_{j=1}^{k} D(\gamma_j, \gamma_{I_{p_j}}) \quad (4)$$

where γ_j is a feature (radius profile) of the weak detector and $\gamma_{I_{p_j}}$ is a radius profile acquired from the image I at position p_j. The weight m_s is used to measure the compactness of the features' central votes (see [8]). Figure 2 shows a *k=3* example of a weak detector's object center guess. The big circle's center is the real object center. The small circle centers are the weak detector's features' object center guesses. $m_s = k$ and it is decreased by 0.5 each time the votes of two features in a weak detector for the center of the object have a distance more than *2r*. In the example m_s equals 2.

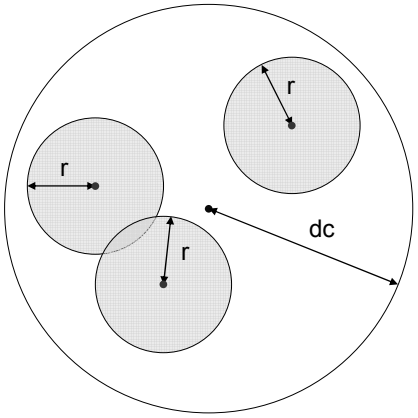

Figure 2: Feature votes for the object center in a weak detector

We have to evaluate the validation images to estimate the optimal th_{h_i} values. On the positive validation image set $D_{h_i}(I_{p_1, p_2 \dots p_k})$ is considered infinity if the respective object center estimations are not close enough to the real object center. *dc* is the maximum distance allowed.

C Building a Strong Detector

After defining the weak detectors h_i we should combine them to form a strong detector. The output of the strong detector *H* is the combination of the of the weak detectors' outputs:

$$H(I) = sign(\sum_{i=1}^{T} h_i(I) \cdot w_{h_i} - th). \quad (5)$$

T is the number of weak detectors, *th* is the threshold of detection. Adaboost [10] is used to calculate the w_{h_i} weights.

IV. Object Detection

Object detection tries to fit the weak detectors to the feature positions in every possible combination. A weak detector will fire if all of the following conditions are satisfied:

- equation (3) is 1
- the object center guesses are within a *dc* radius circle of their center of gravity
- the center guesses fit in their yaw rotation values too.

A Hough-style voting is used to calculate the object positions. Multiple planes are defined, each of them represents a certain yaw rotation of the object, and another series of planes are responsible for the different pitch rotations. A 2-D Gaussian is added to the adequate plane at the object center estimation weighted by the *w* weights each time a weak detector detects an object. Neighboring planes get votes too with smaller weights to compensate rounding errors.

Mean-shift algorithm is used to calculate local maxima of the sum of planes. These will be the object centers if they are above the *th* threshold (see equation 5). The rotational position of the object can be calculated by searching the plane that gives the highest value at the object center's position.

V. Conclusion

With the help of local radius profiles we were able to build a quick real-time object recognition system that can be used for any object, even for objects without any surface texture. Despite the low complexity of the algorithm it is able to detect complex objects with any 3D orientation and can operate successfully on images where the objects are partially occluded too.

Acknowledgement

I would like to acknowledge my advisor Atsushi Sugahara for his helpful ideas and endless patience during the development of the object recognition system. I also would like to thank Toshiba Corporation for making this research possible.

References

[1] R. Fergus, P. Perona, and A. Zisserman, "Object Class Recognition by Unsupervised Scale-invariant Learning," in *CVPR*, pp. 264-271, 2003.

[2] A. Torralba, K. P. Murhy, and W. T. Freeman, "Sharing features: efficient boosting procedures for multiclass object detection," in *CVPR*, pp. 762-769, 2004.

[3] B. Leibe, A. Leonardis, and B. Schiele, "Combined object categorization and segmentation with an implicit shape model," in *ECCV workshop on statistical learning in computer vision*, pp. 17-32, 2004.

[4] A. Opelt, A. Pinz, and A. Zisserman, "Fusing shape and appearance information for object category detection," in *Proceedings of the British Machine Vision Conference*, 2006.

[5] J. De Winter and J. Wagemans, "Contour-based object identification and segmentation: Stimuli, norms and data, and software tools," *Behavior Research Methods, Instruments, & Computers*, 36 (4): 604-624, 2004.

[6] J. Shotton, A. Blake, and R. Cipolla, "Multi-Scale Categorical Object Recognition Using Contour Fragments," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 30 (7): 1270-1281, 2008.

[7] V. Ferrari, F. Jurie, and C. Schmid, "From images to shape models for object detection," *Inria Technical Report*, *RR-6600*, 2008.

[8] A. Opelt, A. Pinz, and A. Zisserman, "A Boundary-Fragment-Model for Object Detection," in *ECCV*, pp. 575-588, 2006.

[9] S. Arya, T. Malamatos, and D. M. Mount, "Space-Time Tradeoffs for Approximate Spherical Range Counting," *16th Ann. ACM-SIAM Symposium on Discrete Algorithms*, pp. 535-544, 2005.

[10] Y. Freund, and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Computer and System Sciences*, 55 (1): 119-139, 1997.