Goal Oriented Edge Detection

Binnur Kurt, Muhittin Gökmen
Computer Engineering Department
İstanbul Technical University
Istanbul, Turkey
{bkurt,gokmen}@itu.edu.tr

***Abstract*—In many vision applications, there is a great demand for an edge detector which can produce edge maps with very different characteristics in nature, so that one of these edge maps may meet the requirements of the problem under consideration. Unfortunately it is not evident how to choose the desired or the optimum edge maps from these solutions that the edge detector offers. The proposed solutions are usually too general that cannot be easily adapted to the application needs by tuning edge detection parameters. One edge detector that we have studied in this study is Generalized Edge Detector which is capable of producing edges with very different characteristics. Although the edge maps based on this representation are reasonable, no one set of scale parameters alone yields a solution close to the desired edges. In this study, we have developed powerful edge operators and have used them under a goal-based edge detection framework. Proposed framework is a two-stage process. First, user marks some pixels in the database as edge and non-edge pixels. Then feature vectors comprised of filter responses to G-Filters at different scales are extracted at these marked pixels. Edge detection problem is imposed as two-class classification problem. Classifier itself is not adequate to extract desired edges for the application under consideration. In the second stage continuous edges are treated as one contour. Then contours are matched with the contours in the training set. Only matched contours are kept and the other contours are eliminated. The purpose of the first stage is to keep only prominent edges and remove irrelevant edges with respect to the application. The classifier decides which discontinuity is prominent or irrelevant. Experimental studies on real license plate images show that the proposed edge detector can successfully detects edges only on license plate regions.**

I. Introduction

The aim of edge detection is to provide a meaningful description of object boundaries in a scene from intensity surface. These boundaries are due to discontinuities manifesting themselves as sharp variations in image intensities. There are different sources for sharp changes in images which are created by structure (e.g. texture, occlusion) or illumination (e.g. shadows, highlights). Extracting edges from a still image is certainly the most significant stage of any computer vision algorithm requiring high accuracy of location in the presence of noise. In many contour-based vision algorithms, such as, curved-based stereo vision [1], contour-based image compression [2] and edge-based target recognition [3], edge-based face detection [4] their performance is highly dependent on the quality of the detected edges. Therefore, edge detection is an important area of research in computer vision. Despite considerable work and progress made on this subject, edge detection is still a challenging research problem due to the lack of a robust and efficient general purpose algorithm. Any edge detector should tackle with the tradeoff between good localization property forcing the location of the detected edges to be close as much as possible to the real edges and good noise rejection property forcing the intensity surface to be smooth. Without a priori assumption, one can not select the best tradeoff. In fact, deciding whether a pixel belongs to a contour is an ill-posed problem. The detection of sharp changes in image intensity requires the computation of derivatives of the noisy image at different orders. As known, the numerical computation of derivatives of the noisy data is an unstable process since it amplifies the noise. To overcome the noise problem, edge detection algorithms first employ a noise suppression process prior to the differentiation operation. This can be performed by smoothing the noisy image by a low-pass filter. Most of the efforts in edge detection have been devoted to the development of an optimum edge detector which can resolve the tradeoff ([5]–[8]). Furthermore, extracting edges at different scales and combining these edges have attracted a substantial amount of interest. In the course of developing optimum edge detectors that can resolve the tradeoff between localization and detection performances, several different approaches have resulted in either a Gaussian filter or a filter whose shape is very similar to a Gaussian ([5],

[6], [9]). Furthermore, these filters are very suitable for obtaining scale space edge detection since the scale of the filter can be easily controlled by means of a single parameter. Although these filters are used very widely, it is very difficult to claim that they can provide the desired output for any specific problem. For instance, there are some cases where the improved localization performance is the primary requirement. In these cases, a sub-optimum filter which promotes the localization performance becomes more appropriate.

II. Goal Oriented Edge Detection

In many vision applications , there is a great demand for an edge detector which can produce edge maps with very different characteristics in nature, so that one of these edge maps may meet the requirements of the problem under consideration. Unfortunately it is not evident how to choose the desired or the optimum edge maps from these solutions that the edge detectors offer. The proposed solutions are usually too general that cannot be easily adapted to the application needs by tuning edge detection parameters. One edge detector that we have studied in this study is Generalized Edge Detector (GED) [10] which is capable of producing edges with very different characteristics as we change the space parameters known as λ and τ. Although the edge maps based on this representation are reasonable, no one set of scale parameters alone yields a solution close to the desired edges. In this study, our aim is to find out powerful edge operators and use them under a goal-oriented edge detection framework.

Proposed framework is a two-stage process. First, user marks some pixels in the database as edge and non-edge pixels. Then feature vectors comprised of filter responses to $G-$Filters at different scales are extracted at these marked pixels. Edge detection problem is imposed as two-class classification problem. Support vector machine (SVM) is used as a classifier. Classifier itself is not adequate to extract desired edges for the application under consideration. In the second stage continuous edges are treated as one contour. Then contours are matched with the contours in the training set. Only matched contours are kept and the other contours are rejected.

The purpose of the first stage is to keep only prominent edges and remove irrelevant edges with respect to the applications requirements. The classifier decides which discontinuity is prominent or irrelevant. So how the training set is constructed is very important for the performance of the detector. Training set should contain almost equal number of edge and non-edge examples.

III. Stage I: Extracting Feature Vector

One of the key element in the framework is feature extraction. A user guides the algorithm by labeling the pixels as edge pixels and non-edge pixels. Since edges are discontinuity points, user should be careful in marking these edge pixels. Non-edge points usually belong to regions where intensity values are homogeneously distributed. When compared to An example is given in Fig. 1. Labeling is performed over noisy checkerboard image (Fig. 1b). Edge pixels are colored as black and non-edge pixels are colored as gray (Fig. 1c) and Fig. 1d). Non-edge points are labeled as blobs rather than labeling as pixel. Totally 502 edge pixels and 614 non-edge pixels are marked in Fig. 1c and 502 edge pixels and 1257 non-edge pixels are marked in Fig. 1d.

We first discretized $\lambda\tau - space$ by taking samples at regular intervals (λ_i, τ_i) where $\lambda_i = \lambda_0 + i * \Delta\lambda$ and $\tau_j = \tau_0 + j * \Delta\tau$,$i = 0, 1, \ldots, I$, and $j = 0, 1, \ldots, J$. The corresponding edge detection filters are given by $G_x(\lambda_i, \tau_j)$ and $G_y(\lambda_i, \tau_j)$. These filters are applied to the training images denoted by $f_k(m, n)$ only at labeled edge pixels $(edge_{f_k}(r) : \mathbb{R}^2 \times \mathbb{N} \rightarrow \mathbb{N}^2)$ and non-edge pixels $(none_{f_k}(r) : \mathbb{R}^2 \times \mathbb{N} \rightarrow \mathbb{N}^2)$. Filter responses are then aggregated into two feature vectors denoted by u and v; one for edge samples and one for non-edge samples. Formal definitions of the feature vectors are given as follows

$$\begin{aligned} u_{i,j,r}\left(f_k\right) &= \left(\left\|\left[G_x\left(\lambda_i, \tau_j\right) * f_k\right]\left(edge_{f_k}\left(r\right)\right)\right\|\right), \\ v_{i,j,s}\left(f_k\right) &= \left(\left\|\left[G_y\left(\lambda_i, \tau_j\right) * f_k\right]\left(nonedge_{f_k}\left(s\right)\right)\right\|\right) \end{aligned} \tag{1}$$

In order to study how sampling the $\lambda\tau-$space effects the feature vector distribution in the feature space, we set I to 2 and J to 1. In this case, the feature space is two dimensional where the feature vectors are reduced to the following form

$$\begin{aligned} \mathbf{u_r}(\mathbf{f_k}) &= (u_{1,1,r}, u_{2,1,r}), \\ \mathbf{v_s}(\mathbf{f_k}) &= (v_{1,1,s}, v_{2,1,s}). \end{aligned} \tag{2}$$

We have used six different sampling and two labeling schemes in our experiments for the two dimensional feature space. The corresponding λ and τ parameters used in the experiments and the feature space distributions are given in Fig. 2 and Fig. 3. Although noisy checkerboard image (Fig. 1b) is used in feature vector extraction, the vector distribution is very appropriate for classification for all sampling and marking schemes. Please note that the original checkerboard image (Fig. 1a) contains only step transitions and there are several topological structures in the intensity surface such as corners and

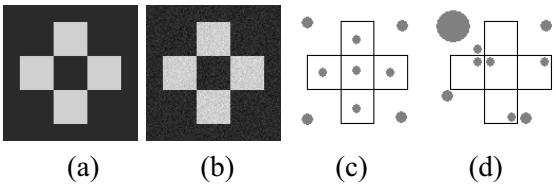


Fig. 1. Labeling pixels as edge and non-edge pixels (a) Original checkerboard image, (b) Noisy checkerboard image (c) Marked pixels (black pixels are edge points, gray pixels are non-edge points (d) Another marking used in training.

crossings. Since the marking in the Fig. 1d contains non-edge labels near these topological structures, we may expect that the edge detector trained with this labels yields better results compared to the edge detector trained with the other one. Let us call the edge detector trained with the labels in Fig. 1c as *Detector-1* and the edge detector trained with the labels in Fig. 1d as *Detector-2*.

The noisy checkerboard image shown in Fig. 1b is used in training, whereas the noisy checkerboard image with smaller SNR value (18dB) shown in Fig. 1c is used in testing. Edge detection results are shown in Fig. 4 and Fig. 5. Experimental results verify that marking process is the heart of the algorithm. Whereas *Detector-1* fails in detecting edges in the noisy checkerboard image with $SNR = 18dB$, *Detector-2* successfully captures the step transitions.

Another issue related to the proposed approach is the sampling scale space parameters (λ and τ). In two dimensional feature space case, detectors obtained with distant and large scale parameters such as $(\lambda_1 = 32.0, \tau_1 = 0.5), (\lambda_2 = 64.0, \tau_1 = 0.5)$ and $(\lambda_1 = 8.0, \tau_1 = 0.5), (\lambda_2 = 16.0, \tau_1 = 0.5)$ have poor detection performance.

Here we have used two different sampling and two labeling schemes in our experiments for this case. The corresponding λ and τ parameters used in the experiments and the feature space distributions are given in Fig. 6 and Fig. 7. Edge detection results are shown in Fig. 8. Three dimensional vector space performs better than the two dimensional space. Note that no post processing such as thinning or hysteresis thresholding is applied to the edge detection results. Let us call the edge detector trained with the labels in Fig. 1c as *Detector-3* and the edge detector trained with the labels in Fig. 1d as *Detector-4*.

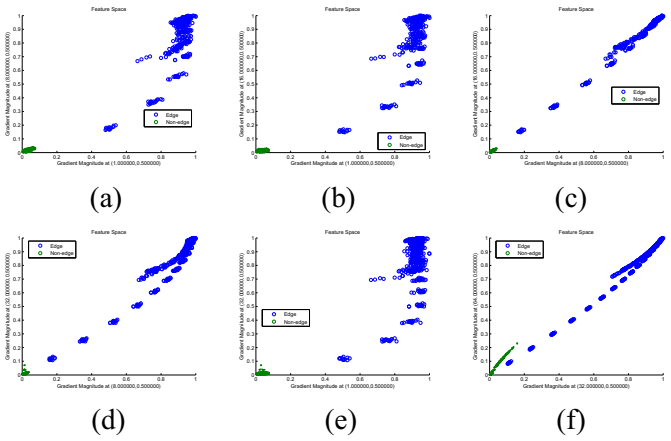


Fig. 2. Feature vector scatter diagrams with respect to different $\lambda\tau-$space sampling schemes when the labels given in Fig. 1(c) is used. (a) $(\lambda_1 = 1.0, \tau_1 = 0.5), (\lambda_2 = 8.0, \tau_1 = 0.5)$, (b) $(\lambda_1 = 1.0, \tau_1 = 0.5), (\lambda_2 = 16.0, \tau_1 = 0.5)$, (c) $(\lambda_1 = 8.0, \tau_1 = 0.5), (\lambda_2 = 16.0, \tau_1 = 0.5)$, (d) $(\lambda_1 = 8.0, \tau_1 = 0.5), (\lambda_2 = 32.0, \tau_1 = 0.5)$, (e) $(\lambda_1 = 1.0, \tau_1 = 0.5), (\lambda_2 = 32.0, \tau_1 = 0.5)$, (f) $(\lambda_1 = 32.0, \tau_1 = 0.5), (\lambda_2 = 64.0, \tau_1 = 0.5)$.

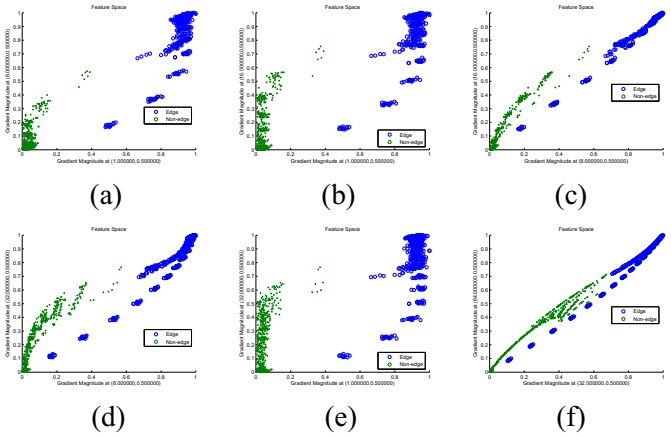


Fig. 3. Feature vector scatter diagrams with respect to different $\lambda\tau-$space sampling schemes when the labels given in Fig. 1(d) is used. (a) $(\lambda_1 = 1.0, \tau_1 = 0.5), (\lambda_2 = 8.0, \tau_1 = 0.5)$, (b) $(\lambda_1 = 1.0, \tau_1 = 0.5), (\lambda_2 = 16.0, \tau_1 = 0.5)$, (c) $(\lambda_1 = 8.0, \tau_1 = 0.5), (\lambda_2 = 16.0, \tau_1 = 0.5)$, (d) $(\lambda_1 = 8.0, \tau_1 = 0.5), (\lambda_2 = 32.0, \tau_1 = 0.5)$, (e) $(\lambda_1 = 1.0, \tau_1 = 0.5), (\lambda_2 = 32.0, \tau_1 = 0.5)$, (f) $(\lambda_1 = 32.0, \tau_1 = 0.5), (\lambda_2 = 64.0, \tau_1 = 0.5)$.

IV. Stage II: Contour Matching

Proposed framework has two steps. In the first, user marks some pixels in the database as edge and non-edge pixels. Then feature vectors comprised of filter responses to $G-$Filters at different scales are extracted at these labeled pixels. Edge detection problem is imposed as two-class classification problem. Classifier itself is not adequate to extract desired edges for the application under consideration. In the second step continuous edges are treated as one contour. Then contours are matched

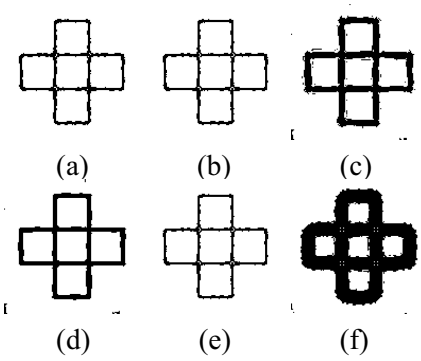


Fig. 4. Edge detection test results for the noisy checkerboard image (18dB). The labels given in Fig. 1(c) is used. The following $\lambda\tau-$space sampling schemes are used: (a) $(\lambda_1 = 1.0, \tau_1 = 0.5), (\lambda_2 = 8.0, \tau_1 = 0.5)$, (b) $(\lambda_1 = 1.0, \tau_1 = 0.5), (\lambda_2 = 16.0, \tau_1 = 0.5)$, (c) $(\lambda_1 = 8.0, \tau_1 = 0.5), (\lambda_2 = 16.0, \tau_1 = 0.5)$, (d) $(\lambda_1 = 8.0, \tau_1 = 0.5), (\lambda_2 = 32.0, \tau_1 = 0.5)$, (e) $(\lambda_1 = 1.0, \tau_1 = 0.5), (\lambda_2 = 32.0, \tau_1 = 0.5)$, (f) $(\lambda_1 = 32.0, \tau_1 = 0.5), (\lambda_2 = 64.0, \tau_1 = 0.5)$.

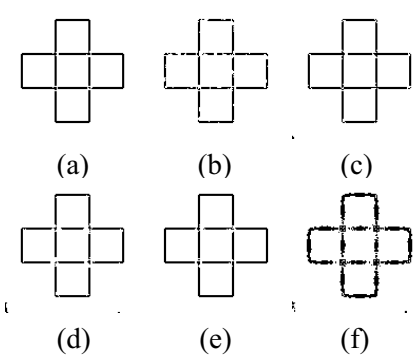


Fig. 5. Edge detection test results for the noisy checkerboard image (18dB). The labels given in Fig. 1(d) is used. The following $\lambda\tau-$space sampling schemes are used: (a) $(\lambda_1 = 1.0, \tau_1 = 0.5), (\lambda_2 = 8.0, \tau_1 = 0.5)$, (b) $(\lambda_1 = 1.0, \tau_1 = 0.5), (\lambda_2 = 16.0, \tau_1 = 0.5)$, (c) $(\lambda_1 = 8.0, \tau_1 = 0.5), (\lambda_2 = 16.0, \tau_1 = 0.5)$, (d) $(\lambda_1 = 8.0, \tau_1 = 0.5), (\lambda_2 = 32.0, \tau_1 = 0.5)$, (e) $(\lambda_1 = 1.0, \tau_1 = 0.5), (\lambda_2 = 32.0, \tau_1 = 0.5)$, (f) $(\lambda_1 = 32.0, \tau_1 = 0.5), (\lambda_2 = 64.0, \tau_1 = 0.5)$.

with the contours in the training set. Only matched contours are kept and the other contours are rejected. In contour matching we have used the algorithm developed in this study. The algorithm is explained in [11].

Contour matching stage will be explained using license plate recognition system. Applications such as traffic measurement and management, traffic surveillance, car park automation require successful license plate recognition (LPR) system. A typical LPR system is comprised of two parts: license plate detection (LPD) and license plate localization (LPL), each targets different aims. LPD aims at detecting the presence of plate in an image, while LPL aims at extracting the actual boundary and segmenting the plate characters which are ready for the optical character recognition.

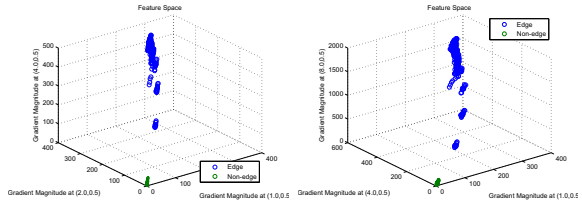
The license plate image database contains 8321 gray-scale digit characters segmented from the real vehicle license plates. Edge detector explained in the previous section is trained with the digit characters and their boundaries in the database. Fig. 9a contains a vehicle image. After the first stage the edges given in Fig. 9b are detected. Contour matching is applied to the detected contour. Since non-digit contours are not matched, they are eliminated. After the contour elimination, the contours given in Fig. 9c are obtained.

V. Conclusion

This study presents a new approach to edge detection. We presented a new edge detection formulation within a goal-based, regularized, shape-aware framework. This framework provides mechanisms for achieving detection without requiring any thresholding or parameter tuning, also taking the contour geometry into account. We proposed a two-stage framework. In the first stage, a user marks some pixels in the database as edge and non-edge pixels as the underlying application requires. Then feature vectors comprised of filter responses to G-Filters at different scales are extracted at these marked pixels. Edge detection problem is imposed to two-class classification problem. Classifier itself is not adequate to extract desired edges for the application under consideration. In the second stage continuous edges are treated as one contour. Then contours are matched with the contours in the training set. Only matched contours are kept and the other contours are eliminated.

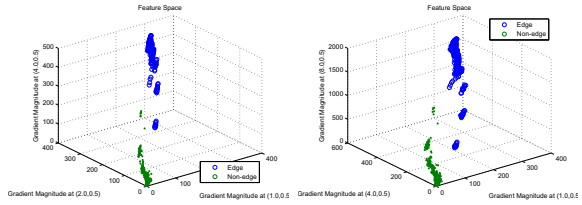
The purpose of the first stage is to keep only prominent edges and remove irrelevant edges with respect to the application. The classifier decides which discontinuity is prominent or irrelevant. So how the training set is constructed is very important for the performance of the detector. Training set should contain almost equal number of edge and non-edge examples.

The use of goal-oriented approach will find its applications in robust, accurate, efficient edge and contour extraction. We believe that the approach developed here leads to powerful edge/contour-based systems especially for applications involving face recognition and license plate detection.



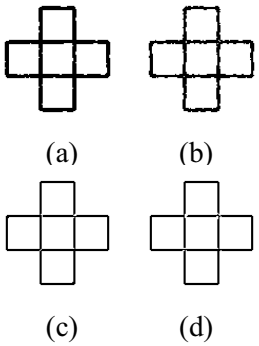
(a) (b)

Fig. 6. Feature vector scatter diagrams with respect to different $\lambda\tau-$space sampling schemes when the labels given in Fig. 1(c) is used. (a) $(\lambda_1 = 1.0, \tau_1 = 0.5)$, $(\lambda_2 = 2.0, \tau_1 = 0.5)$, $(\lambda_3 = 4.0, \tau_1 = 0.5)$, (b) $(\lambda_1 = 1.0, \tau_1 = 0.5)$,$(\lambda_2 = 4.0, \tau_1 = 0.5)$,$(\lambda_3 = 8.0, \tau_1 = 0.5)$.



(a) (b)

Fig. 7. Feature vector scatter diagrams with respect to different $\lambda\tau-$space sampling schemes when the labels given in Fig. 1(d) is used. (a) $(\lambda_1 = 1.0, \tau_1 = 0.5)$, $(\lambda_2 = 2.0, \tau_1 = 0.5)$, $(\lambda_3 = 4.0, \tau_1 = 0.5)$, (b) $(\lambda_1 = 1.0, \tau_1 = 0.5)$, $(\lambda_2 = 4.0, \tau_1 = 0.5)$, $(\lambda_3 = 8.0, \tau_1 = 0.5)$.



(a) (b)

(c) (d)

Fig. 8. Edge detection test results for the noisy checkerboard image (18dB). The labels given in Fig. 1(c) is used in (a) and (b). The labels given in Fig. 1(d) is used in (c) and (d). The following $\lambda\tau-$space sampling schemes are used: (a),(c) $(\lambda_1 = 1.0, \tau_1 = 0.5)$, $(\lambda_2 = 2.0, \tau_1 = 0.5)$, $(\lambda_3 = 4.0, \tau_1 = 0.5)$, (b),(d) $(\lambda_1 = 1.0, \tau_1 = 0.5)$, $(\lambda_2 = 4.0, \tau_1 = 0.5)$, $(\lambda_3 = 8.0, \tau_1 = 0.5)$

(a) (b) (c)

Fig. 9. Edge detection results (a) Vehicle image (b) Detected edges (c) Detected contours.

REFERENCES

[1] **Nasrabadi, N.M.**, 1992. A Stereo-Vision Technique using Curve Segments and Relaxation Matching, IEEE Transactions On Pattern Analysis and Machine Intelligence, Vol. 14 **(5)**, pp. 566-572.

[2] **Kurt, B. and Gökmen, M.**, 2005. Image Compression Based On Centipede Model, ARI, The Bulletin of the Istanbul Technical University, Vol.54 **(3)**, pp. 35-43.

[3] **Olson, C.F. and Huttenlocher, D.P.**, 1997. Automatic Target Recognition by Matching Oriented Edge Pixels, IEEE Transactions on Image Processing, Vol. 6**(1)**, pp. 103-113.

[4] **Yılmaz, A. and Gökmen, M.**, 1999. Face Recognition Based on Eigenhills, 7th Conference on Signal Processing and Applications, SIU'99.

[5] **Marr, D. and Hildreth, E.**, 1980. Theory of Edge Detection, Proceedings of Roy. Soc., (Sec B,207), pp.187-217.

[6] **Canny, J.F.**, 1986. A Computational Approach to Edge Detection, IEEE Transactions On Pattern Analysis and Machine Intelligence, Vol. 8, pp. 679-698.

[7] **Deriche, R.**, 1987. Optimal Edge Detection using Recursive Filtering, Proceedings of IEEE First International Conference on Computer Vision, pp. 501-505.

[8] **Shen, J. and Castan, S.**, 1986. An Optimal Linear Operator for Edge Detection, Proceedings of IEEE First Conference on Pattern Recognition, pp. 109-114.

[9] **Poggio, T., Voorhees, H. and Yuille, A.**, 1985. A Regularized Solution to Edge Detection, Technical Report, MIT AI Lab., AI Memo 833.

[10] **Gökmen, M. and Jain, A.K.**, 1997. $\lambda\tau$-Space Representation of Images and Generalized Edge Detector, IEEE Trans. on PAMI, Vol. 19 **(6)**, pp. 545-563.

[11] **Çapar, A., Kurt, B. and Gökmen, M.**, 2008. Gradient Based Shape Detectors, *appear to* Machine Vision and its Applications.