REAL-TIME OBJECT DETECTION FOR "SMART" VEHICLES

D.M. Gavrila

Image Understanding Systems
DaimlerChrysler Research
Ulm 89081, Germany
dariu.gavrila@DaimlerChrysler.com

V. Philomin

Computer Vision Laboratory
University of Maryland
College Park, MD 20742, U.S.A.
vasi@cs.umd.edu

ABSTRACT

This paper presents an efficient shape-based object detection method based on Distance Transforms and describes its use for real-time vision on-board vehicles. The method uses a template hierarchy to capture the variety of object shapes; efficient hierarchies can be generated offline for given shape distributions using stochastic optimization techniques (i.e. simulated annealing). Online, matching involves a simultaneous coarse-to-fine approach over the shape hierarchy and over the transformation parameters. Very large speed-up factors are typically obtained when comparing this approach with the equivalent brute-force formulation; we have measured gains of several orders of magnitudes.

We present experimental results on the real-time detection of traffic signs and pedestrians from a moving vehicle. Because of the highly time sensitive nature of these vision tasks, we also discuss some hardware-specific implementations of the proposed method as far as SIMD parallelism is concerned.

1. INTRODUCTION

Slowly but steadily, vehicles are becoming "smarter". Using various sensors, they can provide the driver with relevant information about the surroundings and if desired, even perform simple vehicle control tasks (e.g. [5] [4]). The first products are already gearing up to the market, see for example the IR-based night vision system of the Cadillac DeVille car and the radar-based "Distronic" Active Cruise Control system of the new Mercedes-Benz S-Class car.

An important component of more advanced on-board vision systems is the ability to detect objects. For example, a Traffic Sign Assistant might inform the driver if he makes a wrong turn in a one-way street or if he is speeding. Alternatively, a system to detect pedestrians might reduce the accident rate by taking either passive or active measures to deal with upcoming collisions.

In this paper, we present a shape-based method which can be used for that purpose; it is general enough to detect objects of arbitrary shapes. Models or other parametrizations need not to be established explicitly, which is an advantage when dealing with non-rigid objects such as pedestrians. Instead, the method is able to generate an efficient representation from example shapes off-line; matching proceeds on-line using a novel variant of Distance Transform (DT) - based matching.

The outline of the paper is as follows. Section 2 reviews previous work on DTs. Section 3 presents the proposed DT-based representation and matching method. Because of stringent speed requirements, we discuss ways to additionally speed up the algorithm by hardware-specific means (i.e. SIMD instructions). Section 5 lists our experiments on traffic sign and pedestrian detection. We conclude in Section 6.

2. PREVIOUS WORK ON DTS

Matching with DT is illustrated schematically in Figure 1. It involves two binary images, a segmented template T and a segmented image I, which we'll call "feature template" and "feature image". The "on" pixels denote the presence of a feature and the "off" pixels the absence of a feature in these binary images. What the actual features are, does not matter for the matching method. Typically, one uses edge- and corner-points. The feature template is given off-line for a particular application, and the feature image is derived from the image of interest by feature extraction.

Matching T and I involves computing the distance

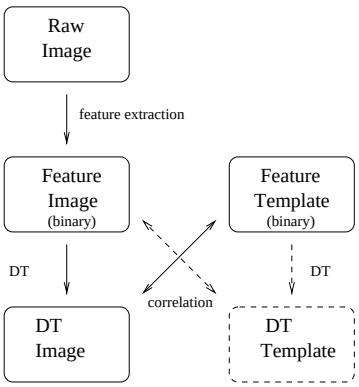


Figure 1: Matching using a DT

transform of the feature image I. The template T is transformed (e.g. translated, rotated and scaled) and positioned over the resulting DT image of I; the matching measure $D(T, I)$ is determined by the pixel values of the DT image which lie under the "on" pixels of the transformed template. These pixel values form a distribution of distances of the template features to the nearest features in the image. The lower these distances are, the better the match between image and template at this location. There are a number of matching measures that can be defined on the distance distribution. One possibility is to use the average distance to the nearest feature. This is the *chamfer* distance.

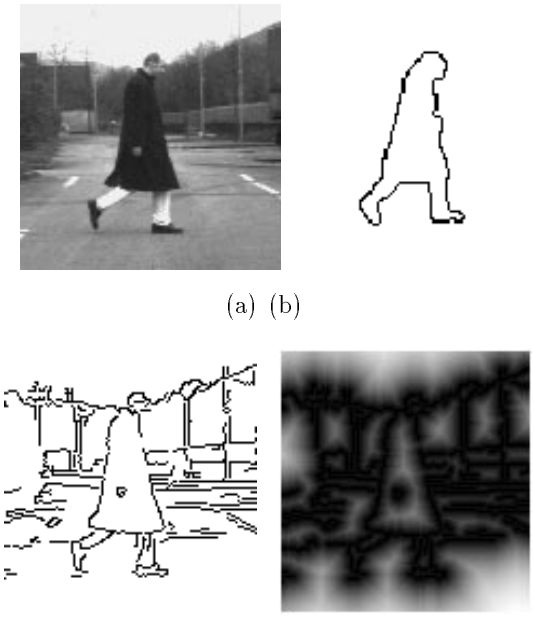
$$D_{chamfer}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t) \qquad (1)$$

where $|T|$ denotes the number of features in T and $d_I(t)$ denotes the distance between feature t in T and the closest feature in I. Thus, the chamfer distance consists of a correlation between T and the distance image of I, followed by a division. Other more robust (and costly) measures reduce the effect of missing features (i.e. due to occlusion or segmentation errors) by using the average truncated distance or the f-th quantile value (the *Hausdorff* distance) [8] [15].

In applications, a template is considered matched at locations where the distance measure $D(T, I)$ is below a user-supplied threshold θ

$$D(T, I) < \theta \qquad (2)$$

Figure 2 illustrates the matching scheme of Figure 1 for the typical case of edge features. Figure 2a-b shows an example image and template. Figure 2c-d shows the edge detection and DT transformation of the edge image. The distances in the DT image are intensity-coded; lighter colors denote larger distance values.



(a) (b)

(c) (d)

Figure 2: (a) original image (b) template (c) edge image (d) DT image

The advantage of matching a template (Figure 2b) with the DT image (Figure 2d) rather than with the edge image (Figure 2c) is that the resulting similarity measure will be smoother as a function of the template transformation parameters. This enables the use of various efficient search algorithms to lock onto the correct solution, as will be discussed shortly. It also allows more variability between a template and an object of interest in the image. Matching with the unsegmented (gradient) image, on the other hand, typically provides strong peak responses but rapidly declining off-peak responses.

A number of extensions have been proposed to the basic DT matching scheme. Some deal with hierarchical approaches to improve match efficiency and use multiple image resolutions [2]. Others use a pruning [13] [8] or a coarse-to-fine approach [15] in the parameter space of relevant template transformations. The latter approaches take advantage of the smooth similarity measure associated with DT-based matching; one need not to match a template for each location, rotation or other transformation. Other extensions involve the use of a un-directed ("symmetric") similarity measure between image and a template [8] (e.g. see Figure 1). Yet other extensions deal with multiple feature types [6] [11], consider salience measures [14], or use probabilistic frameworks [10].

With the exception of [11], previous work on DT-based matching has dealt with the case of matching one template against an image, allowing certain geometrical transformations (e.g. translation, rotation, affine). Work by Olson [11] discusses a way to avoid matching duplicate points across multiple templates. Like Olson, we consider the more general case where we match multiple templates. However, templates do not need to have any point in common with other templates; the idea is to capture shape variability by a "prototype" template and a distance parameter. Consequently, a different search algorithm results, see next Section.

3. MATCHING MULTIPLE SHAPES

We now discuss the online and offline components of the proposed method.

3.1. Matching using a template hierarchy

When matching N templates to an image, which bear no relationship to each other, one can hope for little performance gain compared to a method which matches the templates separately. Fortunately, in most applications, there is an underlying structure in the template distribution and by combining a template hierarchy with a coarse-to-fine search over the image, one can do better. The idea is that at a coarse level of search, when the image grid size of the search is large, it would be inefficient to match each of the N objects separately, if they are relatively similar to each other. Instead, one would group similar templates together and represent them by a prototype template; matching would be done with this prototype, rather than with the individual templates, resulting in a (potentially significant) speed-up. This grouping of templates is done at various levels, resulting in a hierarchy, where the leaf level contains the N templates one needs to match with and the intermediate levels contains the prototypes.

To make matters more concrete, consider first the case of a coarse-to-fine search where one matches a single template under translation. Assume there are L levels of search ($l = 1, ..., L$), determined by the size σ_l of the underlying uniform grid and the distance threshold θ_l which determines when a template matches sufficiently enough to consider matching on a finer grid (in the neighborhood of the promising solution). Let τ_{tol} denote the allowed tolerance on the distance measure between template and image at a "correct" location. Let μ denote the distance along the diagonal of a unit grid element. Then by having

$$\theta_l = \tau_{tol} + \frac{1}{2}\mu\sigma_l \tag{3}$$

one has the desirable property that, using un-truncated distance measures such as the chamfer distance, one can assure that the coarse-to-fine approach will not miss a solution. The second term accounts for the (worst) case that the solution lies at the center of the 4 enclosing grid points which form a square.

Now consider the case where the above L-level search is combined with a L-level template hierarchy. Matching can be seen as traversing the tree structure of templates. Each node corresponds to matching a (prototype) template $\mathbf{p}$ with the image at node-specific locations. For the locations where the distance measure between template and image is below a user-supplied threshold θ_p, one computes new interest locations for the children nodes (generated by sampling the local neighborhood with a finer grid) and adds the children nodes to the list of nodes to be processed. The matching process starts at the root, the interest locations lie initially on a uniform grid over relevant regions in the image. The tree can be traversed in breadth-first or depth-first fashion. In the experiments, we use depth-first traversal, which has the advantage that one needs to maintain only $L - 1$ sets of interest locations.

Let $\mathbf{p}$ be the template corresponding to the node currently processed during the traversal and let $C = \{\mathbf{t}_1, ..., \mathbf{t}_c\}$ be the set of templates corresponding to its children nodes. Let δ_p be the maximum distance between $\mathbf{p}$ and the elements of C.

$$\delta_p = \max_{t_i \in C} D(\mathbf{p}, \mathbf{t}_i) \tag{4}$$

Then by having

$$\theta_p = \tau_{tol} + \delta_p + \frac{1}{2}\mu\sigma_l \tag{5}$$

one has the desirable property that, using untruncated distance measures such as the chamfer distance, one can assure that the coarse-to-fine approach using the template hierarchy will not miss a solution. The thresholds one obtains by Equation (5) are quite conservative, in practice one can use lower thresholds to speed up matching, at the cost of possibly missing a solution (see Experiments).

3.2. Constructing the template hierarchy

Here we discuss a way to automatically generate the template hierarchy from the available example templates. The proposed algorithm uses a bottom-up approach and applies a "K-means"-like algorithm at each

level of the hierarchy. The input to the algorithm is a set of templates $\mathbf{t}_1, ..., \mathbf{t}_N$, their dissimilarity matrix (see below) and the desired partition size K. The output is the K-partition and the prototype templates $\mathbf{p}_1, ..., \mathbf{p}_K$ for each of the K groups $S_1, ..., S_K$. The K-way clustering is achieved by iterative optimization. Starting with an initial (random) partition, templates are moved back and forth between groups while the following objective function E is minimized

$$E = \sum_{k=1}^{K} \max_{\mathbf{t}_i \in S_k} D(\mathbf{t}_i, \mathbf{p}_k^*) \quad (6)$$

Here, $D(\mathbf{t}_i, \mathbf{p}_k^*)$ denotes the distance measure between the i-th element of group k and the prototype for that group at the current iteration. The distance measure is the same as the one used for matching (e.g. chamfer or Hausdorff distance). Entry $D(i, j)$ is the ijth member of the dissimilarity matrix, which can be computed fully before grouping or only on demand.

One way of choosing the prototype $\mathbf{p}_k^*$ is to select the template with the smallest maximum distance to the other templates. Clearly, a low E-value is desirable since it implies a tight grouping; this lowers the distance threshold that needs to be used during matching (by Equation 5) which in turn likely decreases the number of locations which one needs to consider during matching. We use simulated annealing [9] to perform the minimization of E.

Simulated annealing is a well-known stochastic optimization technique where during the initial stages of the search procedure, moves can be accepted which increase the objective function. The idea is to do enough exploration of the search space, before resorting to greedy moves, in order to avoid local minima. Candidate moves are accepted according to probability p

$$p = \frac{1}{1 + e^{\frac{\Delta E}{T}}} \quad (7)$$

where T is the temperature parameter which is adjusted according to a certain "cooling" schedule (we use an exponential schedule [9]).

Since the template hierarchy is constructed off-line, it is worth spending a substantial effort to devise an efficent template hierarchy (in the sense of minimizing E), because this results in on-line computational gains.

4. HARDWARE-SPECIFIC OPTIMIZATIONS

Many general-purpose processors come nowadays equipped with special Single-Instruction Multiple Data (SIMD) instruction sets. For example, Intel's Pentium II MMX technology allows concurrent byte-wise operations on 64-bit registers. In order to fully exploit the capabilities of our on-board processor, we implemented the two main speed bottlenecks of our method in MMX. Here is pseudo-C-code for the original (sequential) implementation of the computationally-intensive chamfer transform (with x-y kernel, forward pass, image width W, image height H), adjusted from [1]:

```
for (int i=1; i<H-1; ++i)
 for (int j=1; j<W-1; ++j)
  D[i][j] = min(D[i][j],
              min(D[i][j-1]+x,
                min(D[i-1][j]+x,
                  min(D[i-1][j-1]+y, D[i-1][j+1]+y))));
```

and here is the equivalent SIMD pseudo code (assuming we have K-byte registers Rx, Ry and R1-R5, and R[i] denotes the i-th byte of register R; disregarding boundary conditions):

```
Ry = [y, ..., y]; Rx = [x, ..., x];
for (int i=1; i<H-1; ++i)
  for (int j=1; j<W-K-1; j+=K) {
     R0 = [D[i-1][j-1], ..., D[i-1][j-1+K-1]];
     R0 = SIMD_add(R0, Ry);
     R1 = [D[i-1][j+1], ..., D[i-1][j+1+K-1]];
     R1 = SIMD_add(R1, Ry);
     R2 = [D[i-1][j]  , ..., D[i-1][j+K-1]];
     R2 = SIMD_add(R2, Rx);
     R3 = [  D[i][j]  , ...,   D[i][j+K-1]];
     R3 = SIMD_add(R3, Rx);
     R4 = SIMD_min(R0,
            SIMD_min(R1,
               SIMD_min(R2, R3)));
     R5[0] = D[i][j-1];
     for (int k=1; k<K, ++k)
         R5[k] = min(R5[k-1], R4[k]);
     [  D[i][j]  , ...,   D[i][j+K-1]  ] = R5;
}
```

The number of addition and comparison operations in the inner loop for the SIMD case is $7+K-1$, the equivalent number for the sequential formulation is $8 * K$ (actual program profiling showed a speed-up of factor 4 for the case of MMX, with K=8).

The other main computational bottleneck is the correlation between templates and the (chamfer) images. Since SIMD correlation is well known, we do not list it here (the measured speed-up for MMX was also factor 4).

5. EXPERIMENTS

To illustrate the proposed matching method, we applied it to the detection of traffic signs and pedestrians and performed experiments off-line as well as on-board our demo vehicle, see Figure 3. Subsequent references to processing speed involve a 450 MHz dual-Pentium II processor with MMX.

In both sets of experiments we used a three-level template hierarchy, which during matching was traversed in a depth-first order. We used oriented edges as features; the orientations were discretized in 8 values ("feature types"). To improve efficiency, the template points were pre-sorted according to their feature type. Similarly, 8 distance images were derived from the scene image, so that correlation took place between corresponding types [6]. An increase in computational efficiency was obtained by subsampling the template points, based on the level of the corresponding node in the hierarchy. We used a point sampling rate of 8,4,1 for the three levels from top to bottom, respectively. The spatial grid sizes on which templates were matched with the image were $\sigma = 8, 4, 1$, respectively.

Because of real-time requirements, we ended up using the chamfer distance measure (i.e. Equation 1). To alleviate effects of missing data, we imposed a (relatively low) maximum value on the chamfer image pixels. Independently of the distance measure used, we found that having essentially only one edge segmentation threshold was not always appropriate. A restrictive value would result in sufficient edges to guide the search at the coarser level of the hierarchy, but matching at the finer level would suffer. Setting the edge threshold to include all edges needed for a fine-level match would be computationally intensive and degrade the underlying coarse-to-fine concept. We explored two approaches to this issue. The first involved using multiple edge thresholds and multiple sets of distance images based on the level of the hierarchy where matching was conducted (we used two edge thresholds, for leaf and non-leaf level matching, respectively). The other solution involved using a normalized (un-thresholded) gradient image when matching at the leaf level.

5.1. Traffic signs

Our first application was the detection of circular and triangular (up/down) traffic signs, as seen on highways and secondary roads (e.g. see [3]). In these experiments we did not consider signs which are significantly tilted and/or skewed in the image; only scale and translation were explicitly accounted for. We used templates for circles and triangles with radii in the range of 7-18 pixels (the images are of size 360 by 288 pixels). This lead to a total of 36 templates, for which a template tree was specified "manually" as in Figure 4. In order to optimize for speed, we chose to scale the templates (off-line), rather than scale the image (on-line).

Figure 3: on-board camera and display

We did extensive tests on the traffic sign detection application. Off-line, we used a database of 1000 traffic sign images, taken during both day- (sunny, rainy) and night-time conditions. We obtained high single-image detection rates, typically, of over 95%, when allowing solutions to deviate by 2 pixels and by radius 1 from the values obtained by a human. At this rate, there were one or two detections per image which were not traffic signs, on average. These false positives were overwhelmingly rejected in a subsequent verification phase, where a RBF network was used as pictograph classifier (the latter could distinguish about 10 pictographs). See Figure 5. The traffic signs that were not detected, were either tilted or otherwise, reflected difficult environmental conditions (e.g. rain drops, partial occlusion by window wiper, direct sunlight into camera). Under the latter conditions, detection rates could decrease by as much as 15%, to 80%. We spent many hours testing our system on the road. The traffic sign detection (and recognition) system currently runs at 10-15 Hz.

5.2. Pedestrians

Our second application involves the detection of pedestrians. Not surprisingly, it is the more challenging task of the two; it involves a much larger variety of shapes that needed to be accounted for and the pedestrian contours are less pronounced in the images. Note that with a few exceptions (e.g. [12] much of the previous work on "Looking at People" [7] has involved a static camera; initial segmentation was possible by

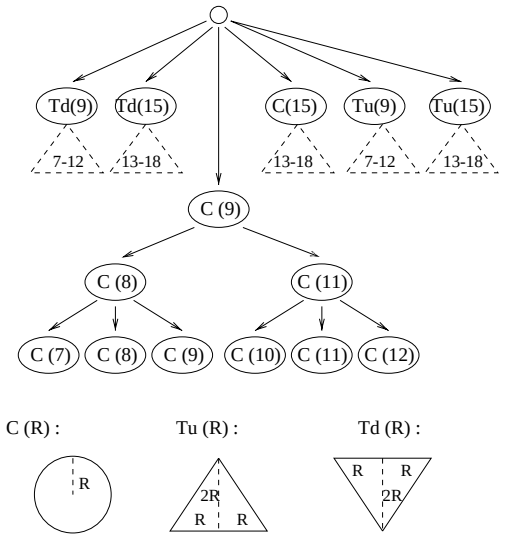


Figure 4: A hierarchy for traffic sign shapes (hard-coded)

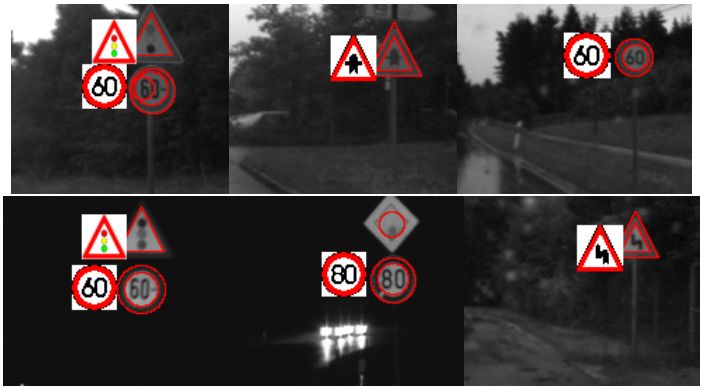


Figure 5: Traffic sign detection (and recognition)

background subtraction. Furthermore, it is difficult for a user to hard-code the shape hierarchy; it needs to be constructed automatically from examples. We compiled a database of about 1000 pedestrian shapes, normalized for scale. Using 5 scales (range 70-102 pixels) we obtained a pedestrian hierarchy of about 5500 templates, following the method described in Section 3.2. See Figure 6 for a partial view. Observe how the shape similarity increases towards the leaf level.

Our preliminary experiments on a database of 700 pedestrian images (distinct from the sequences used for training) resulted in a detection rate of about 75-85% per image, when requiring the number of false positives to be two or less per image. See Figure 7 for a few detection results. The last two images of Figure 7 were some of the IR-images we started recording; as seen

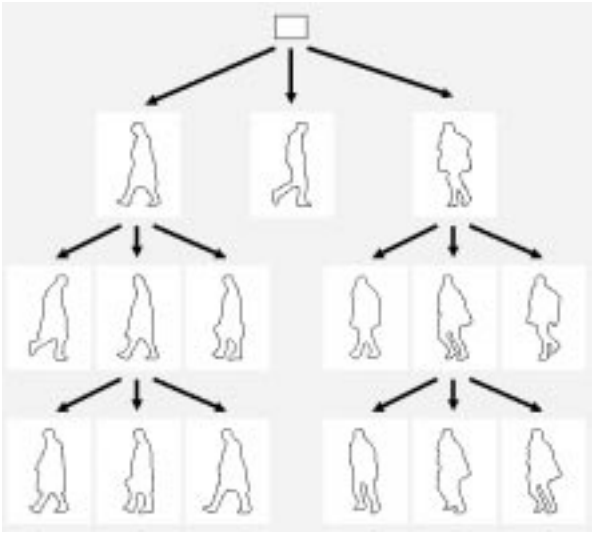


Figure 6: A hierarchy for pedestrian shapes (partial view)

they provide good segmentation opportunities. Figure 8 shows examples of false positives; typically they are found on trees or windows. Using the flat-world assumption and knowledge about camera geometry, we have set region of interests for the template in the hierarchy, so that the erroneous template positions shown in Figure 8 can be a-priori be excluded, speeding up matching greatly. The current pedestrian system runs at 1-5 Hz, the first road trials are currently underway.

In general, given image width W, image height H, and K templates, a brute-force matching algorithm would require $W \times H \times K$ correlations between template and image. In the presented hierarchical approach both factors $W \times H$ and K are pruned (by a coarse-to-fine approach in image space and in template space). It is not possible to provide an analytical expression for the speed-up, because it depends on the actual image data and template distribution. Nevertheless, for this pedestrian application, we measured speed-ups of three orders of magnitude.

6. CONCLUSIONS AND FUTURE WORK

We proposed a method for shape-based object detection using Distance Transforms, which takes a combined coarse-to-fine approach in shape and parameter space, incorporating a multi-stage segmentation technique as well. An indication of the efficiency of the method was the significant speed up obtained by comparing the method to an equivalent brute-force method.

Figure 7: Pedestrian detection results

Figure 8: False positives

We also demonstrated that it allows to perform some challenging tasks in (near) real-time, involving object detection from a moving vehicle. Naturally, some limitations exist. For example, even with a multi-stage edge thresholding technique, matching remains dependent on a reasonable contour segmentation. Furthermore, although we dealt with a sizeable amount of shape variation when considering pedestrian shapes, this method might be not the most appropriate to detect pedestrians very close to the camera when shape variations become even larger. Nevertheless, the proposed method operated quite successfully in our vehicle, and with further work (e.g. temporal integration of results, integration with stereo/IR) we hope to come close to the demanding performance rates that might be required for actual deployment of such a system.

7. REFERENCES

[1] H. Barrow et al. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *International Joint Conference on Artificial Intelligence*, pages 659–663, 1977.

[2] G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–865, November 1988.

[3] G. Piccioli *et. al.* Robust method for road sign detection and recognition. *Image and Vision Computing*, 14:209–223, 1998.

[4] U. Handman *et al.* An image processing system for driver assistance. In *Proc. of Intelligent Vehicles Conference*, pages 481–486, Stuttgart, Germany, 1998.

[5] U. Franke, D. Gavrila, S. Görzig, F. Lindner, F. Pätzhold, and C. Wöhler. Autonomous driving goes downtown. *IEEE Intelligent Systems*, 13(6):40–48, 1998.

[6] D. Gavrila. Multi-feature template matching using distance transforms. In *International Conference on Pattern Recognition*, pages 439–444, Brisbane, 1998.

[7] D. Gavrila. The visual analysis of human movement: A survey. *Computer Vision Image Understanding*, 73(1):82–98, 1999.

[8] D. Huttenlocher, G. Klanderman, and W.J. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993.

[9] S. Kirkpatrick, Jr. C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1993.

[10] C.F. Olson. A probabilistic formulation for hausdorff matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, Santa Barbara, CA, 1998.

[11] C.F. Olson and D.P. Huttenlocher. Automatic target recognition by matching oriented edge pixels. *IEEE Transactions on Image Processing*, 6(1):103–113, January 1997.

[12] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 193–199, San Juan, 1997.

[13] D.W. Paglieroni, G.E. Ford, and E.M. Tsujimoto. The position-orientation masking approach to parametric search for template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(7):740–747, 1994.

[14] P.L. Rosin and G.A.W. West. Salience distance transforms. *GMIP*, 57(6):483–521, November 1995.

[15] W. Rucklidge. Locating objects using the hausdorff distance. In *International Conference on Computer Vision*, pages 457–464, 1995.