Advanced Computer Vision Techniques with Application to Three-Dimensional Multi-Object Tracking

Emmanouel Liodakis

March 2010

Abstract

Modern object recognition and tracking programs, while effective in recognizing a single object, consistently fail when assigned to tasks dealing with multiple objects. Case-specific training sets and the hand-tuning of variables prevent such programs from being truly generalizable. In an attempt to overcome this inherent flaw, I have implemented a supervised learning algorithm that is neither limited by the choice of objects nor by programmable parameters. Through the use of Histograms of Oriented Gradients, the algorithm recognizes and tracks random objects. It is trained for each object on only a small number of images - approximately 500 still-image photographs. HOG is a robust feature set for shape based object detection. Classification and tracking efficiency is also maintained through the use of advanced implementations of both non-maximum suppression and particle filtering. Probabilistic modeling increases the likelihood of true positives and therefore increases accuracy. Increased classification accuracy allows the algorithm to be both theoretically interesting and a practical alternative to current recognition and tracking techniques.

Contents

1 Introduction

The last decade has seen substantial research go into object recognition and tracking algorithms. Many state-of-the-art classifier programs, while effective in recognizing a single object, consistently fail when generalized to tasks concerning multi-object recognition. Case-specific training sets and the fine tuning of variables prevents these tracking programs from classifying random sets of objects. In an attempt to overcome this inherent flaw, a new tracking algorithm was created that both effectively classified data and generalized to any random set of objects. The algorithm could be trained on any data-set and all hand-tuned variable were minimized and replaced by probabilistic models.

In order to test the effectiveness of the algorithm it was asked to identify and label all occurrences of 5 objects - mugs, staplers, keyboards, clocks, and scissors - in random video clips. Sets of 500 still images would be used to train the algorithm to recognize each of the objects.

Given the complexity of computer vision algorithms, the program was broken into sub-tasks each of which was implemented and tested separately. The sections to follow describe in detail each of these sub-process implementations.

2 Hypothesis

It is hypothesized that the object classification algorithm can effectively categorize multiple objects in a video frame with an accuracy of over 95 percent.

3 Materials

OpenCV Library

Object Data Set (500 Images)

Open Source C++ Compiler

4 Histograms of Oriented Gradients

4.1 Boosted Tree Classifiers

Boosted Decision Trees are often regarded as one of the best supervised learning algorithms in computer vision (Freeman and Roth, 24). In order to create an effective multi-object classifier, Boosted Trees were used as the backbone of the tracking program. In contrast to other algorithms, such as Logistic Regression, Boosted Trees provide a generalizable platform. When the Logistic Regression classifier was implemented to test the effectiveness of Boosted Trees the program required a number of 'hacks' to dynamically update the learning rate, based on the change in error rate between iterations. Its accuracy varied widely based on both the hand-tuned variables and the number of training examples used. Moreover, the Boosted Trees classifier was effective in choosing relevant template features for the different objects. This added benefit made Boosted Trees both the most reasonable and effective algorithm for the designated task. In order to not replicate already existent code, CVBoost was used to implement the Boosted Tree portion of the classifier. CVBoost is the boosted tree implementation in the open-source OpenCV library.

4.2 Template Dictionary for all Object Types

In order to effectively generalize the object recognition program, the algorithm needed to generate its own template dictionaries. These dictionaries would serve as a data-set to train the Boosted Trees. Thus, for each object a seperate sliding window would be used, where the aspect ratio of the sliding window was the same as the aspect ratio of the training images of the object. Since each object used a different sized sliding window, each object had its own seperate template dictionary.

Initially a naive approach was used to get the template dictionary for each object. The procedure was as follows:

Procedure

1) Take each labeled image, and resize it down to the size of the corresponding sliding window.

2) Crop out 10 random fragments from each image. The dimensions of the fragments were randomly picked between 4 and half the corresponding dimension of the sliding window. This gave us a feature dictionary with about 2000 fragments.

3) Use CvBoost trees to pick the most relevant subset of feature templates. Train the CvBoost tree with the 2000 fragment template dictionary. Each Cv-Boost tree used 100 decision trees, each of depth 4.

4) Create a feature dictionary by picking all the features used in the nodes of the individual decision trees within each CvBoost tree.

The naive implementation of the template dictionary successfully categorized objects a quarter of the time. These results, however, were not as effective as single-object classifiers and thus a different training set was needed. For a significant performance boost, we would

require a more powerful category of features. In hopes of finding such features, it was decided to implemented Histograms of Oriented Gradient. While not thoroughly tested, current research has suggested that they were very descriptive when used as features.

4.3 Histograms of Oriented Gradients

Histograms of Oriented Gradients are a robust feature set that deliver state of the art performance for shape based object detection (Kanade and Schneiderman, 163). Object appearance and shape can often be characterized rather well by the distribution of local intensity gradients i.e. edge directions. This is implemented by dividing the image window into small spatial regions called cells, and for each cell accumulating a local 1-D histogram of gradient directions over the pixels of the cell. The following procedure was used to calculate this feature set for each object:

1) At each pixel, compute derivatives (i.e. gradients) along the directions of the X and Y axis. We used the CvSobel() function with the following convolution kernels: [-1 0 1] and $[\text{-}1\ 0\ 1]^T$.

2) At each pixel compute:

Direction of the gradient = $\tan^{-1} dy/dx$

Magnitude of the gradient = $\sqrt{dx^2 + dy^2}$

Where dy and dx are the derivatives in the x and y directions respectively.

3) For each input image, we maintained a list of 5 images corresponding to each bin of the histogram. We used 5 bin histograms to capture gradient direction, discretizing the angle space from 90 to -90 degrees into bins of 36 degrees each. Let us refer to these images as 'Bin-Images'. These 'Bin-Images' were initialzied to be 0s at each pixel. Then, as we compute the gradient at each pixel of the input image, we update the same pixel of the

corresponding 'Bin-Image' with the magnitude of the gradient.

4) Computing the Histogram for each Cell: Each Cell is merely a float array of size 5 (where 5 is the number of histogram bins). For each Cell, iterate through the pixels in the Cell and append the histogram values (from the 'Bin-Images') into the corresponding bins of the Cell.

5) Block Normalization: We used 2x2 Cell Blocks. For each Block, we appended the histograms of the Cells constituting the Block and then normalized these values using the CvNormalize function from OpenCV. Note that the blocks are overlapping i.e. each Cell is a part of 4 Blocks (except for the Cells along the edges). Thus each Block has a feature vector of size 20. The final feature vector is created by concatenating the feature vectors of all the Blocks in the Detector Window.

4.4 Exploration of Parameter Space

1) Number of Histogram bins: We began experimentation with 9 bins. This is the number most researchers concerned with HOG have used in the past (descretizing the space from 90 to -90 degrees into bins of 20 degrees each). However, during testing we obtained similar results with 5 and 9 bins. Intuitively it might seem like 9 bins will help us capture the shape of objects better. However, 5 bins was sufficient to capture the shapes we had to recognize. Using 5 instead of 9 bins, however, reduced the size of the feature vector almost by half. The reduction in feature vector size led to a significant decrease in the Boosted Tree training time. Since there was no loss in performance, but a gain in training efficiency, we decided to choose 5 bins per histogram.

2) Gradient masks: We tried 3x3 Sobel masks that performed Gaussian blur and differentiation. We also tried masks that only performed differentiation (3x1 and 1x3 masks). Both delivered identical performance.

3) Performing Block Normalization vs Normalizing the Cell Histogram Vector: Gradient strengths vary over a wide range owing to local variations in illumination and foreground-background contrast. Hence, a critical part of the HOG algorithm is Block Normalization, which achieves effective local contrast normalization. We saw a 38% increase in performance by using Block Normalization.

We tested our code with and without Block Normalization:

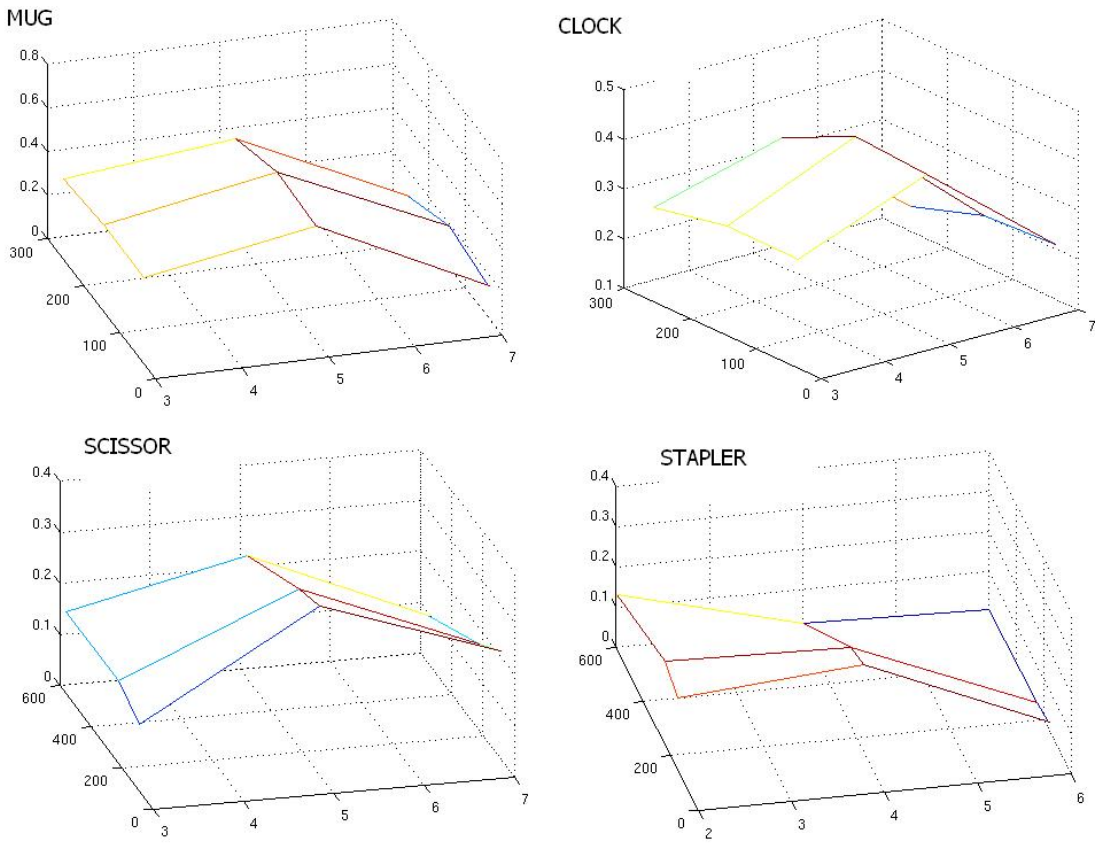
Case 1: Create the feature vector by concatenating the histogram values for each Cell. Normalize this feature vector using CvNormalize.

Case 2: Compute Block features as described in 'Outline of Procedures' step 5.

With Block Normalization we saw a higher number of True Positives and False Positives. The increase in False Positives negated the increase in True Positives. However, our hope was that Non-Maximal Suppression and Temporal Reasoning (the other two significant implementations within the algorithm) would help reduce the number of False Positives and thus increase the effectiveness of the algorithm. Block normalization also lead to a feature vector that was about 4 times the size of the feature vector obtained in case 1. This increase in the size of the feature vector increased the boosted trees training time.

4) Cell Sizes and Number of Cells: For each object type, we tried about 3 different cell sizes, and 3 different values for the number of cells along each dimension. Figure 1 depicts the percentage of incorrect identifications obtained for different values of the above parameters. The x-axis is Cell Width. The y-axis is number of cells. z-axis is percentage of incorrect guesses. (NOTE: Each Cell had equal number of rows and columns.)

Figure 1



5 Non Maximal Suppression

5.1 Design

The object recognition algorithm classifies objects by running a 'scanning window' detector across each image at a dense grid of locations and scales. This type of detector regularly produces a large number of positive identifications close by the correct detection. This makes it necessary to use non-maximum suppression to thin out multiple responses and to remove spurious positive detections (Lowe, 95).

Basic non-maximum suppression involves grouping object predictions into equivalence

classes that are based mainly on closeness. Prediction boxes are removed either when they overlap each other within a specific group by a predetermined threshold or when the group does not contain a threshold number of predictions. As may be expected by this type of algorithm, you produce a large number of threshold and tolerance parameters which need to be hand-tuned (Gil and Werman, 505).

The inherent limitations associated with these type of non-maximum suppression algorithms motivated the implementation of a more principled and complete approach. This algorithm was then tested against a common, naive implementation.

5.2 Algorithm

The non-maximum suppression algorithm attempts to model the probability associated with a group of object detections given a single real object. This type of probability distribution can be used to determine the total set of correct predictions by mixing different probability distributions together and maximizing the likelihood of the detections with an Expectation-Maximization algorithm.

Given a set of predictions and knowledge of the actual object we can calculate the probability of each prediction given that object as follows:

$$p_{ssg}(z_j|r) = \frac{1}{(2\pi)^{3/2}\mu_s^3 c_x c_y c_z} \exp\left[-\frac{1}{2\mu_s^2}\left(\frac{(x_j-\mu_x)^2}{c_x^2} + \frac{(y_j-\mu_x)^2}{c_x^2} + \frac{(z_j-\mu_x)^2}{c_x^2}\right)\right]$$

This model is well defined by a scale-sensitive Gaussian as was shown by Oskarsson an dAstrom (dAStrom, 2006). Constants cx, cy, cz used to determine the probability are actually determined in training by looking at the empirical standard deviations between each prediction and the actual object.

We can assume that we observe a random number, N, of detections per frame. We know

these detections either represent real objects, repeated detections of objects, or spurious background detections. We are thus able to determine an effective mixture model using the following equation:

$$L(\theta_k|D) = \sum_{j=1}^{N} \log\left(\sum_{i=1}^{K} \alpha^i p_{ssg}(\triangle z_j^i) + \alpha^0 p(z_j|bg)\right)$$

Assuming that we know the number of objects in each frame and that the distribution is modeled by a Scale-Sensitive Gaussian we can predict where each of the real objects is. We do this with an Expectation-Maximization algorithm.

Expectation Step: We determine the posterior probabilities for each real object to have generated each predicted object.

Maximization Step: Given these probabilities we update all necessary values and determine the likelihoods.

Since this is an EM algorithm, when it begins to converge you can break about of the loop. Given this, all we have left to predict is the correct number of faces we need to maximize the following equation for a set of different possibilities regarding the number of objects in the frame:

$$J_k = L(\theta_k^*|D) - \eta P$$

The maximum likelihood is determined above and P denotes the number of free parameters and is thus equal to 4K, where K is the number of faces.

Figure 2

Pseudo Code:

```
For Different Numbers of Faces {
        Initialize Variables
        while (Current Likelihood Past Likelihood  Minimum Threshold {
            Calculate posterior probabilities
            Update Prediction Parameter
        }
        Calculate Maximum Likelihood Prediction Given Updated Parameters
}
Take Maximum Likelihood;
```

5.3 Data

In order to test the enhanced non-maximum suppression algorithm described earlier it was run against a simpler algorithm which averages two predictions if they have a specific amount of overlap by taking the average values of the x and y locations and the width and height. To maintain consistency between tests, I ran both non-maximum suppression algorithms on the same classification video.

Figure 3

Test 1		
Without-Maximum Suppression	Algorithm 1 (Simple)	Algorithm 2
true-positives: 29	true-positives: 10	true-positives: 29
false-positives: 121	false-positives: 84	false-positives: 55
false-negatives: 10	false-negatives: 10	false-negatives: 10

Test 2		
Without-Maximum Suppression	Algorithm 1 (Simple)	Algorithm 2
true-positives: 26	true-positives: 22	true-positives: 27
false-positives: 144	false-positives: 74	false-positives: 67
false-negatives: 13	false-negatives: 17	false-negatives: 12

Test 3		
Without-Maximum Suppression	Algorithm 1 (Simple)	Algorithm 2
true-positives: 21	true-positives: 20	true-positives: 21
false-positives: 135	false-positives: 59	false-positives: 45
false-negatives: 18	false-negatives: 19	false-negatives: 18

In each test we see that the advanced non-maximum suppression algorithm surpassed the simpler algorithm in effectiveness. By design, the simpler algo- rithm did not remove spurious background data. Its averaging also saw a decrease in true positives. By averaging predictions, prediction boxes were moved from images and the object stopped being classified.

However, the novel non-maximum suppression algorithm used the Scale-Sensitive Gaussian to accurately fine tune where the predictions were. The algorithm removed random, inaccurate background images and also correctly averaged clustered prediction groups. Instead of averaging guesses it used the this proven probabilistic model to give the most reasonable prediction. These two clear advantages made it a superior choice to regular non-maximum suppression.

6 Multitarget Particle filter

6.1 Design

Additionally, a temporal algorithm was implemented to capture sequential patterns within the data. Particle filters are applied to determine hidden variables within stochastic dynamic systems detected by observation processes. The difficulty lies in the fact that the estimation of the states requires the assignment of the observations to the multiple targets. Further testing the discovery of J-P. Le Cadre, C. Hue, and P. Perez, a version of their particle filter was used to fully implement this portion of the classifier algorithm. Specifically, an extension of the classical particle filter where the 'stochastic vector of assignment is estimated by a Gibbs sampler' (Cadre, Hue, Perez, 803). The particle filter was used to estimate the trajectories of multiple targets from their noisy bearings, thus showing its ability to solve the data association problem in the object tracking task. The algorithm was extended here to the estimation of multiple state processes given realizations of several kinds of observation processes and was used to track with success multiple targets in a bearings only context.

The implementation of the Multitarget particle filter allowed the model to capture temporal sequences which ultimately allowed it to better predict the actual number of objects present (Comaniciu and Ramesh, 72). This model was implemented on top of the Boosted Trees. To make the model efficient, we computed during training the variance matrix for the distribution of predictions given an object by testing the model on a labeled video.

6.2 Algorithm

Multitarget tracking (MTT) deals with state estimation of an unknown number of moving targets. In order to implement this, it was assumed that the available measurements may both arise from the targets if they are detected, and from clutter. The main difficulty comes

from the assignment of a given measurement to a target model. As soon as the association variables are considered as stochastic variables and moreover statistically independent the complexity is reduced. Under such assumptions, particle filters are particulary adapted. They mainly consist in propagating a weighted set of particles which approximates the probability density of the state conditionally to the observations. For this particular task, which estimates the true location of an unknown number of objects, we have combined the two major steps, prediction and weighting, of the classical particle filter with a Gibbs sampler-based estimation of the assignment probabilities.

Formally, the implemented algorithm does the following:

We considered a dynamic system represented by the stochastic process (X_t), where each X represents the true location of all objects, whose temporal evolution is modeled by some function of its previous state and a random normally distributed variable. It is observed at discrete times via realizations of the stochastic process (Y_t) where Y is a vector representing our guess as to the number and locations of objects. We assume Y to be a function of X and a normally distributed error variable. We will denote by $Y_{0:t}$ the sequence of the random variables $(Y_0, ..., Y_t)$ and by $y_{0:t}$ one realization of this sequence. Our problem consists in computing at each time t the conditional density L_t of the state X_t given all the observations accumulated up to t, i.e., $L_t = p(X_t|Y_0 = y_0, ..., Y_t = y_t)$ as well as estimating the expectation $E(X_t|Y_{0:t})$. The Recursive Bayesian filter resolves exactly this problem in two steps at each time t. Suppose we know L_{t-1} . The prediction step is done according to the following equation: $p(X_t = x_t|Y_0 = y_0, ..., Y_{t-1} = y_{t-1}) = p(X_t = x_t|X_{t1} = x_{t-1})L_{t-1}(x)dx$. The observation y_t enables us to correct this prediction using Bayess rule: $L_t(x_t) = p(Y_t = y_t|x_t)p(X_t = x_t|Y_0 = y_0, ..., Y_t = y_t)/\int(p(Y_t = y_t|x_t)p(X_t = x_t|Y_0 = y_0, ..., Y_{t-1} = y_{t-1})dx$. The particle filter approximates the densities (L_t) by a finite weighted sum of N densities centered on elements in the sample space of (X_t), named particles. To implement the prediction, we made the following assumptions:

We assumed the initial prior marginal $p(X_0)$ was given by a multivariate gaussian distribution around the first set of observations. We assumed that the observations arose from either an actual object or from the clutter, and that if they originated from an actual object they were distributed according to a gaussian distribution around the true location of that object. To sample from the distribution X_t , given x_{t-1} , y_t , we first used Optical flow to calculate the mean state s of x_t.

The algorithm then consists in making evolve the particle set $S_t = (s_{n,t}, q_{n,t}), n = 1...N$ where $s_{n,t}$ is the nth particle and $q_{n,t}$ its weight, such that the density L_t can be approximated using the density of S_t . To avoid the degeneracy of the particle set, i.e. only few particles with high weights and the others with very small ones, a resampling is done in an adaptive way when the number of effective particles, estimated by N_{eff} , is under a given threshold. Besides the discretization of the filtering integrals, the use of such particles enables to voice many hypothesis on the position of the object and to keep in the long term only the particles whose position is likely given the sequence of observations.

6.3 Experimentation

The first experimentation performed using the particle filter consisted of varying the variance matrices. We first estimated a standard deviation of 4 pixels (since we were sliding the frame 4 pixels) for the width, height, x and y location. We were able to slightly improve this by varying it to 2.4 pixels. Ultimately, however, our strongest gain was by experimentally figuring out the true variance in these values, which gave us a variance matrix that achieved slightly better scores.

The second experimentation consisted of varying the threshold for N_{eff} - the value used to predict whether a particle set is effective. Ultimately, we found that the threshold had a large impact on the performance. Indeed, N_{eff} is computed as the sum of squares of weights,

and we calculated 1.9 to be the best threshold for our value of n - we had 100 particles. The table of results is reproduced below:

Figure 4

	Baseline Model	Particle Filter	Best Variance	Best Threshold for N_{eff}
true-positives	20	27	28	28
false-positives	106	95	97	94
false-negatives	19	12	11	11

The multitarget particle filter interestingly did not drastically reduce the number of false positives. The reason for this, as touched upon in the design section of this paper, is that the implementation of the particle filter uses a gaussian distribution to estimate the probability of an observation given an object. The actual variance computed in training, however, was such that in most cases if the observation and object were far enough away (estimated to be around half the frame), then then the probability of that object having generated the observation were computed as zero - the exponenent of a large enough negative number is returned as zero. The model thus predictably guesses an object for observations far enough away from other observations, since the probability that an observation comes from the clutter is often calculated to be quite low.

The strength of the particle filter, however, is evident in the number of true positives guessed. Because the particle filter uses optical filter methods when assigning the probability of an objects location given its past location, it performs remarkably well in correctly predicting objects even when few predictions have been made in a particular frame. It also correctly predicts the location of objects when there are a few observations slightly off from the object. Ultimately, the temporal components of the algorithm allow it to significantly increase the number of true positives.

7 Conclusion

The algorithm was able to successfully generalize over a large set of objects without losing single-object classification effectiveness. While Boosted Trees have been implemented in a variety of different computer vision problems, additional focus on training generalization allowed for a novel approach to computer vision. Theoretical extensions concerning both the feature dictionary and non-maximal suppression helped to generalize the algorithm and programmatic extensions dealing with the Cadre, Hue, and Perez particle filter helped to maintain effectiveness. The algorithm was effectively able to categorize and track any object given only five hundred still-image photographs of the object and twenty minutes of training time.

8 Bibliography

Belongie, S. and Malik, J. and Puzicha J. 'Matching shapes.' The 8th ICCV, Vancouver, Canada, pages 454461, 2001.

Beymer, D. and McLauchlan, P. and Coifman, B. and Malik J., A Real-time Computer Vision System for Measuring Traffic Parameters, Computer Vision and Pattern Recognition (1997), pp. 495501.

Black, M. and Jepson A., 'A Probabilistic Framework for Matching Temporal Trajectories: Condensation-Based Recognition of Gestures and Expressions, European Conference on Computer Vision' (1998), pp. 909924. Full Text via CrossRef

Burges, C. 'A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery' (1998), pp. 121167.

Chow, C., and Liu, C. 'Approximating discrete probability distributions with dependence trees.' IT(14), No. 11, November 1968, pp. 462-467.

Comaniciu, D and Ramesh, V. and Meer P., 'Real-Time Tracking of Non-Rigid Objects using Mean Shift, Computer Vision and Pattern Recognition' (2000), pp. 142149.

Freeman, W. and Roth, M. 'Orientation histograms forhand gesture recognition. Intl. Workshop on Automatic Face- and Gesture- Recognition', IEEE Computer Society, Zurich, Switzerland, pages 296301, June 1995.

Friedman, N. and Russell, S. 'Image Segmentation in Video Sequences: A Probabilistic Approach, Uncertainty in Artificial Intelligence' (1997), pp. 175181.

Gil, J. and Werman, M. 'Computing 2-d min, median, and max filters.' IEEE Trans. Pattern Anal. Mach. Intell., 15(5):504507, 1993.

Lowe D. 'Distinctive image features from scale-invariant keypoints.' IJCV, 60(2):91110, 2004.

Mikolajczyk, K. and Schmid, C. 'Scale and affine invariant interest point detectors.' IJCV, 60(1):6386, 2004.

Pitt, M. and Shephard N. 'Filtering via Simulation: Auxiliary Particle Filters' Journal of the American Statistical Association, Vol. 94, No. 446 (Jun., 1999), pp. 590-599

Poortere, V. and Cant J. and Van den Bosch, B. 'Efficient pedestrian detection: a test case for svm based Workshop on Cognitive Vision', 2002. categorization.

Ronfard, R and Schmid, C. and Triggs, B. 'Learning to parse pictures of people.' The 7th ECCV, Copenhagen, Denmark, volume IV, pages 700714, 2002.

Schneiderman H. and Kanade T. 'Object detection using the statistics of parts.' IJCV, 56(3):151177, 2004.

Schwartz, E. 'Spatial mapping in the primate sensory projection: analytic structure and relevance to perception.' Biological Cybernetics, 25(4):181194, 1977.

Ullman, Shimon. 'High-Level Vision: Object Recognition and Visual Cognition.' 1 ed. London: The Mit Press, 2000. Print.

Viola, P. and Jones, M. and Snow, D. 'Detecting pedestrians using patterns of motion and appearance.' The 9th ICCV, Nice, France, volume 1, pages 734741, 2003.

Yali, A. '2D Object Detection and Recognition: Models, Algorithms, and Networks.' illustrated edition ed. London: The Mit Press, 2002. Print.