Tracking-Based Semi-Supervised Learning using Stationary Video

Andrew Chou
Stanford University
akchou@stanford.edu

Alex Teichman
Stanford University
Project Mentor
teichman@cs.stanford.edu

Abstract

This paper deal addresses the semi-supervised problem of tracking and recognizing objects in videos taken with stationary cameras. Building on work on Stanford's autonomous vehicle using laser range finders to solve the same problem, this paper aim to develop accurate methods for classifying objects without the additional benefit of 3D laser scans. We set out with three main goals, each building on the previous ones. The first is to perform background subtraction to remove all background objects (those objects that are stationary in the frame of the camera). The second is to track the foreground objects through every frame of the video. Finally, the third goal is to use semi-supervised methods to classify tracked foreground objects. A successful semi-supervised approach will greatly reduce the amount of training data needed for many classification problems.

1. Introduction

The objective of this paper has three subgoals:
(1) Remove background objects.
(2) Track foreground objects.
(3) Classify foreground objects using
semi-supervised learning. The combination of these three goals will allow us to train a complex classifier with very little manually labelled data. Each of the three steps in our Semi-Supervised learning method build on previous work as described below.

1.1. Removal of Background Objects

The background removal stage requires a video, or sequence of images, as input and outputs a binary mask for each frame or image. The ones in a mask (displayed as white in this paper) represent foreground pixels in that mask, while the zeros in a mask (displayed as black in this paper) represent background pixels in the mask. In this context foreground objects have the property that they are capable of movement outside of some fixed region that is roughly the size of the object. For example, humans, bicylists, automatic vehicles, and animals are some of the things that are considered to be foreground objects. Conversely, most inanimate objects such as buildings, plants, benches, and poles should ideally be classified as background. More subtle objects obejcts that should be classified as background are bodies of water, fans, fountains, and trees swaying in the wind. Distinguishing these moving background objects from real foreground objects is at the heart of the problem.

1.2. Foreground Object Tracking

The goal of the foreground object tracking step is to take the foreground masks from the background removal step and determine which foreground objects in each mask correspond to objects in other masks. The output should contain labels for each object as well as an outline for the object at each frame. Ideally the foreground object tracking step should be robust to overlap of objects, false positives (Labelling part of a fountain as foreground, for example.), false negatives (failure to detect a foreground object for a few frames), extrance and exit of objects during the sequence, as well as total number of objects needing to be tracked.

1.3. Semi-Supervised Classification

Classification is done in a semi-supervised way using methods already developed in a previous paper by Teichman and Thrun [7]. Results and classification accuracy will be quantitatively evaluated using a data set provided by Alex Teichman.

1.4. Background Subtraction

Standard methods for modelling the background include a Mixture of Gaussians method. OpenCV's implementation of Mixture of Gaussians for background subtraction does reasonably well when tested on a scene with swaying trees and fountains [3]. However there are a significant number of spurious points in the tree and fountain that are not adequately modelled by the Mixture of Gaussians. Sheikh and

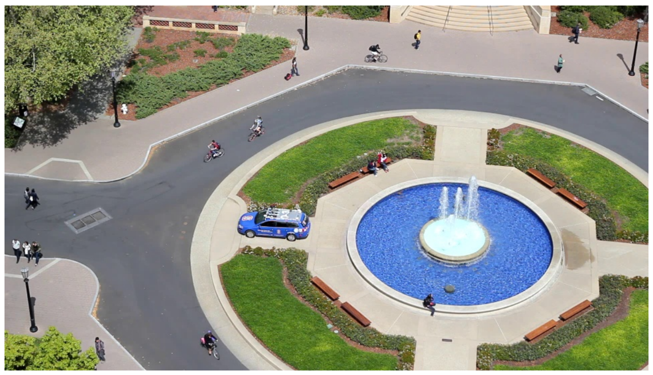
Figure 1. A sample frame from one of our test videos.

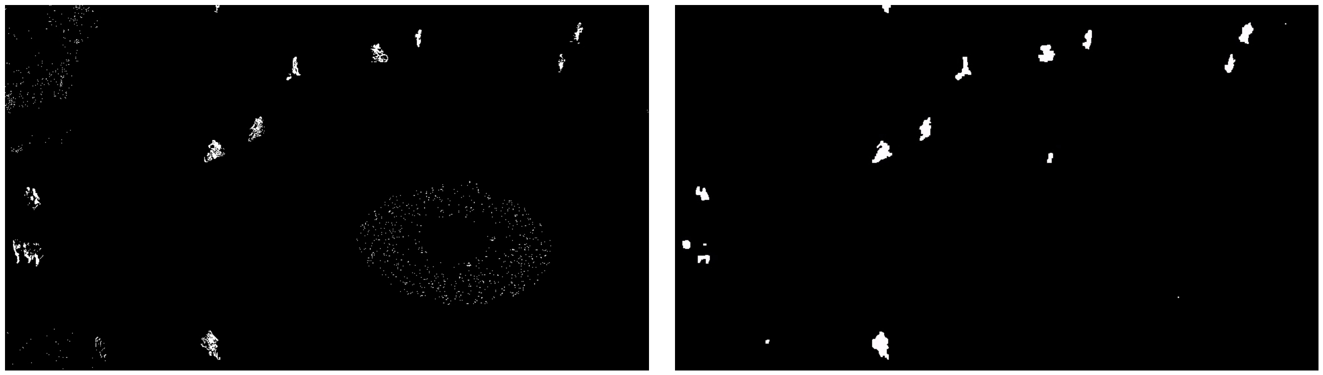
Figure 2. (left): A foreground mask using the Mixture of Gaussians method from OpenCV. (right): The equivalent foreground mask using Shah *et al*.

Shah use a Bayesian Modelling method to reduce the error in the subtraction [6].

1.5. Competing Bayesian Models Method

Shah *et al*.'s method handles cases, such as the one with trees fountains, with non-stationary backgrounds. They use competing Bayesian foreground and background models at each pixel to determine which pixels are foreground and which are background. The models are discretized in the five dimensional RBGXY space. Thus pixelwise similarity is determined by both appearance and location in the image. Each model starts with a uniform prior distribution across the RGBXY feature space. The background distribution is favored slightly in the assumption that the majority of pixels are part of the background. At each frame the pixels are classified as either foreground or background depending upon the likelihoods of the models. Pixels classified as background are added to the background model, and pixels classified as foreground are added to both models. This allows mistakenly classified foreground pixels to slowly fade into the background.

Each model has a specified history length to make them more robust so that the background and foreground become relatively stable over time. This is a key advantage of Competing Bayesian Models over other methods because the foreground detections are dependent upon past frames and are thus more consistent from frame to frame. This allows pixels that look similar to the background to still be classified as foreground if there have been similar foreground pixels nearby in the recent past. For example, this could be useful if a person wearing a green shirt is tracked across pavement, but then walks in front of trees. Without the foreground model the person would be lost. It is also important that the foreground history be shorter than the background history so that when foreground pixels are added to both the foreground and background models they will have a larger effect upon the foreground model.

The histories that used in this paper were fairly short (100 frames for the background and just 5 frames for the foreground) in order to make the algorithm robust to slowly changing backgrounds. This feature turns out to be very important to compensate for lighting changes such as a those caused by passing clouds on outdoor scenes. The histories associated with each model also allow the foreground detections to be consistent over time.

The most significant improvement of Shah *et al*.'s

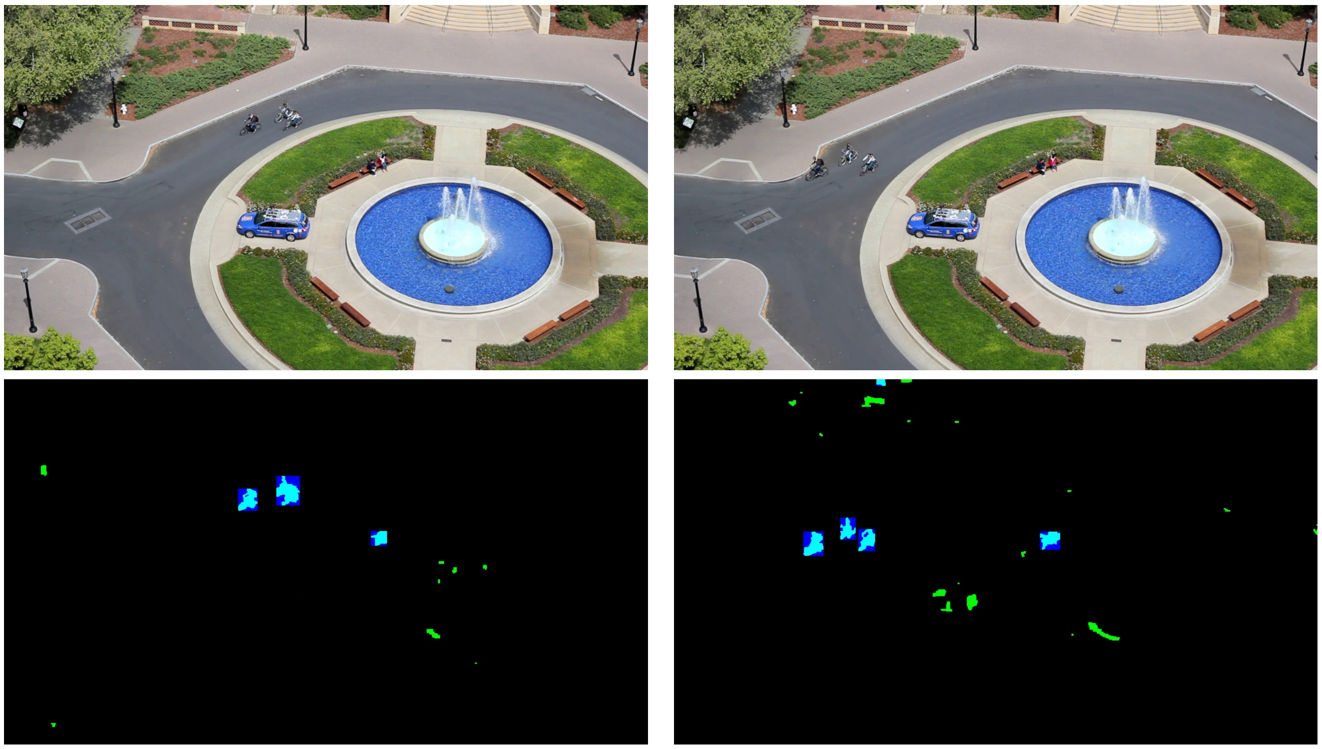

Figure 3. (top left): A frame from a test video in which one biker is occluded by another biker. (bottom left): The output of a variant on Berclaz *et al.*'s k-paths algorithm overlaid on top of the output from Shah *et al.*'s background subtraction. Unfortunately the two overlapping bikers are detected as a single object using a connected regions algorithm. (top right): Another frame from a slightly later point in the test video. (right): The combined output of kpaths and background subtraction. Note that while the bikers could not be distinguished while one was occluded by the other, they are immediately split into two separate detections once they are separated. This is a result of a modification to the kpaths algorithm that allows an object to appear anywhere in a scene and at any time, but at a very high cost. However, resolving this issue in its entirety is beyond the scope of this project.

method of other similar methods is its secondary graph cut step. In this step each pixel represents a node in a weighted graph, with the addition of two extra nodes; one representing the background and one representing the foreground. The graph has weighted edges between each pixel node and the foreground and background nodes, with weights dependent upon the probabilty of each pixel being in the foreground or background based on the competing models in the first step. Finally, each pixel node has weighted edges to each of its four neighboring pixel nodes. Then a minimum graph cut is performed to determine which nodes are connected to the background node and which are connected to the foreground node. This approach ensures that neighboring pixels will be grouped together unless there is very strong counter-evidence.

1.6. Comparison with Other Methods

A qualitative evaluation of the results of both the Mixture of Gaussians approach and Shah's Bayesian Modelling method showed that Shah's approach yields many fewer spurious foreground points. Furthermore, Shah's approach has segmentation built into the algorithm because the min-cut algorithm that it uses tends to group spatially proximal pixels. In contrast, Mixture of Gaussians detects differences generally along object edges, so a secondary method would be needed in order to cluster the points detected as foreground. We tried low-pass filtering the Mixture of Gaussian output with a square kernel in conjunction with thresholding, however the results were qualitatively not as clean as the results from Shah *et al.* We also tried dilation but had similarly poor results. Thus Shah's method was chosen as the background subtraction step for the method described in this paper.

2. Computational Challenges of Background Subtraction

When computing the competing foreground and background models, it is computationally too expensive to calculate a Gaussian distribution around each point the 5 dimensional space and for each pixel. Instead, this paper uses an approximation that gives the bucket that the pixel is in a high likelihood, while the 80 neighboring buckets in the five-dimensional space get slightly lower likelihoods. All other buckets are unaffected by that particular pixel. Also, Shah's original paper used frames that were 240x360 pixels. They were able to process 11 frames per second using a 3.06 GHz Intel Pentium 4 processor with 1 GB RAM. The frames that used in this project are 960x544 pixels, so in order to achieve a comparable level of performance the competing models portion of the algorithm was parallelized with up to 18 threads. Ultimately this method is able to process roughly 5 frames per second using a 2.7GHz dual core i7 with 4GB of RAM.

3. Foreground Object Tracking

In our foreground object tracking phase we would ideally be able to distinguish each object from every other object as well as retain knowledge about the identity of each object as we move from frame to frame of the video. Given just the foreground mask for each frame and the original frames themselves it is difficult to separate overlapping objects without a class model. This separation process is beyond the scope of this paper, so for now we assume that the objects we are tracking are non-overlapping.

3.1. K-Shortest Paths Object Tracking

Since we need to be able to track an arbitrary and constantly changing number of objects, we use a variant on the K-Shortest Paths foreground object tracking method developed by Berclaz *et al.* [2]. Most similar methods require a fixed number of objects. Berclaz *et al.* assume the use of an appearance model to help track objects. As we do not have such a model, and are in fact trying to train a similar classifier, we will use our background subtraction method to give us probabilities of a foreground object at each location and time.

The Berclaz method discretizes the image into buckets (we use buckets of 10x10 pixels) each of which represent a node in a directed acyclic graph (DAG). Each frame is a layer of the DAG and there are two extra nodes; a source and a sink node. Directed edges are connected from each bucket to its 9 neighboring bucket (including itself) in the next frame. The source node has an edge going to each node in the first frame and to every node on the border of every frame. Similarly, the sink has edges coming in from the borders and the last frame. Each edge (u,v) is weighted according to the probability of occupancy at node u.

Some edges (the edges that have an occupancy probability of greater than one half) have negative weights, so Bellman-Ford is used (at least in our implementation) to find a minimum cost path from source to sink [1]. When a minimum cost path is found then all of its nodes and edges are removed from the graph and the next minimum cost path is found. This process is repeated in order to find an arbitrary number of paths. Once no more paths have a cost below some predetermined threshold then the algorithm stops. See [2] for more details.

3.2. Occlusion

In an attempt to solve the overlap problem we used a low pass filter on the location so that two intersecting paths can still be made continuous by linking them through a lower probability region. However this lowered the probability of many single tracks and resulted in too many lost paths to be useful.

This paper instead focuses on making sure the objects are tracked in separate bounding boxes once they are no longer detected as part of the same connected region. In this vein a partial solution to the occlusion problem can be achieved by allowing objects to appear and disappear anywhere in the kpaths DAG at anytime. This is achieved by connecting both the source and sink nodes to every other node in the kpaths graph.

However, since it is unrealistic for objects to continually pop into and out of existence a very high cost is placed on the extra edges to the sink node. This ensures that the algorithm must have extremely high confidence in an object's existence from the surrounding frames before allowing an object to spontaneously appear or disappear. Of course there is still no cost associated with the appearance or disappearance of an object along the border of a frame or at the start of end of a video sequence. There are no extra openings in the video sequences tested in this project. However if a video sequence were to have known doors or other openings that objects could realistically appear and disappear from then it would be important to allow nearby nodes to connect to the source and sink nodes at little or no cost.

3.3. Object Tracking Performance

In practice it is too expensive to encode a long video as a single graph, so this paper uses video segments with 100 frames each and then links the paths between segments. Detections at the start of each segment are only allowed to appear for free near the endpoints of objects tracked in the previous segment. When segments of 100 frames are used then kpaths can run in near real time speeds on a 2.7GHz dual core i7 with 4GB of RAM. Of course this doesn't include the 100 frame lag needed to allow the frames to accumulate.

4. Semi-Supervised Classification

As mentioned above, classification is done in a semi-supervised way using methods already developed in a previous paper by Teichman and Thrun [7]. This method starts

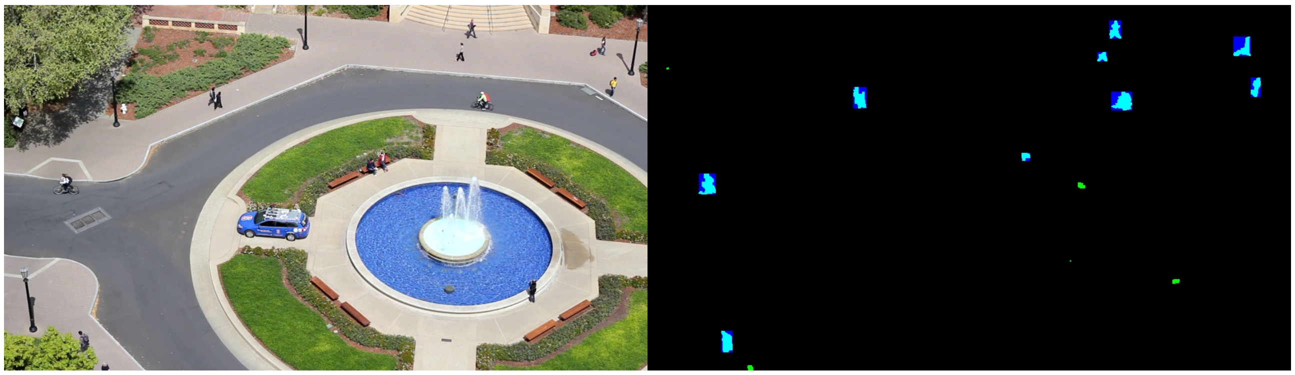
Figure 4. (left): A sample frame from one of our test videos. (right): The output of a variant of Berclaz *et al.*'s k-paths algorithm overlaid on top of the output from Shah *et al.*'s background subtraction.

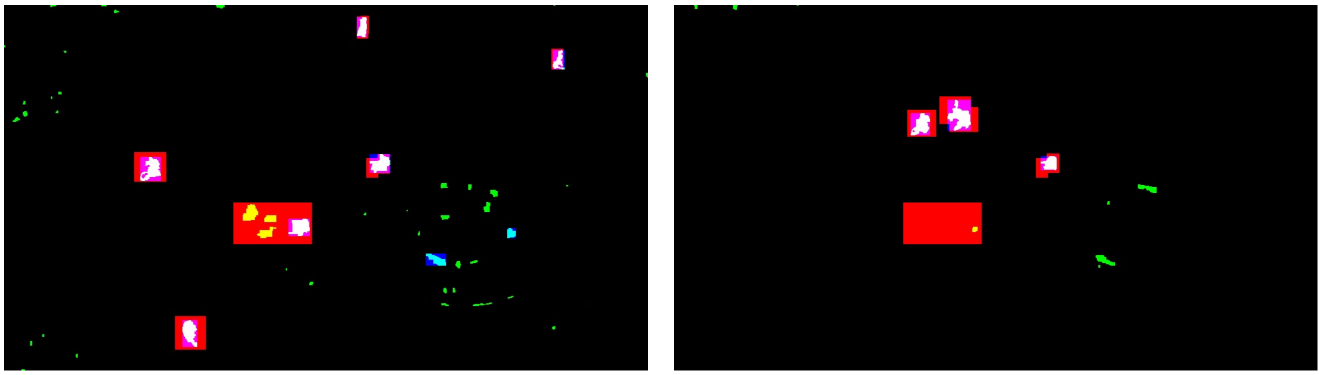
Figure 5. These are frames from the output of Shah *et al.*'s background subtraction overlaid with the handlabelled bounding boxes (red) and K-Shortest Paths tracked objects (blue). The boxes appear purple where they overlap.

with a small number of hand labelled video sequences and a large set of randomly sampled background images. It then learns a base classifier and uses it to classify new images individually. Each new image is itself part of a video sequence enabling us to calculate an overall confidence of classifying the sequence as a whole. Confidently classified positive sequences are then added to the set of positive training examples. In this way a few new diverse training examples can be pulled in during each iteration by association with confidently classified frames in the same sequence.

Our base classification method needs to be able to be able to learn to identify object classes only using bounding boxes. We could feasibly use slightly more informative bounding polygons, but their use is beyond the scope of this paper. With such limited information about each object it seems difficult to train a complex model to perform classifications. However, Felzenszwalb *et al.* use a part based method that can be trained using only bounding boxes [5].

4.1. Felzenszwalb's Part Based Model

Felzenszwalb *et al.*'s part based model classifier uses histogram of gradient (HoG) features at multiple levels of an image pyramid to represent each potential detection. In the initial stages of the algorithm a set of root filters (each representing the overall shape of the object when the object is viewed from a different orientation) are trained using stochastic gradient descent on the results of a SVM to give high responses to foreground instances and low responses to background instances. Then a set of part filters for each root filter are learned at a finer level of detail of the image pyramid. Each part filter has a default position relative to its associated root filter. Thus the final score of a detection is a combination of the root filter response, the part filters responses, and a negative term for how far each part filter needs to be shifted away from its default position in order to get a good response. The robustness of the algorithm comes from the multiple root filters that can each give a confidence of detecting the object in a different pose. For details see [5] and [4].

Felzenszwalb's part based model is used here as the base classifier for Teichman *et al.*'s semi-supervised algorithm [7].

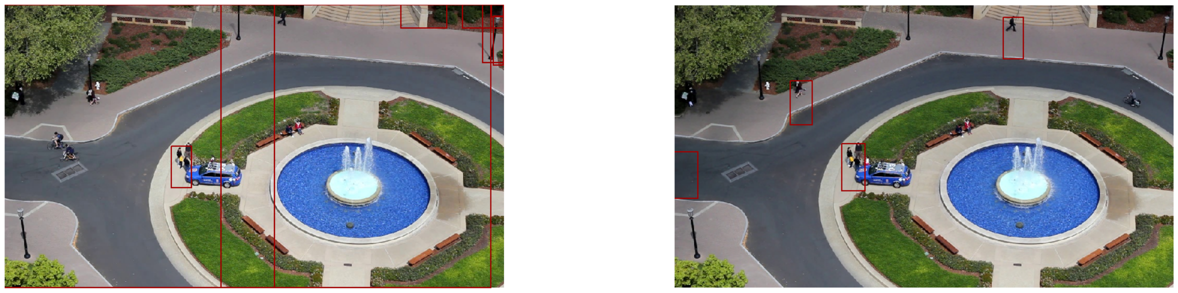

Figure 6. These are frames of the tracks that were added using Felzenszwalb *et al*.'s part based model. On the left you can see that the people were not recognized, but because they were part of a track they were able to be added to the set of positive training examples.

Kpaths	Test 1	Test 2
Detections	3934	2614
Used	3249	2614
Overlap	5339	2614
Missed	285	2197
Accuracy	82.6%	100%
Recall	94.9%	54.3%

Table 1. K-Shortest Paths Results. Detections is the number of total number of objects detected in each test (an object detected in multiple frames is counted multiple times). Used is the number of detections that were matched with hand labelled objects. Overlap is the number of hand labelled objects that matched with a detection. Here a "match" is only declared when the bounding box of the detected object covered greater than 50% of the bounding box of the hand labelled object and the bounding box of the hand labelled object covered greater than 50% of the bounding box of the detected object. Missed is the number of hand labelled objects that were not matched. Accuracy is the percentage of detected objects that were matched and recall is the percentage of hand labelled objects that were matched.

5. Results

We do not have a qualitative method for evaluating Shah's Bayesian Modelling method, but it looks good qualitatively and it works well as the input to our variant on Berclaz *et al*.'s K-Shortest Paths algorithm.

K-Shortest Paths did a good job overall as can be seen in Table 1 and in in figure 5. Felzenszwalb's Classifier worked well adding tracks, but the average precision did not improve. If we had more time we would do more iterations and hopefully the results would improve. However it is notable that the tracks the classifier was able to add contained frames that were initially classified as background. Performance is also an bottleneck in Felzenszwalb's Classifier as

	Original Test	Second Test
Frames	4000	5300
Tracks	22	24
AP	22.6	22.6

Table 2. Felzenszwalb's Part Based Classifier. The original test consisted of 22 hand labelled tracks over 4000 frames. Note that this took about 5 minutes to do because of the tracks. Two new tracks were identified over 1300 frames. However, when they were added they did not increase the average precision (AP) of classifications measured over the 11 recall points from 0%-100% with 10% intervals.

one iteration takes nearly nine hours to run on a modern cluster computer such as Stanford University's Corn Cluster.

References

[1] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1).

[2] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PP(99):1, 2011.

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4. http://people.cs.uchicago.edu/ pff/latent-release4/.

[5] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9), September 2010.

[6] Y. Sheikh and M. Shah. Bayesian modeling of dynamic scenes for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1778–1792, 2005.

[7] A. Teichman and S. Thrun. Tracking-based semi-supervised learning. In *Robotics: Science and Systems*, Los Angeles, CA, USA, 2011.

6. Appendix

The work in this project builds Alex Teichman's work (advised by Sebastian Thrun) that performs the the same object tracking and semi-supervised classification, but while using a laser range finder. This project aims to perform both steps with only video input.