

Cs 600.461: Computer Vision Final Project Report

Giancarlo Troni
gtroni@jhu.edu

Raphael Sznitman
sznitman@jhu.edu

Abstract

Given a Youtube video of a busy street intersection, our task is to detect, track, segment, recognize and count as many moving objects as possible. The task is a difficult one, as the camera is moving, the resolution is low, many objects are moving at once and creating occlusions. We propose to tackle this problem by first rectifying the frames as much as possible, allowing us to build an accurate model of the background and calculating an accurate perspective model. Using background subtraction with some local matching, we detect moving objects. Segmentation is further refined by using opening and closing of detected regions. Tracking is used by using a motion model and a Kalman filter. We recognize tracked objects by the use of simple tests in a hierarchical classification tree. Counting is performed by observing tracked objects moving out of frames. Statistical results are provided for analysis and some final remarks are discussed.

1. Introduction

For this given project, we are presented with a Youtube ¹ video which shows the busy day-time activity on an Indian street intersection. The camera filming the scene is handheld and captures passing people and various vehicles (e.g. Cars, buses,...) by being located above the street level. This two minute long, low resolution video offers a great challenge for computer vision.

Using the various techniques learned in this class, our task is to correctly detect, track, segment, recognize and count as many relevant objects as we possibly can. Contrary to trying to focus on one particular task, our objective is to do all of these to a satisfiable level. In addition, we will put a particular emphasis on counting as we believe that counting accurately, requires most of the tasks described above to be done accurately. Naturally, we are also interested in doing these tasks as correctly and efficiently as possible.

We begin below by presenting an in-depth analysis of the various problems this video displays. A framework is



Figure 1. A typical frame from our given video.

simultaneously presented, partitioning different problems into subtasks. Our solution to these subtasks are then presented with examples at different points. Finally, we show various examples of our results at different stages of the process, as well some statistical results used to evaluate our recognition, detection, tracking and counting methods and we conclude with a few remarks on improvements for future work.

2. Problem Description

The video provided is of great complexity and has many different aspects to it, each deserving close attention. First, the camera is constantly moving and above the street level creating a strong perspective effect throughout the video. There are both small and large camera motions at various points in the video.

Many different types of objects are seen throughout the video. Some of these are cars, buses, people, motorcycles, and so on. These are generally seen with different size, color, and orientations. These objects often move together in groups of objects but also pass in front of each other, creating occlusions.

One should also note that the resolution of the video is of poor quality. Recognition of objects by humans is already

¹<http://youtube.com/watch?v=RjrEQaG5jPM>

non-trivial in certain regions of images and similarly, when objects are close together, recognition of what type of object is present is difficult. The low resolution imposes great difficulty for collecting training data of particular objects types.

2.1. Working Framework

In order to tackle these challenges it is important to fully partition the various issues into subproblems. In our view, the solution to the general problem can be broken into four important sequential subproblems. These are:

- *The creation of a reference Frame:* This an important issue mainly because the camera was hand-held. Thus, a first step is to calibrate frames in order to factor any motion camera. This will provide easier data for detecting objects which are moving, as opposed to appearing to move because of camera motion. In addition, this calibration will facilitate the formation of a background model.
- *Detection of moving objects:* Is vital to tracking and object recognition. That is we are interested in acquiring the center of mass of moving objects, which can be used to initialize our tracker and maintain accurate trajectories. Obviously, it is difficult to estimate how many such centers we have in a given frame, and because of the low resolution in frames, accurate detection is non-trivial.
- *Tracking and Recognition of objects* This is the core of the problem, as we must both classify detected objects, and track these particular recognized objects. There are many subproblems in this section. Some of the more pertinent ones are how to choose good features for object classes, and how to track objects when occlusion or when non-detection occurs for several frames.
- *Counting the number of objects* is non-trivial because it is asking to determine the number of detected moving objects. Because of faulty detection and double-counting this may become difficult to do accurately.

It is important to note, that in the framework presented, the different sections are highly dependent on previous sections. For example, good image rectification will provide better results for object detection. For this reason, extra attention was directed to the initial stages in order to have more tractable solutions for later stages. In other words, we attempted to reduce noise and clutter as much as possible, in order to have fairly simple algorithms for more complex tasks.

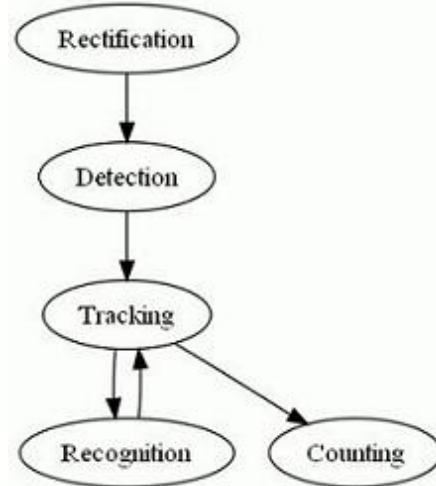


Figure 2. The framework of our given system.

2.2. Assumptions

In order to have a tractable problem and be able to satisfy all the criteria, some assumptions about our scene and objects are required.

First, we assume that we know all the types of objects which we are going to observe in the video. That is we have defined 6 possible classes of objects $\mathcal{C} \in \{Cars, Buses, People, Motorcycles, Taxis, Other\}$. We provide an 'Other' class of objects in order to classify objects we are unsure of. For example, we classify groups of objects traveling together as 'Other'. In addition we also assume we know the underlying distribution of each object category. These were estimated by looking at the video.

We also assume prior knowledge on the semantics of the image. By this we mean that objects at boundaries are either coming into the image or are out.

3. Solution and Algorithms

Described below are the algorithms and procedures which were used in order to solve our problems. Let us just briefly present some notation in the interest of clarity.

We will denote our video sequence, \mathcal{M} , as series of frames such that $\mathcal{M} = [f_0, f_1, \dots, f_m]$. As noted before, classes will be referred to as $\mathcal{C} = [c_1, c_2, \dots, c_6]$. Finally, an object detected in a particular frame will be written as O_f^i , where i refers to the particular object and f the frame in which we are detecting. Further notation will be added when necessary.

3.1. Image Rectification

Two important processes were done in order to perform good image rectification. The first was accounting for cam-

era motion and the second was having a perspective model. We now describe both of these in more detail.

First let us restate that, in order to remove camera motion from \mathcal{M} , we assumed nothing about the background or regions in our frames. Our goal here is to rectify each frame, f_i , in reference to the first frame f_0 .

We begin by extracting SIFT points and descriptors from all of our frames. We denote the points collected at frame i by p_i , and the descriptors associated with these are d_i . We now calculate the distance between two points, for all possible pairs of points in two frames, by

$$dist(p_b^j, p_{b+i}^{j'}) = \tan^{-1}(d_b^j, d_{b+i}^{j'}) \quad (1)$$

where b denotes frame indices which are multiples of 15, $i = 1, 2, \dots, 14$, while j and j' denote SIFT points found in respective frames. Matched points are those whose ratio between the best and second best match for a single point, is smaller than 0.6. In essence, we are finding correspondences between frames in intervals of 15 frames. This is because we estimate significant camera motion every 15 frames.

We now construct a 2D affine model for each f_i such that

$$X_0 = R_i X_i + T_i \quad (2)$$

where X_i are the SIFT points in f_i with correspondences, R_i is the rotation at frame f_i , and T_i is the translation for frame f_i .

For each f_i , we now begin by finding the solution to this model by using all correspondence points. We assume that 60% of these points are part of the background model. For each point, we now generate the error from our R and T computed. Now using only the best 60% of our points, iteratively, compute the optimal R_i and T_i .

Finally, we warp each f_i according to the corresponding R_i and T_i . This now provides us with new frames which have been corrected for camera motion.

Once we have corrected for camera motion, we now estimate the location of the vanishing point. We do this by calculating the Hough lines in our scene. We then calculate the intersection point between two lines. This provides us with the vanishing point, $p_v(x, y)$.

We now compute a simple perspective model by the use to the y_v coordinate previously computed. Our proposed model

$$pers(y) = \frac{1 + \frac{-1}{y_v}y}{K} \quad (3)$$

where K is a constant, only uses the y is component of the vanishing point. From observing the vehicle sizes throughout our frames, we noticed that the size of objects stayed constant across the x axis. For this reason, we do not include this coordinate in our model. Finally, we can now use this perspective model to create a factor which can scale

speed and area of objects at different heights throughout our frames.

3.2. Detection and Segmentation

In order to do successful detection, we are going to be working with the rectified frames computed previously. In other words f_t has already been rectified.

We begin by computing a background model \mathcal{B} by taking a median image from \mathcal{M} . The result from this procedure provides us with an image of our scene with all the objects having been averaged out. This can be seen in the middle image of Figure 3.

We can now subtract \mathcal{B} from any given frame in order to have regions where objects are located. We will denote this type of image as F_t . In addition, we do not simply subtract \mathcal{B} from f , but rather observe and remove, at every pixel the closest pixel in a small window of \mathcal{B} . More formally, the resulting image of objects is

$$F_{i,j} = \min_{i',j' \in W} (|f_{i,j} - \mathcal{B}_{i',j'}|) \quad (4)$$

where i, j refers to the pixel index, and W is a small window around i, j . We do this in order to get rid of noise remaining from our video rectification and \mathcal{B} . This typically leaves us with an image such as on the right in Figure 3.

We now search for connected patches in F . Found components become candidates for our set of detected objects $[O_t^1, O_t^2, \dots, O_t^p]$, where p is the number of patches found. At this point, we perform opening and closing of regions in order to segment moving objects as much as possible. Finally we threshold the scaled area (using our estimated perspective function) of each patch to determine if a patch is indeed an object or just noise. This leaves us with the set of detected objects in a frame $\mathcal{O}_t = [O_t^1, O_t^2, \dots, O_t^o]$, where o is the number objects in frame t . At this point segmentation is simply searching in each patch for the boundary pixels of each object O_t^i .

At this point we calculate various properties of the object. These will be key for both tracking and recognition of the objects. Some of the properties we calculate at this point are area, center, bounding box, perimeter and velocity. All of these have been scaled to void perspective effects.

3.3. Tracking

We now assume that the tracker is given the set of points \mathcal{O}_t at each frame. Similarly, these detected objects hold the properties described above.

We begin our tracking algorithm, by estimating for each O_t^i , the new position of object. This is done with a simple linear model

$$\vec{X}_t = \vec{X}_{t-1} + \vec{V}_{t-1} \Delta t \quad (5)$$



Figure 3. Above we show (From left to right) a rectified image, followed by our learned background and finally show our segmented image.



Figure 4. An image from our final video. Here we are displaying the detection, tracking and recognition.

where \bar{X}_t is an estimated position of an object O_t^i at frame t , \vec{V} is the velocity estimate of the same object. From these given position estimates, we search for potential object locations from our current observation. That is, we search O_{t+1} to justify our model. We do this by searching over a small area of possible locations. At this point, there are two possibilities.

If there are no matches to the search performed, then we attempt to find the new location of the object by using the intensity values of a small patch around the center of the object (from the previous step), and attempt to match this patch in a small local neighborhood.

Alternatively, when one or more matches are present, then we perform a weighted average on the possible location matched in order to create a new single point for the objects center of mass. More specifically, for O^i and the set of matched locations $P = p_1, p_2, \dots, p_n$ we compute the new position by

$$pos(O^i) = \frac{1}{N} \sum_{j=1}^n p_j w_j \quad (6)$$

where

$$w_j = dist(p_j, X_t)^{-1} \quad (7)$$

and

$$N = \sum_{j=1}^n w_j \quad (8)$$

We need to do this in order to consolidate a location for the object detected.

At this point, we use a standard Kalman filter in order to extract the final new position of the object detected. At this point we update the properties associated with the object. We now perform recognition on the detected object.

3.4. Recognition

In order to recognize the various detected objects, we use a degenerate tree based hierarchical classifier. At different levels in the tree we test for different features. The idea is to use simple and quick tests in order to classify the objects into sets which can themselves be tested recursively. When only one kind of object remains in a set, the class of the detected object can be determined.

More formally, we can denote $\xi_h^i(O)$, to mean the test i on object O , at height h in our tree. In our case case $h = \{1, 2\}$, and $i = \{1, 2, 3\}$. That is to say $\xi_1^1(O)$ refers to the first test which all objects go through. Similarly, let \mathcal{R}_h^i be the response of the test $\xi_h^i(O)$. Since our classifier deals with classification of multiple classes, the number of responses to ξ will vary.

Our coarse-to-fine approach can be viewed as follows. We begin by testing an object for its size. That is $\xi_1^1(O)$ tests for the size of O . In this case $\mathcal{R}_1^1 \in \{Small, Medium, Large\}$. It is presumed that small objects are people and motorcycles, while medium sized objects are cars and taxis, and that large objects are buses or clusters of objects.

We now further test our object depending on the answer of the first test. If $\mathcal{R}_1^1 = Small$, then we apply $\xi_2^1(O)$ which uses a maximum velocity in order to classify objects in terms of people or motorcycles. The velocities which we

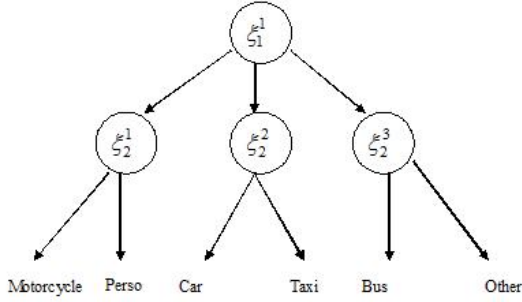


Figure 5. Our hierarchical classifier. Tests, $\xi_i^h(O)$, at different levels, enable a course-to-fine classification.

test have also been scaled to perspective. The intuition is that on average a person cannot have the same maximum speed as a person.

In the case where $\mathcal{R}_1^1 = \text{Medium}$, we now test using $\xi_2^2(O)$ for the yellow color. To do this we look at the distribution of color in a scaled (to perspective) bounding box and compare it to learned color histograms of taxis and cars. In this case $\mathcal{R}_2^2 \in \{\text{Car}, \text{Taxi}\}$. The intuition is that taxis have a unique yellow tint to them and that using the \mathcal{X}^2 distance to compare distributions, we can classify cars and motorcycles.

Finally, if $\mathcal{R}_1^1 = \text{Large}$, then we simply test for size again and observe if the size falls within a range of sizes which corresponds to bus sizes. That is using $\xi_2^3(O)$ then $\mathcal{R}_2^3 \in \{\text{Bus}, \text{Other}\}$. Objects not fitting in this range are categorized as 'other'.

Because we treat recognition as a black box, which only receives an object at each frame, we do not make any claims that objects will continuously be recognized as the same class through out tracking.

3.5. Counting

In order to count objects in the scene, we use the fact that we are aware of the frame boundaries. The main idea is as follows: when a tracked object is moving towards a boundary, then the object stops being tracked and we note that we have counted this object. In addition we perform a final object classification of the object in order to determine what type of object we should count it as.

Determining if an object is leaving the scene was done by creating a boarder slightly smaller than the frame and observing the position of the tracked object. Figure 6 displays an example of a boarder (not the actual boarder used).

In order to perform the finale object classification of a tracked object exiting the scene we used the maximum a posterior (MAP) estimator. That is an objects class was selected by

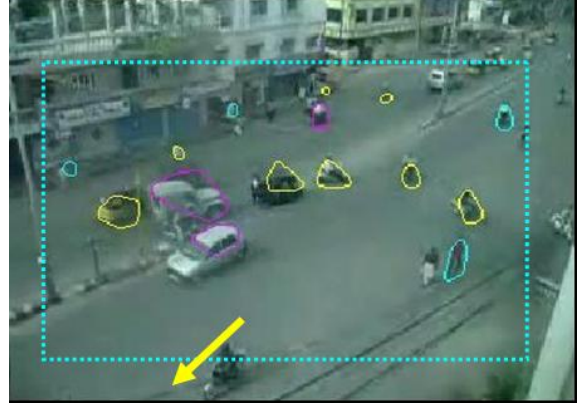


Figure 6. Example boarder around frame used for Counting

Class	True Positive	False Positive	Accuracy
Other	27	13	68%
Car	31	13	75%
Bus	7	11	63%
Person	13	13	50%
Motorcycle	45	14	77%
Taxi	17	5	76%

Table 1. Recognition Results

$$\mathcal{C}(O) = \max_c p(O|\mathcal{C} = c)p(\mathcal{C} = c) \quad (9)$$

That is, we use our estimates of prior class distribution and the likelihood of our observed data to determine the class of an object Calculating the likelihood is trivial because we have the classification history of any particular tracked object. Also, this gave better results than using a maximum likelihood estimator because in some instances, we have little data to estimate the class with.

4. Results

We now present some of the results of our system. All of the statistics were determined by human counting, over 1% of the entire \mathcal{M} .

We begin by presenting our recognition results. These can be observed in table 1. We can see that the recognition system is doing fairly well. In general, we are able to recognize about 70% of the objects given. The system has a tough time though correctly recognizing buses. This is counter intuitive because buses appear to be the largest objects in the scenes. The reason for this result however, is because we have created an 'Other' class which is recognizing groups of vehicles, which in terms of size, is similar to buses.

From the results displayed in table 2, we are very satisfied with how our tracker and detection methods are performing. One thing which is not recorded in table 2 con-

Task	True Positive	False Positive	Accuracy
Detection	50	12	80%
Tracking	43	17	71%

Table 2. Tracking and Detection Results

Type	Count	Percentage
Total	207	
Other	3	1
Car	46	22
Bus	4	2
Person	29	14
Motorcycle	108	52
Taxi	17	8

Table 3. Counting statistic

cerning our detection method, is that it does not take into account when multiple responses occur on a single object.

Finally, we have also counted all of the objects which have been tracked. Table 3 shows our results on this matter. With closer inspection to the numbers, one can notice that our estimate of the number of buses counted is slightly too large and the estimate for Others is too small. This result is because of our MAP estimation method which is not always correctly estimating the true class of our tracked object.

A video of our results is available with the source code which came with this report. In addition, we have posted on Youtube our final video as well.

5. Conclusion

The vision system we present offers a framework and solution to the problem set forth. We present various algorithms and procedures for video rectification, object detection and segmentation, object tracking, recognition and counting. The results we provide are satisfactory, but can be improved given more time.

One should note though that the costs in terms of space and computation time are not negligible when working with the entire video sequence. Storing all the data needs about 2GB of space, and requires approximately 2 hours of computation in order to compute everything.

6. Acknowledgements

Some of our implementation uses code from outside sources. More specifically, the sift implementation used was taken from David Lowe (source). The code to generate Hough lines was directly taken from Matlab. The Kalman filter was taken from Kevin Murphy.