Surveillance – Enhanced Collision Avoidance

Final Report

Rohit Annigeri, Akash Joshi, Yifei Shen, Thaddaeus Voss

May 10th, 2017

**1 Abstract**

Recently, more and more technology companies are interested in developing smart, autonomous cars. Using sensors installed around the car body, an autonomous car can get information from its surroundings and drive itself. This method works well on highways and roads since the car follows the flow of traffic, and generally, the sensor can gather this information under many environmental conditions. However, when crossing the intersection, the situation becomes more complicated. Vehicle sensors might be blocked by other cars and objects and may fail to get information of cars moving in the other direction. These scenarios can be dangerous for autonomous cars and cause accidents. Our group tries to apply a supplementary measure based on the surveillance cameras, which are widely used at intersections, to predict collisions and send information to the cars to compensate for sensors dead zones and avoid accidents. The experiment is applied under a laboratory simulation scenario.

**2 Description of the system**

The basic parts of our system are shown in the block diagram found in Figure 1 below.

At the beginning, two remote controlled cars will be put into experiment field and run freely. One lab camera will simulate the surveillance camera whose field of view will cover the whole experiment field. The DSP board will receive the video from lab camera and does background subtraction to track two cars. For transmission purposes, these two cars will be classified by different colors and the colors are carefully chosen based on experimentation and MATLAB tests to make them more distinguishable.

After tracking the cars, their movement will be predicted based on their current location and previous location. In our case, we predicted 4 frames forward which we found works best.

When there is a high likelihood of a collision, the DSP board will send messages to both cars through XBee communications boards. One car will stop for 650 milliseconds while the other car will take different actions based on each situation. When either of the cars reaches the boundary of the camera’s viewable area, it will make a u-turn to keep itself inside the area, so that we can demonstrate this in a confined area.

Two cars running in the field

Camera data sent to the DSP board

Track the cars using background subtraction and classify them by color

Predict the cars’ future location

Predict the probability of collision

Send messages to cars and control their movement when necessary to avoid collisions and keep them inside the boundary

Figure 1. Flow chart of the system

**3. Description of Possible Algorithms**

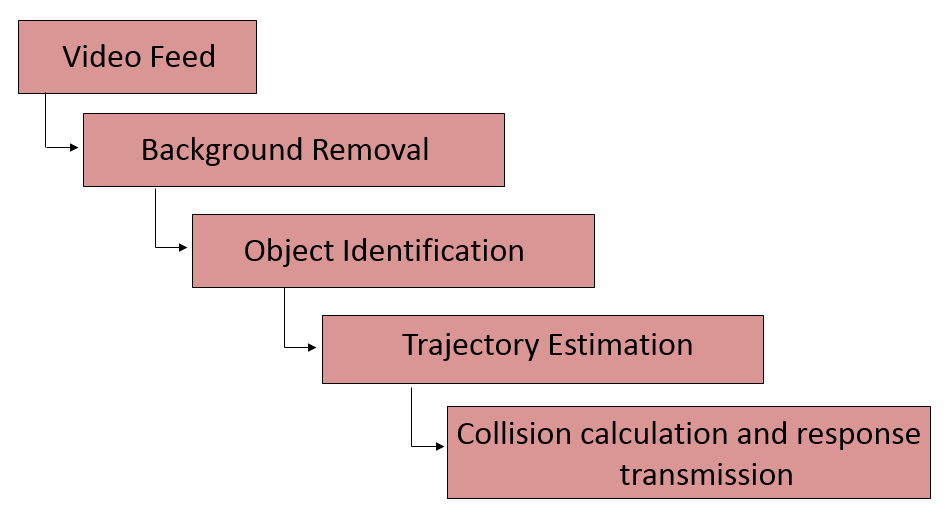


Figure 2. Algorithmic Flow Chart

There are 4 major components to the algorithm on the DSP board, with separate processing on individual cars for signal interpretation and control. The input video feed from the camera was in YCbCr format.

## Background Subtraction:

Core Concept:

The aim for background subtraction is to extract the foreground objects (in this case, cars) in real time from a video feed. The end result of this stage is a binarized real time video of the two moving cars. Possible approaches on the DSP board include frame differencing, mean subtraction, running Gaussian average, and fitting a mixture of Gaussian models. Offline background subtraction and object identification was done via multi class LDA in MATLAB.

Theoretical Approaches:

The approaches were chosen based on “A comprehensive review of background subtraction algorithms” by Andrews Sobral and Antoine Vacavant[2]. Furthermore all approaches were performed in YCbCr color space to reduce overhead during code execution. The simplest approach being frame differencing (both static and iterative) proves to be ineffective: when the object stops moving it gets registered as the background. Additionally, illumination changes and shadows also are misclassified as foreground objects appearing as intermittent noise. To compensate for these changes, the background was modeled using a Gaussian model. A weighted running Gaussian average is calculated for every pixel in every channel[1]. The mean and variance is updated every frame by a learning rate of the current pixel value and the previous mean value. From this a threshold is established:

M­t = (1-a) M­t-1 + (a) I

= (1-a) + (a) d2

d2 = (I - )2

Here, I is the current pixel value, a is learning rate and ,Mt are the variance and mean for frame t. The threshold to determine if a pixel is to be classified as foreground or background is determined by the following check.

|I - Mt| > k

If the pixel is k (sensitivity factor) standard deviations away from the mean pixel value it is considered as foreground pixel.

The control parameters are ‘a’ and ‘k’, where a low learning rate makes the system susceptible to illumination changes and high value ‘k’ would make the system less sensitive.

Practical Implementation:

A modified version of the above equations were implemented on the DSP board because of long execution time and excessive misclassified pixels. The YCbCr pixels were converted into RGB colorspace and a training period of 30 frames was given to determine the background model. The learning rate was dropped to decrease execution time. The equations were modified to use the standard formulae of mean and variance, and the check condition was modified to use variance instead of standard deviation. ‘k’ was determined empirically.

Result:

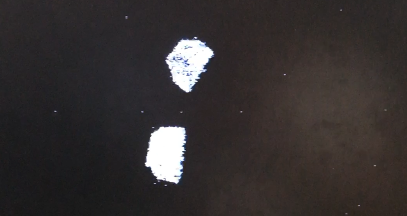


Figure 3. Binarized output of background subtraction.

The final output had minimal noise (misclassification), so there was no need for post-processing in terms of a median filter. This was compensated for in the next stage.

## Object Identification:

Core Concept:

Given the foreground pixels, the aim is to accurately determine individual cars and represent them by their centroids. As the cars have distinct colors, an appropriate threshold can be used to uniquely identify them. The approach was a multiclass linear discriminant analysis[3] in MATLAB, from which thresholds and color space projections were determined.

Theoretical Approaches:

The article referred to is “Using Discriminant Analysis for Multi-class Classification: An

Experimental Investigation”[3]. Multiclass LDA extends Fischer’s LDA, wherein we maximize the ratio of intra-class scatter to the inter-class scatter for multiple classes. The model assumes the data has a Gaussian mixture distribution. The intra-class scatter is sum of all scatter matrices.

Sw =

Where Si is the scatter matrix and xi is mean of class i given by:

=

The inter-class scatter is given by:

=

Where mi is the number of training samples of each class and is the total mean vector.

To maximize the class separation, we search among the Eigen vectors of Sw-1Sb which give us largest Eigen value. Since the maximum rank of Sb is N-1 where N is the number of classes, this also acts as a projection into a lower dimensional subspace similar to PCA. The Gaussian models were fit on multidimensional data, with the dimensions being color channels and (x,y) coordinates of each pixel. Color space transformations and channel reduction were also considered. These were implemented in MATLAB and results are presented below.

Results:

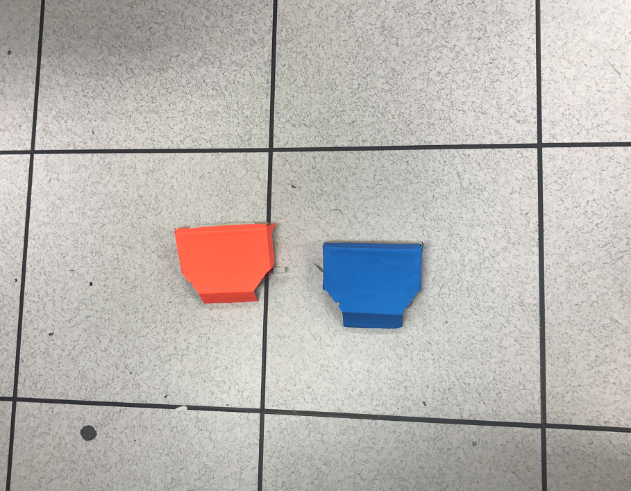
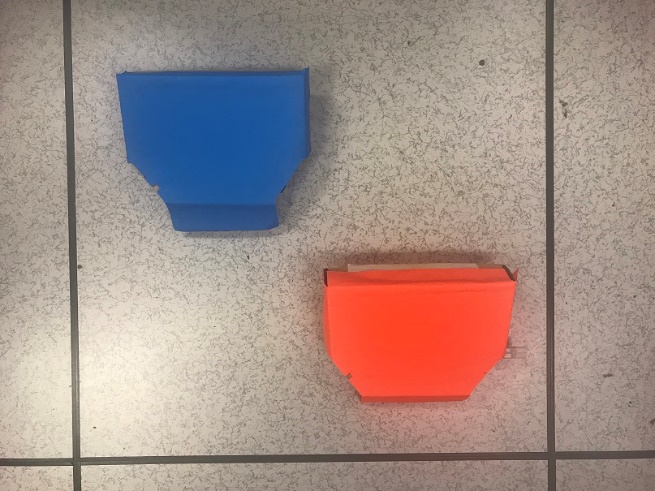
 

Figure 4. Training Image Figure 5. Testing Image

Using only RGB color space the following results are obtained.

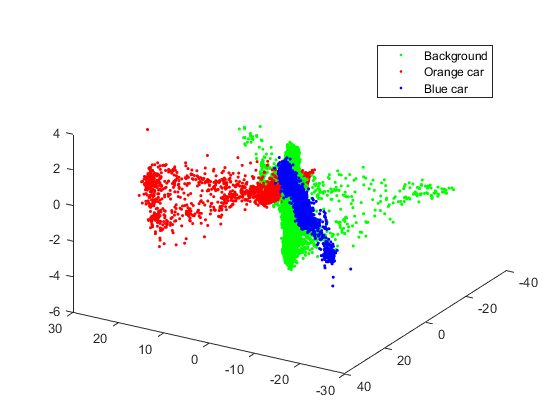
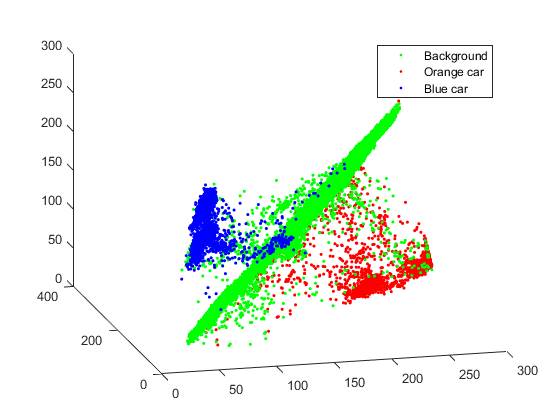


Figure 6. Training Image in RGB color space Figure 7. Training Image in transformed space

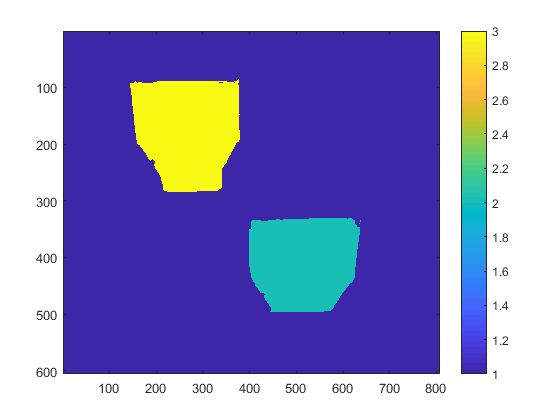


Figure 8. Predicted result of testing image

The labels are as follows 1 = background, 2 = orange car and 3 = blue car.

From Figure 6, it is evident that the classes exist in different space but are not orthogonal to each other, making it slightly difficult to separate out. After transformation, the classes are made orthogonal.

Adding location indices to the dimensions (total of 5) did not change the final prediction, however it is not possible to visualize the data. (MATLAB control parameter add\_loc)

As the input from the video feed is in YCbCr color space, it would be appropriate to visualize the data in that space. The color information is condensed only in 2 dimensions, Cb and Cr, reducing the need for computation on the DSP board. Performing LDA only on CbCr (MATLAB control parameter only\_cbcr), the results are as follows:

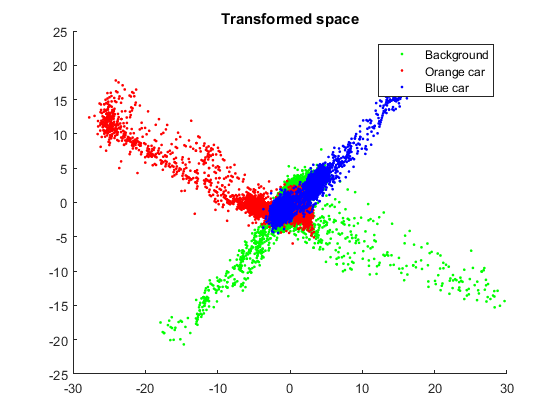
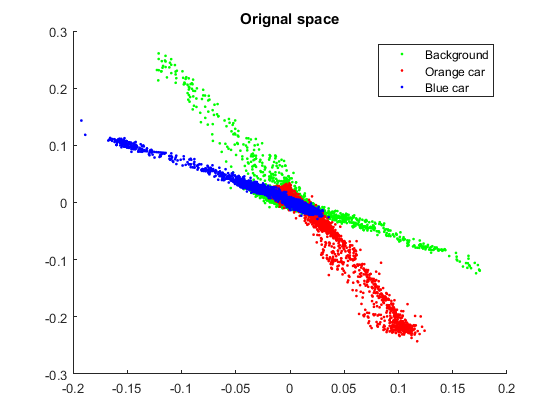


Figure 9. Training Image in CbCr color space Figure 10. Training Image in transformed space

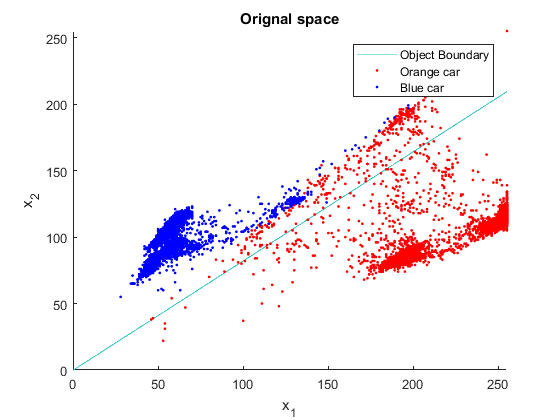
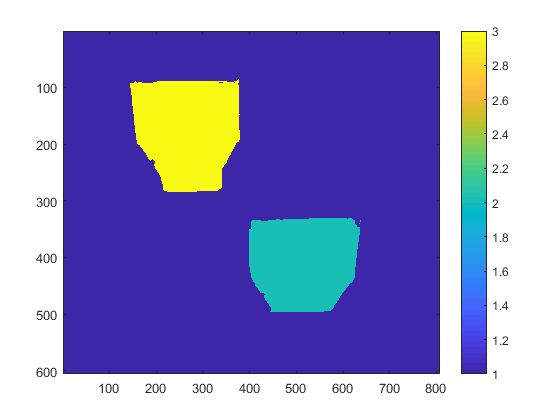


Figure 11. Predicted result of testing image Figure 12. Object boundary in CbCr color space

The input data in CbCr color space closely follows the data in the transformed space, hence it is easier to separate the objects CbCr space itself without the computational overhead of converting to RGB. Once again adding the location indices did not change the final result.

Practical Approach:

Due to memory and speed constraints, the cars were identified by thresholding on the difference between Cb and Cr values for each foreground pixel. The thresholds were determined empirically and in consideration of perceived color variations due to illumination changes. The centroids were calculated by each car’s pixel locations and dividing by the total number of pixels of the car. To compensate for the spurious noise from previous stage, a threshold was applied on the total number of pixels on each car, disregarding the noise.

## Trajectory Estimation:

Core Concept:

The aim here is to accurately predict the position of identified objects in real time. The offline approach was done using Kalman filtering in MATLAB, and the practical approach was an empirical speed-based factor.

Theoretical Approach:

The Kalman filter application to predict the future location of cars was used as a theoretical approach for trajectory estimation. The approach towards this estimation is started by inputting frames from a video sourced from CCTV camera systems. These frames are then passed through background and foreground separation algorithms. The next step is to perform blob analysis to figure out the number of objects in the frame. The MATLAB Vision Toolbox library was used to perform these tasks. This is performed every frame to track the objects in real-time.

These tracked locations of the cars/objects over a period of few frames will provide information for estimating the trajectory of the objects. This entire procedure is repeated for all the frames generated by extracting the frames. We use the real-time video player in MATLAB to show the results every frame with objects enclosed with a box. The centroids of these boxes are used over different frames to predict future centroids of the box.

The Kalman filter is applied on these centroids and the algorithm checks for presence of a centroid in the frame. If these centroids don’t exit the frame, a predict function is used to predict the location of the centroid in that frame based on the location of that centroid in the previous frames. This filter analysis is done using the **configureKalmanFilter** MATLAB function. We configure the filter as a constant velocity filter with locations of objects initialized at a specific location.

Result:

The above analysis of CCTV video footage and the project demo video shows us the results of Kalman Filter and Object Tracking.

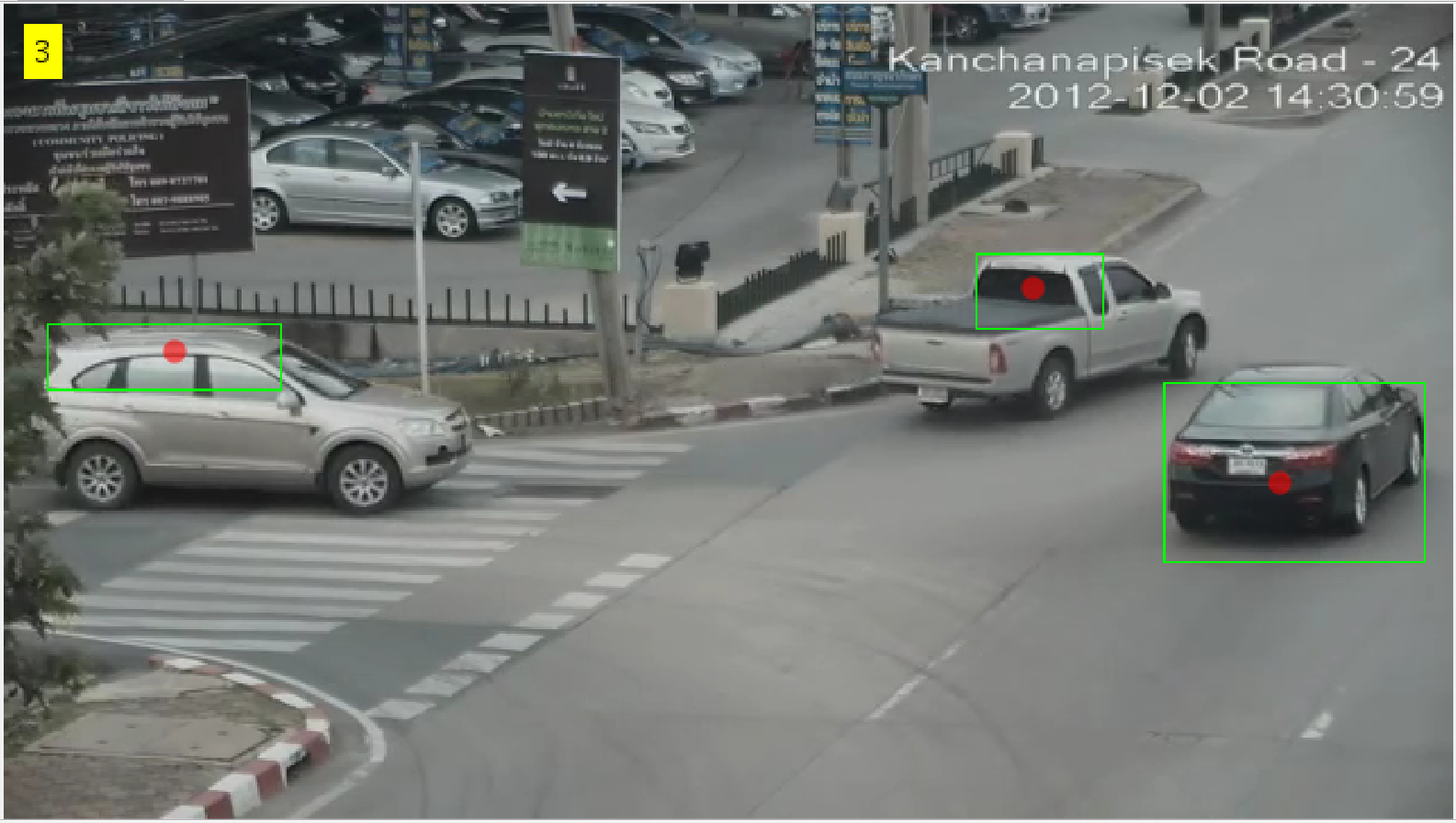
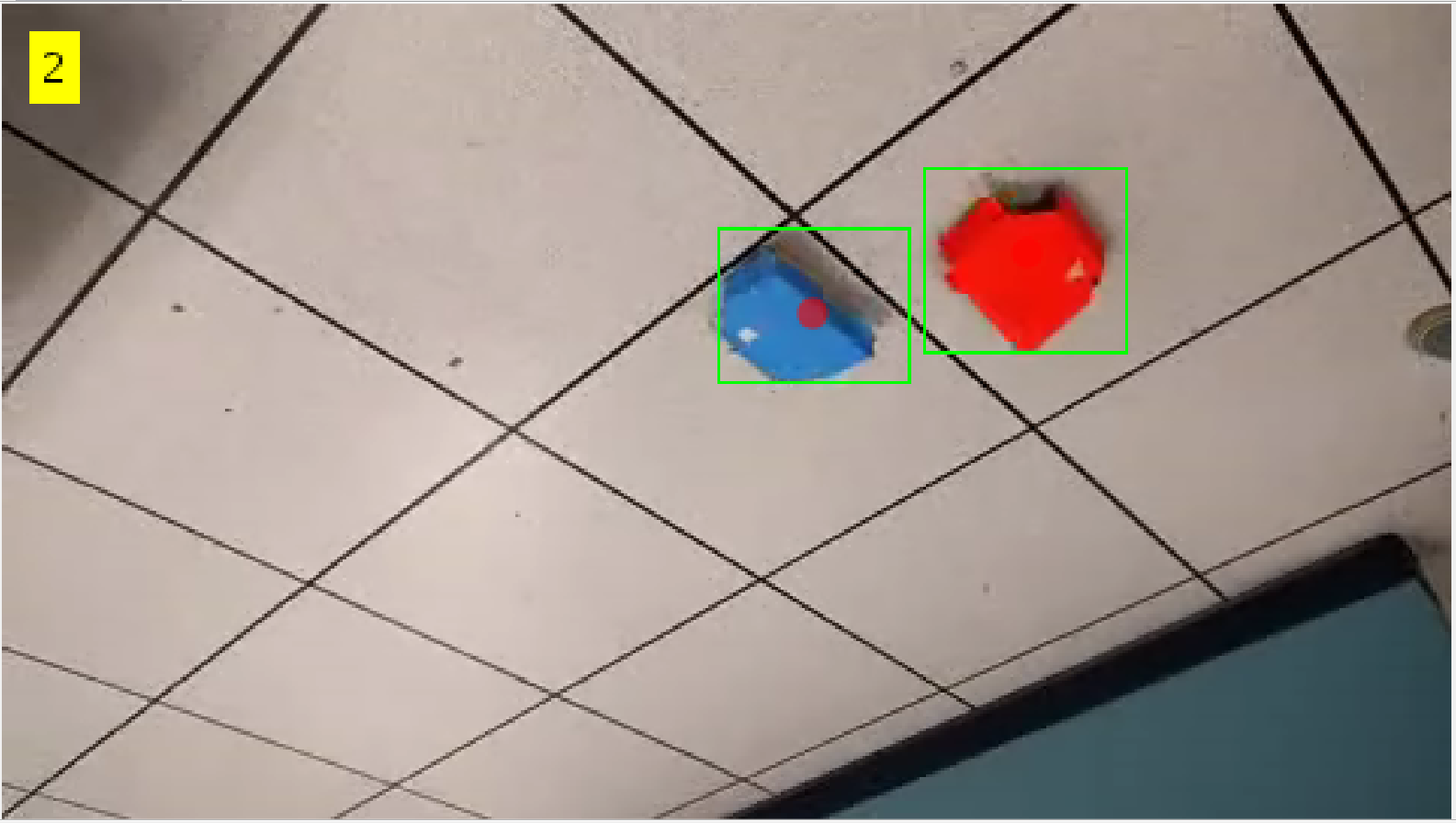
 

Figure 13: Multiple Object Tracking with number of cars detected displayed on top left for input CCTV video footage and footage from our running system

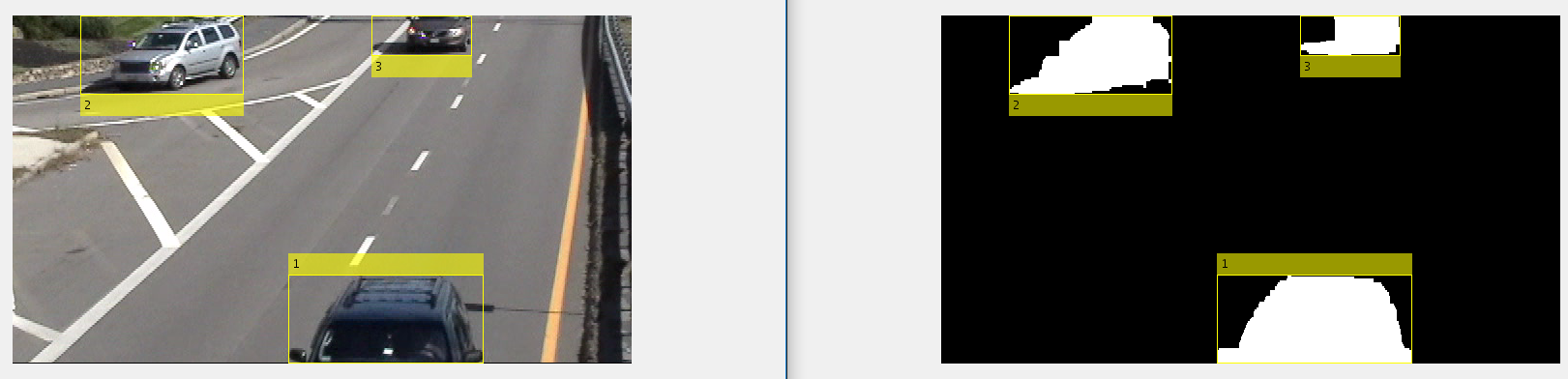


Figure 14: Multiple Object Tracking using Bayesian Estimation for trajectory estimation

Looking at the resulted output we have concluded that a very stable background video is required and this is only possible in the case where the video input is CCTV footage. The results shown below explain the importance of having a stable camera feed into one’s system.

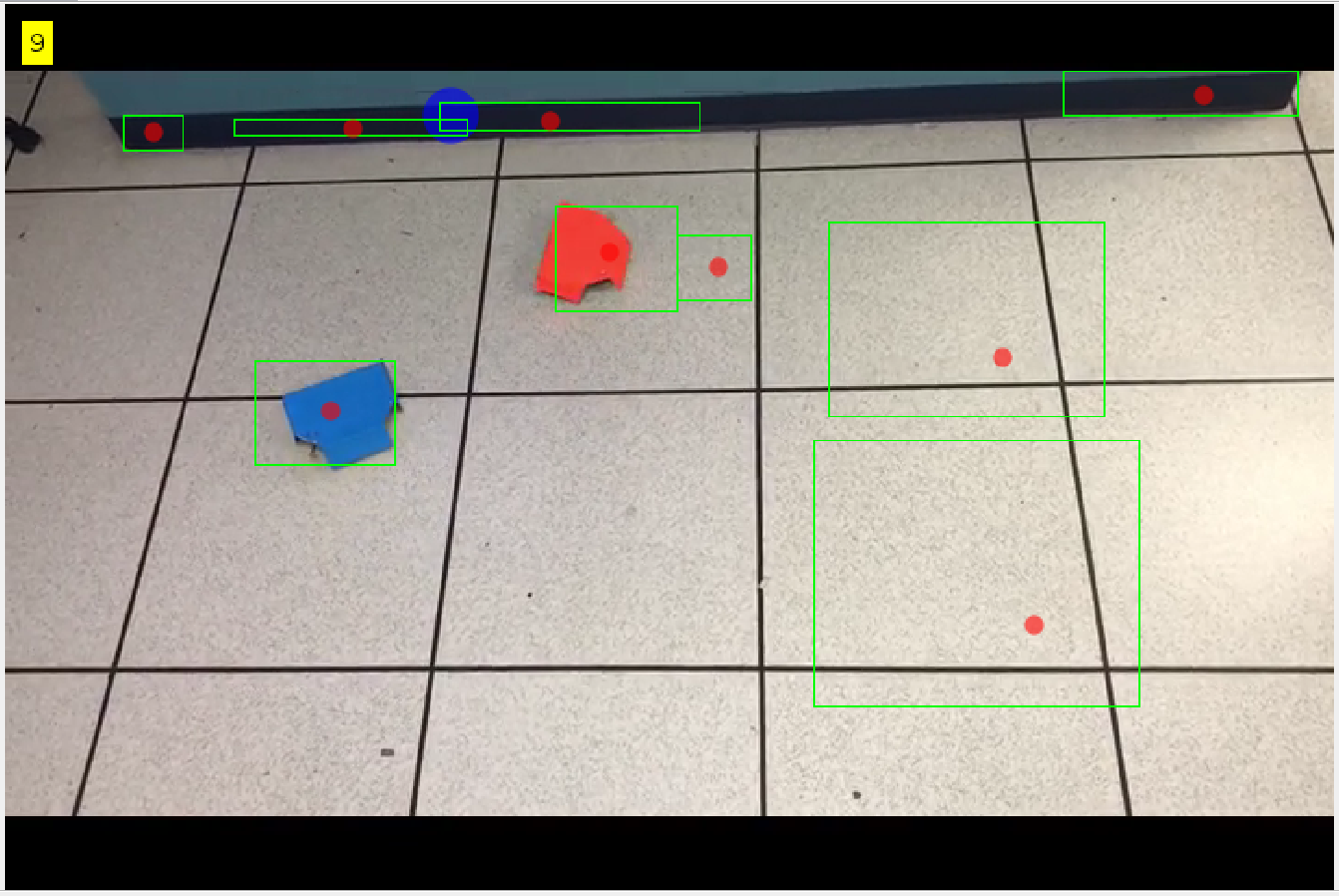
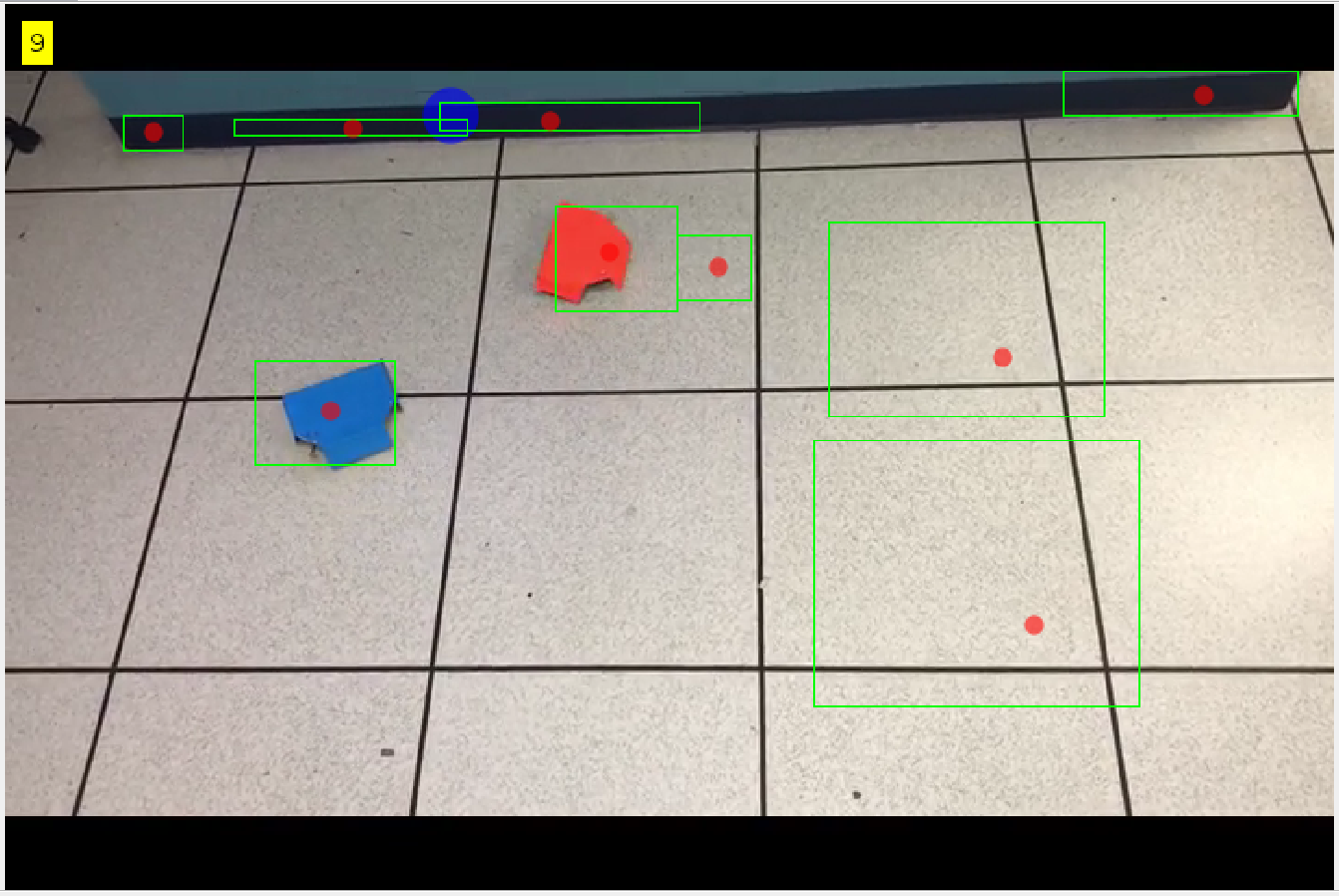
 

Figure 15: Object Tracking fails due to camera stability issues

Practical Approach:

The distance covered by the cars per frame is calculated by the displacement of centroids in the current frame from the previous frame. Based on the turning speed, response time and speed of the cars, a prediction factor was found. The anticipated positions of the cars was the prediction factor multiplied by the distance covered per frame. The nature of the test setup rendered more complicated error correction methods unnecessary.

## Collision Calculation and Response Transmission:

Core Concept:

As the cars approach each other or the boundary of the test setup, a transmission was sent to take appropriate action on the cars. The conditions to be fulfilled were that the orange car was to avoid the blue car and both cars must remain within the field of view, in this order.

Approach:

As the predicted positions of the cars approach the boundary or within a threshold of each other, a response was transmitted. When approaching a boundary, the cars were made to perform a U-turn. The threshold on collision was found empirically by measuring out the turning speed and response time. On threat of collision the orange car was signaled to perform a left or right turn away from the blue car. The equation to determine which side to turn towards is as follows:

D = (x−x1)(y2−y1)−(y−y1)(x2−x1); if D > 0 then turn right else turn left.

Where (x1,y1), (x2,y2) is the current position, predicted position of the blue car, and (x,y) is the predicted position of the orange car.

**4 Complexity Analysis**

The system consists primarily of two parts.

1. Control center: This acts as the master junction where the DSP is fed a real-time video and responds with a RF transmission if a collision is predicted.

2. Local environment: This consists of a bounded area and independently moving cars each with their own RF receiver.

Bottlenecks:

1. The limitation of the camera in accurately determining colors and limited field of view. This results in poor background detection as lightly colored cars appear white.

2. Limit on the speed of cars. Fast moving cars may not be tracked by the DSP. The receivers are blocked during execution of turns, responses are ignored and collision is then unavoidable.

3. Compounding errors arising in the decision process from inaccurate prediction/centroid may lead to the orange car turning the wrong way. This case occurs when the travel paths of both cars are almost parallel.

## Optimizations:

Much of the computational cost in the program comes from the background subtraction and object identification task. Before optimization, for one frame, this task took 200,000,000 instruction cycles to process one frame and detect each object in the frame and classify it using one center pixel. To optimize this step, and the subsequent steps that came after developing the background subtraction, we implemented three main ways of optimizing code in C for the DSP board. The first was to examine where float multiplications were in the code and only use them if absolutely necessary. In many cases, integer precision was enough for this step. Second, we replaced the now integer multiplications with bitwise shifts. Implementing these two optimizations, the number of cycles that background subtraction and object identification took reduced from 200 million to around 140 million, a considerable difference. However, we still noticed that there were parts of this function that was causing unnecessary lag. The third optimization involved changing the way that pixels are accessed. Originally, we accessed each pixel that was stored in RAM in each iteration of the for-loop. Once we changed that to access a local variable with the same information, the instruction time for this step reduced to around 120 million from the original 200 million cycles before any optimization. Once we discovered these two optimizations, we made sure to implement them whenever we could as we developed our program. This improvement of 40% resulted in being able to process many more frames per second than before, and thus, more collisions being avoided.

**5 Major Challenges**

1. Accurately estimating the background, in the case of a changing background (illumination) or an object stopping. The solution used was to train the background as a Gaussian model in RGB color space for 30 frames and prevent its update during runtime.

2. Loss during response transmission to cars was found to be 20%. Solution used was to decrease the overall number of transmissions and use a watchdog on the cars to ensure they return to the testing area. Response to cars were coupled together and both cars received the same packet.

3. Decision outcomes were case-based, where cars approaching each other and the boundaries needed special attention. Also, enough time needed to be given allow preventive action by the orange car. Solution used was to prioritize the order of action, giving preference to avoiding collision than staying within the testing area.

**6 Modeling Tasks**

The task ahead of us is to now model several vehicle collision scenarios. There are some aspects of the model that are fixed such as, we are working with two cars moving in a straight line within the boundaries of the camera’s field of vision. The cars also have two different colored chassis’ and this helps in object detection of the cars. The cars then receive data from the DSP board if they are about to collide or exit the field of vision.

The two cars were built as a 2-wheel drive system using an Arduino and motor drivers. The Arduino was also connected to a RF Link receiver which communicates with the DSP board where RF Link transmitter is connected.

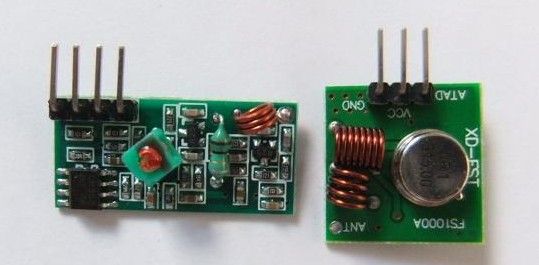


Figure 16: RF 315/455 Mhz Transmitter/Receiver Module

The entire system consists of two receivers and a transmitter connected to the 2 cars and a DSP board, respectively. The data package is transmitted to both the cars at the same. Both the cars read the data and do the following as shown in the table:

Table 1: Instruction dataset for both cars

|  |  |  |
| --- | --- | --- |
| **Data Package** | **Car 1** | **Car 2** |
| 0000 | No Response | No Response |
| 0001 | No Response | No Response |
| 0010 | No Response | No Response |
| 0011 | No Response | U-Turn |
| 01xx | Left Turn | Stop |
| 10xx | Right Turn | Stop |
| 11xx | U-Turn | No Response |

Looking at the table above we can conclude that only one car is given a correction algorithm/route and the other car just stops when a collision is about to happen. Once the car receives the data, it performs a task, and during this time, the DSP board does not send data to the car.

Bottlenecks:

1. The battery voltage would change over time and that would change the speed of the motors due to lack of a feedback system
2. The turning angle for making a right or left or U-turn would also get affected due to change in battery voltage
3. The RF transmitter and receiver would have a data loss of around 15% and this would sometimes lead to a crash

Optimization:

We can optimize the system to become more robust and provide repeatability by having a closed loop system altogether. This can be achieved by adding a speed encoder sensor and a PID algorithm, such that the cars are at the desired speeds at all the time. We also need to add a magnetometer sensor to measure the angle by which the car takes a left or a right turn. Instead of using RF Link Transceivers pairs, we can use a Xbee Mesh Network such that we always have data transmission confirmation from all three Xbee Nodes. Additionally, the use of a servo motor would also help in getting an accurate steering angle.

**7 Human Factors**

While our project and its final test setup concern the control of cars using no human input or interface, we aimed to include the human evaluation of the warnings it sends to cars in near-collision and collision scenarios. We continuously considered how not only cars should operate in a real-world scenario, but also how a human in an autonomously operated vehicle would like their car to behave in dangerous scenarios. In deciding to stop one car and have the other car avoid the stopped car when a collision is likely, we modeled the likeliest scenario in today’s current vehicle-vehicle collisions. Additionally, we made sure to consider how vehicles travel in modern infrastructures. Rather than let the cars move randomly always, we developed our final system so that the vehicles move in a straight line and turn at 90 degree angles as they do at intersections especially.

Originally, we had planned to give each car a destination point in the camera’s viewing field to travel to, as humans travel with a destination in mind, and the car would travel in straight lines and turn at 90 degree angles to get to the destination point while avoiding the other car in a collision scenario. This proved difficult as our viewing area in the lab was relatively small for the size of the cars. As such, in the end we implemented a system where the cars move in straight lines and turn at 90 degree angles, but without the destination points as a feature, while still avoiding collisions when they are about to occur.

**8 Training**

To appropriately train our system to always perform as intended, we employed two main methods of training. While these are not traditional forms of training as employed in neural networks or the like, they nonetheless provided a basis for our system to understand the scenarios it would be placed under to operate effectively. The first was to consider and provide all of the test scenarios that the cars were likely to face when running freely. This included the vehicles approaching each other at a 90 degree angle, head-on, in parallel or in close following travel scenarios, among others. We understood that these were the likeliest situations the cars would get themselves into, and as such, paid close consideration to these particularly.

In performing the offline LDA in MATLAB, we also offered real training data to this algorithm that detects and labels each car. This training data was in the form of screenshots from the camera through the DSP in raw format, and from photos taken with other cameras from cellular phones of the two cars. This training data allowed us to accurately train our LDA so that the cars would be theoretically classified in real-time had we decided to implement the LDA algorithm on the DSP board itself.

Continued on the following page

**9 Final Project Schedule**

Below is the completion date of each of the main tasks of the system. The team members who worked on each associated tasks are listed next to each (A = Akash Joshi, R = Rohit Annigeri, Y = Yifei Shen, T = Thaddaeus Voss).

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Milestone** | **Associated Tasks** | **Completion Date** |
| Milestone 1 | Detect objects in a scene, identify each of the objects, and predict future location | * Isolate moving objects from background (R, Y) * Detect objects in the scene (R, T) * Identify objects (R, T) * Predict future location (Y, R) | March 13 |
| Milestone 2 | Successfully run intersection simulation using remote-controlled cars | * Obtain two remote-controlled cars (A) * Develop code to control two cars using relevant hardware (A) * Perform tests controlling cars at different speeds, in collision and non-collision scenarios (R, Y, T, A) | April 7 |
| Milestone 3 | Derive percentage likelihood of a crash using car metrics and apply these warnings to intersection simulation | * Transmit successfully over GPIO to remote-controlled cars (A, R) * Develop algorithm to keep cars within the intersection (R, T, Y) * Develop and implement algorithms to avoid a collision (Y, R, T, A) | April 19 |

**10 Final Test Set-Up**

For the demonstration of our full project, we used the following equipment:

* + 1 TI TMS320DM6437 Fixed-Point DSP Board
  + 1 Overhead Video Camera
  + 2 Remote-Controlled Cars each with an RF Receiver Board
  + 1 RF Transmission Board connected to the DSP board by GPIO

To simulate our technology working in a real-world environment, we treated the camera’s viewable area on the ground in the lab as a mock intersection. We ran the remote-controlled cars on the intersection in different scenarios with the camera pointed at the intersection from above, the video being continuously fed into the DSP. These scenarios included vehicle-vehicle non-collision, near-collision, and collision scenarios. To visualize what was happening inside the DSP board as our code ran, we overlaid the calculated centroids of each car and the predicted future location of the car onto the live feed displayed by an ordinary computer monitor. In this way, we could see where the two cars would be in the future and if a collision is likely.

**11 Board**

We used the TI DSP Board TMS320DM6437 Fixed-Point for processing the video feed from a camera. This was the main point of processing for all of our project’s programming needs, and where we developed all code in C. This board allowed us to process the video feed using either s-video or component wired communications.

In addition to the TI DSP board, we also used XBee communications boards with antennae to communicate between the DSP and two remote control cars over RF frequency. We observed the cars and calculated their movement metrics from a video feed out of the camera connected to the TI DSP board. To process the data coming in through RF communication to the CBee communications boards on each car, there was also an Arduino board on each car. The Arduino processed the commands coming from the DSP transmission to move the car appropriately, given each scenario.

**References**

[1]A. Sobral and A. Vacavant, "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos," *Computer Vision and Image Understanding*, vol. 122, pp. 4–21, May 2014.

[2] J. Gallego, M. Pardàs, and G. Haro, "Enhanced foreground segmentation and tracking combining Bayesian background, shadow and foreground modeling," *Pattern Recognition Letters*, vol. 33, no. 12, pp. 1558–1568, Sep. 2012.

[3] (Li, Zhu and Ogihara 453-472) [5] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik, "A real-time computer vision system for vehicle tracking and traffic surveillance," *Transportation Research Part C: Emerging Technologies*, vol. 6, no. 4, pp. 271–288, Aug. 1998.