**Cornell Cup USA**

**Application**

**Situational Awareness Fault-finder Extension**

**Team members:**

**Nikolas Davis**

**Cuc Duong**

**Matthew Stehr**

**Tuan Hoang**

**Faculty Advisors:**

**Professor Marek Perkowski**

**Professor Mark Faust**

# Abstract

A majority of motorcycle accidents are the result of other vehicles failing to detect the presence of the biker. Besides dressing brightly, the cyclist must maintain a high situational awareness and drive defensively to compensate for their lack of conspicuity. An intelligent system that enhances an operator's awareness could prevent fatal accidents. Such a system could also assist the operator of a larger vehicle to notice the motorcyclist. Furthermore, distracted drivers could be alerted to potentially dangerous situations.

We have worked on building an affordable Situational Awareness Fault-Finder Extension (SAFE) device, which could be used with automobiles, motorbikes, or bicycles. SAFE is able to track lateral and posterior vehicles, monitoring their relative speed, position, and direction. SAFE sends auditory alert to users about the potential hazardous vehicles. Based on the distance, velocity, and direction of the vehicles, the alarm varies in amplitude and pitch. A vehicle goes to the predefined safety region will trigger the alarm system. For examples, slowly approach vehicle in stop sign scenario, or parallel moving vehicles will be small sound and low pitch. On the other hand, a speeding approach vehicle with the direction toward our driver will be loud alarm with high pitch. An alarm appears from the left side of the speaker warns the driver about dangerous vehicles on the left hand side, and similar for the right side alarm. For fatal vehicles from behind, the left and right speaker signal at the same time.

In the market, the most similar product to SAFE is Mobileye. You could probably say that Mobileye is the leader of the vehicle collision avoidance nowadays with more than 15 years on researching and releasing 5 product versions. However, while Mobileye features prevent the driver from dangerous situations caused by the driver’s out of focus, SAFE protect the driver from serious situations caused by other vehicles’ unpredicted behaviors. For instance, a truck losing brake system would hit our driver unintentionally except his/her quickly notices the disastrous vehicle and changes his/ her direction or velocity. Moreover, SAFE “keeps an eye on” the back view of the driver to help them releasing the stress on watching the behind scenes, while Mobileye only focuses on the front view.

<http://www.youtube.com/watch?v=HXpiyLUEOOY>

# Challenge Definition Restated

Motorcyclists are 35 times more likely to be in a fatal crash than a passenger car. In 2011, 81,000 motorcyclists were injured in accidents on U.S. roads; of those casualties, 4,612 were fatal [1]. The European based MAIDS report found that the majority of motorcycle accidents could be traced to other vehicles failing to detect and recognize the cyclist [2]. Helping a cyclist recognize such situations sooner could save countless lives.

In the paper 'Situation Awareness during Driving: Explicit and Implicit Knowledge in Dynamic Spatial Memory', Gugety states that driving performance is constricted by the working memory of the drivers [3]. Because of this limitation, drivers focus only on potentially hazardous cars in a crowded traffic environment. Commuting with a preoccupied mind on busy streets is taxing on mental resources. This is especially true in some locations, such as Vietnam, where motorcyclists and bicyclists share the same lanes with automobiles and trucks. As a result, fatal accidents still occur due to the dangerous behavior of large vehicles. One serious example involved a truck with a malfunctioning brake system plowing into a group of motorcycles. The cyclists were not aware of this hazardous truck rapidly approaching from behind. Our group's main objective is to support motorcyclists and bicyclists in avoiding traffic accidents caused by other vehicles.

More specifically, we think that a large portion of this danger comes from the effort required to keep an up-to-date mental map of the surrounding vehicles. The hardest part of this task is when the driver of a two-wheeled vehicle must take their attention away from the road ahead to look behind. This additional cost means that a driver can only ever have a limited rear view. As Gugety notes, human brains must selectively apply their limited concentration to hazardous objects while driving. Given limited resources to begin with, an additional cost in order to maintain rear situational awareness only serves to take away from the overall mental resources. This effectively limits the overall situational awareness any driver may have.

We believe providing the driver with a better sense of their surroundings is the core challenge. We believe a good solution involves providing a representation of surrounding vehicles to the driver. Specifically, we think that mitigating the cost of maintaining rearward awareness would free mental resources overall, allowing the driver to have more time to look forward, and even spend resources thinking at a higher level of decision making and planning. Given information about vehicles behind the driver, a good solution would already have the information needed to provide timely and accurate alarms about hazardous situations.

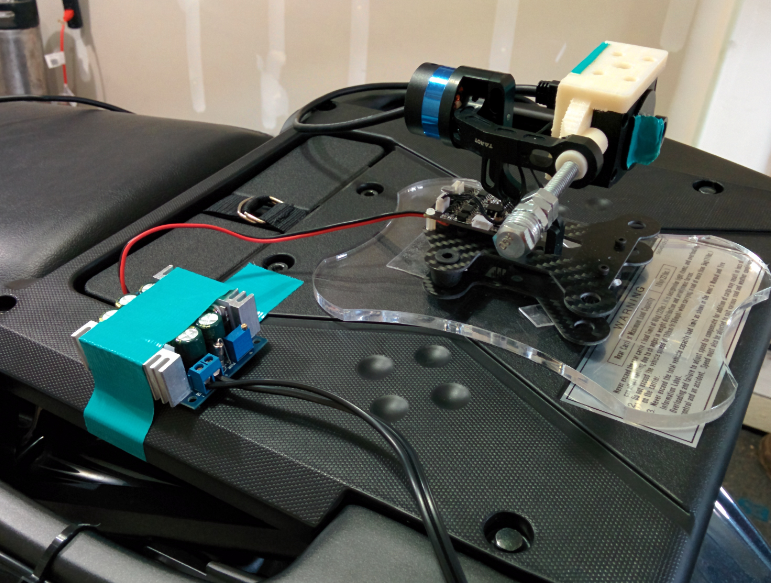
Ideally this solution would also offer use in any case. This includes day and night operation, as well as accounting for various weather conditions. Care would be taken in a good solution to not unintentionally add to an already distracting environment, except when dangerous situations are imminent. As a result, the user should have increased overall situational awareness, and be able to spend more time thinking at a higher level of decision making.

Adding to the original challenge definition, non-distracting aspect of the solution is preferred since in some traffic environments like highway, sensing suddenly or alarming continuously will affects the cyclists’ mental. The interface should provide a gentle service that considers users’ time to action. Importantly, the solution has a certain level of accuracy to minimize bothering alarm in safe cases.

# Solution Updated

# Functionality and design

Our device quickly searches for and identifies specific moving objects – other cars, trucks, semi-trucks, and motorcycles. Then, it makes an estimate of the object’s position and velocity relative to our vehicle. Using all of this information we construct a model of the current situation and give that information to the driver in a way that they can absorb very effortlessly. Our solution gives the driver audio alerts in the case of imminent danger.



Different from the original solution, our device does not provide the size and the shape of the interest objects. In exchange, we add the direction of traveling attribute of each object into our information framework. The fact is that fast changes in direction and distance from objects toward our host are far more important than their shape and size.

As shown in the block diagram in Figure 2 below, our system consists of several high-level modules. These modules are: one CMOS camera sensors that are actively stabilized by a combination of an inertial measurement unit and servos; an ultrasonic sensor; an image processing algorithm; a model builder/decision making module that accumulates information from lower level modules; a user interface rendering program; and finally a user interface system consisting of a audio display and input device.

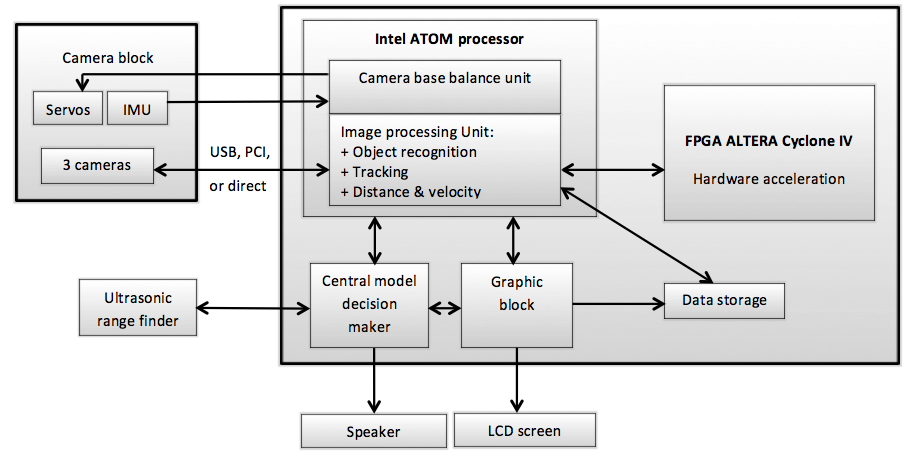


Figure 2. A block diagram of the proposed solution

CMOS camera module provides the core of our system’s ability to sense our environment. Different from the original solution, our camera has a physical infrared filter lens inside to mitigate condense IR beams during hot day. With the IR filter, the input image is much more similar to what human eyes can see. The filter gives a more stable and uniform histogram of image intensity. The purpose of using CMOS camera with infrared sensor was to keep the solution working at night. However, highlight lanes and pavement pain are available so that normal camera can see clearly at night time.

<http://arstechnica.com/business/2014/04/glow-in-the-dark-roads-make-debut-in-netherlands/?kw=100k_pvs&search=100k_pvs>

The original solution suggests using three cameras for three sides of the bikers. In our progress, we recognized that one camera with large view size can handle almost three sides. In exchange, we save a lot in the total cost of the final solution. Likewise, only one ultra-sonic range finder that we built is being used to detect very close objects behind the biker at high speed. Very close rear objects are successfully tracked and managed by the camera.

An important change toward the original solution is the user interface. According to many researches [citation here] about what makes driver fast response and less be distracted, audio likely works better than visual interface in time response. Though audio easier bothers users, visual does not have high efficiency on alarm and giving fast instruction. By these arguments, our solution utilizes only audio alarming and instructing system. This decision also saves the cost for the device.

Image processing unit plays the role of our device’s central heart which performs object recognition, position and velocity tracking. In addition to the proposal, it computes and tracks the direction of vehicles and save all of the useful information into a link list. As notice, the length of the link list is unbounded so the device has no limit in number of detected and tracked objects. During our progress, we recognize our expected image processing unit needs two important features which are new vehicles management and wen-out-of-frame vehicles deletion. Therefore, the two essential features have been added into the framework. Although our proposal suggested using machine learning and classification method of support vector machine and neural network, we use a much simpler method is Bayesian segmentation using Expectation Maximization technique. The method proves of time efficiency with high precision. Not only that, many other techniques such as Canny filter, Hough transform, RANSAC, vanishing point calculation, homography transformation, blob detection, Extended Kalman filter for vehicles, Kalman filter, least-mean-squared linear fitting have been implemented. The technical documentation part explains in details of the techniques inside the image processing unit.

No hardware acceleration has been made.

Data storage

The camera balancing servos has been purchased, which is the GoPro Hero 3 gimbal. We think that the balance servos are nowadays mass produced with optimization in hardware and software. To adapt our camera size into the gimbal, we built a 3D printing holder which provides a nice place to attach the camera in. A mount to hold the gimbal into a motorbike was printed by a laser cutter. Having pre-made servos saved us a huge amount of time and man power to focus on image processing unit. However, it did provide us the opportunity to work with a 3D printer and a laser cutter.

The ultra-sonic range finder meets the original specification of ten feet reading range through a USB connection. The module outputs the distance and the velocity of the closest object right behind the motorbike. Currently, the range finder works at 60 Hz frequency and able to detect very close vehicle.

The central decision maker unit uses the data which are real time objects’ position, velocity, and direction provided by the image processing unit to predict potential collision, then sends an alarm and instruction signal to the audio interface. Setting threshold is the most controversial task in this unit due to the trades off between high precision and more bothering alarm signals.

# Performance measurement

## Vanishing point estimation and homography transformation module test

This section evaluates the timing and accuracy of the vanishing point estimation and homography transformation module. This module outputs the top-down view/image of the traffic on a road. The output image is very important for further detection steps, because incorrect transformation leads to terrible distance and velocity estimation of existing objects.

Timing factor is measured by internal software commands. On the other hand, accuracy is calculated by comparing the true distance toward objects in input images with their estimated distance after passing this module. The test is repeated 100 times to get enough statistics confidence.

The test is conducted under different light conditions. The following table summarizes those two factors.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objects** | **True d (m)** | **Measured d (m)** | **Error (%)** | **Light conditions** |
| 1 | 1 | 1 | 0.00% |  |
| 2 | 1 | 1 | 0.00% |  |
| 3 | 1 | 1 | 0.00% |  |
| 4 | 1 | 1 | 0.00% |  |
| 5 | 1 | 1 | 0.00% |  |
| 6 | 1 | 1 | 0.00% |  |
| 7 | 1 | 1 | 0.00% |  |
| 8 | 1 | 1 | 0.00% |  |
| 9 | 1 | 1 | 0.00% |  |
| 10 | 1 | 1 | 0.00% |  |

|  |  |  |
| --- | --- | --- |
| **Types of light condition** | **Average Distance Error** | **Timing (ms)** |
| Strong, orthogonal sunlight |  |  |
| Weak, orthogonal sunlight |  |  |
| Side angle sunlight |  |  |
| Tunnel light |  |  |
| Average |  |  |

## EM module test

### Initializing parameters test

The initializing parameters step is the key aspect of the EM module. Without a good initial condition, the EM is hard to converge and unable to classify four type of categories which are lane makers, pavement, objects, and undefined objects.

In this section, we evaluate the initializing parameters part by measuring how many frames the EM module need from this step to successfully recognize the four categories. In other words, we want to know how long it takes for the EM algorithm to converge in a certain initial parameter set. Clearly, the few needed frames, the better this step is.

The following table records the current performance of this part.

|  |  |
| --- | --- |
| **Experiments** | **Frames to converge** |
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |
| 4 | 1 |
| 5 | 1 |
| 6 | 1 |
| 7 | 1 |
| 8 | 1 |
| 9 | 1 |
| 10 | 1 |
| Average | 1 |

### EM module test

This section evaluates the timing and accuracy of the EM module. This module outputs the blobs’ position and width which represent the interest vehicles within a frame. It plays an important role to the tracking process since it directly hands its outputs to the process.

Since the input of the module is the homography domain image, we calculate the accuracy based on the homography image. Timing factor is estimated by internal software. The test is repeated 100 times to get enough statistics confidence.

The test is conducted under different light conditions. The following table summarizes those two factors.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Experiment** | **# Objects** | **# Detected Objects** | **# False detected** | **# Undetected Objects** | **Precision** | **Recall** |
| 1 | 2 | 1 | 0 | 1 |  |  |
| 2 | 2 | 1 | 0 | 1 |  |  |
| 3 | 2 | 1 | 0 | 1 |  |  |
| 4 | 2 | 1 | 0 | 1 |  |  |
| 5 | 2 | 1 | 0 | 1 |  |  |
| 6 | 2 | 1 | 0 | 1 |  |  |
| 7 | 2 | 1 | 0 | 1 |  |  |
| 8 | 2 | 1 | 0 | 1 |  |  |
| 9 | 2 | 1 | 0 | 1 |  |  |
| 10 | 2 | 1 | 0 | 1 |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Types of light condition** | **Precision** | **Recall** | **Timing (ms)** |
| Strong, orthogonal sunlight |  |  |  |
| Weak, orthogonal sunlight |  |  |  |
| Side angle sunlight |  |  |  |
| Tunnel light |  |  |  |
| Average |  |  |  |

## Tracking module test

Tracking module not only tracks each interest object in traffic but also estimates their velocity and turning angle. Therefore, this section tests the maximum tracking time of the system toward an object and the accuracy of those parameters estimation.

Recall that our tracking module takes advantages of the Extended Kalman Filter model which internally calculate the velocity and turning angle. The accuracy of this module is extremely important as it greatly affects the alarm module.

While it is easy to measure the maximum tracking time, velocity and turning angle accuracy is quite a problem. Our group decides to test them by driving a car with known velocity and turning angle, then comparing those known values with the module outputs. The module is tested under the results of the EM module.

The test is conducted under different light conditions. The following tables summarize those two factors.

|  |  |
| --- | --- |
| **Objects** | **Maximum tracking time (s)** |
| 1 | 10 |
| 2 | 10 |
| 3 | 10 |
| 4 | 10 |
| 5 | 10 |
| 6 | 10 |
| 7 | 10 |
| 8 | 10 |
| 9 | 10 |
| 10 | 10 |
| Average | 10 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Types of light condition** | **True v (mph)** | **Measured v (mph)** | **Error v** | **True theta (degrees)** | **Measured theta (degrees)** | **Error theta** |
| Strong, orthogonal sunlight |  |  |  |  |  |  |
| Weak, orthogonal sunlight |  |  |  |  |  |  |
| Side angle sunlight |  |  |  |  |  |  |
| Tunnel light |  |  |  |  |  |  |
| Average |  |  |  |  |  |  |

## New vehicle management module test

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Experiment** | **# New Objects** | **# Detected New Objects** | **# False detected** | **# Undetected New Objects** | **Precision** | **Recall** |
| 1 | 2 | 1 | 0 | 1 |  |  |
| 2 | 2 | 1 | 0 | 1 |  |  |
| 3 | 2 | 1 | 0 | 1 |  |  |
| 4 | 2 | 1 | 0 | 1 |  |  |
| 5 | 2 | 1 | 0 | 1 |  |  |
| 6 | 2 | 1 | 0 | 1 |  |  |
| 7 | 2 | 1 | 0 | 1 |  |  |
| 8 | 2 | 1 | 0 | 1 |  |  |
| 9 | 2 | 1 | 0 | 1 |  |  |
| 10 | 2 | 1 | 0 | 1 |  |  |

## Alarm module test

## System functional test

Maximum range

Detection capacity

Timing

Tracking accuracy

Alarm accuracy

# Technical documents

## Overview of algorithm

Our present point of focus is the computer vision algorithm that allows us to detect and track vehicles rearward of our motorcycle. Our work is currently extensively based on the system developed by Nieto [1]. This algorithm starts with a monochrome camera input with a maximum resolution of 640x480 pixels in 8 bit per pixel format. The final outputs from this algorithm are the tracking boxes of all detected objects. These boxes and their associated vectors will be fed to the final danger detection and potential user visualization stages.

The algorithm starts with an image from the camera. This image is briefly preprocessed so as to ensure optimal features are present in the road portion of the image. The image is then run through a lane-marking detector that is very similar to a 1-D horizontal differential filter. This highlights the lane markings in the image. The output from this stage is then processed through a Canny edge detector, and finally through a probabilistic Hough transform.

We use the lines from the Hough algorithm – representing lane markings that in truth are parallel in our scene – to detect their intersection at infinity, or the vanishing point. Since there will likely be multiple intersection points detected, we use a Random Sample and Consensus, or RANSAC, algorithm to detect and join the primary intersection point. This vanishing point describes the vector that describes the world Z-axis completely. From this we are able to derive the only unknowns of our camera coordinates – the rotational pitch and yaw represented by θ and γ respectively.

Knowing our camera coordinate systems – and so our image planes – relation to the world coordinate system, we are able to perform inverse perspective mapping (IPM) to retrieve a plane view of the road. This view is often called top-down or birds-eye view. This view allows us to avoid the perspective effect, and associated non-linearity of distances, in our image. We are now able to use the IPM image to measure real world distances to object behind our vehicle.

At this point, the image is segmented into one of four classifications; pavement, lane marking, object, or unknown. This is accomplished using Bayesian segmentation and parameters dynamically generated using the Expectation Maximization (EM) algorithm. We assume each classification contributes to a pixel’s intensity, and therefore a pixel intensity corresponds to the probabilities of that pixel representing each class. These are assumed to be Gaussian distributions, and their parameters are updated for each frame by and EM step. Using Bayes’ theorem and these parameters, class probabilities are calculated, and elements with high probabilities of being objects are classified as such. This generates a binary image where each pixel is either an object, or not.

Next, the binary image goes through a morphological opening step to clean up noise and yield blobs that highlight where objects can be. The leading edge of each blob is determined, as well as the height of the object. From this information a bounding box is calculated for each object.

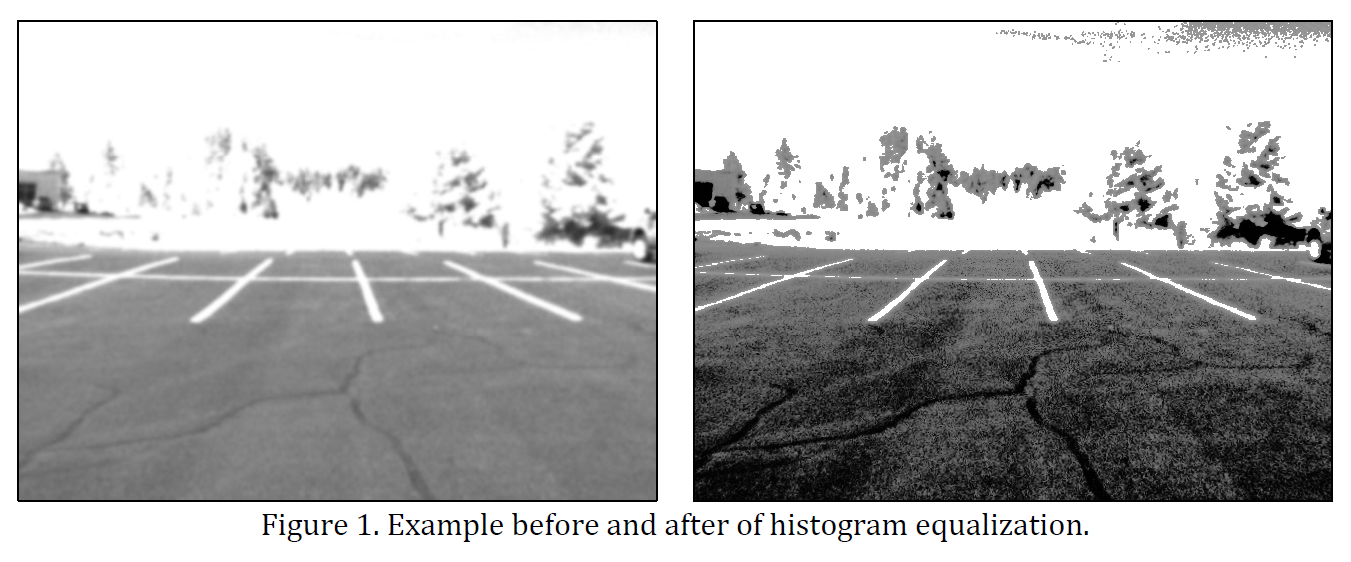
## In-depth look

### Preprocessing

The camera that we are using has a built in auto-brightness and auto-exposure.

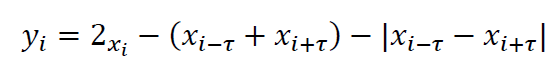
However, these features must be based internally on a histogram of the entire image.

This does not work well for many situations, in which sun glare may dominate up to half of the image. An image like this would have the road features drowned out in a feeble attempt to capture sun and sky features. Currently we are working around this by performing histogram equalization. Our ultimate goal is to have a manual brightness and exposure function that is based on the histogram of the lower third of the image, so that we are always best capturing the road features from the image exposure itself.

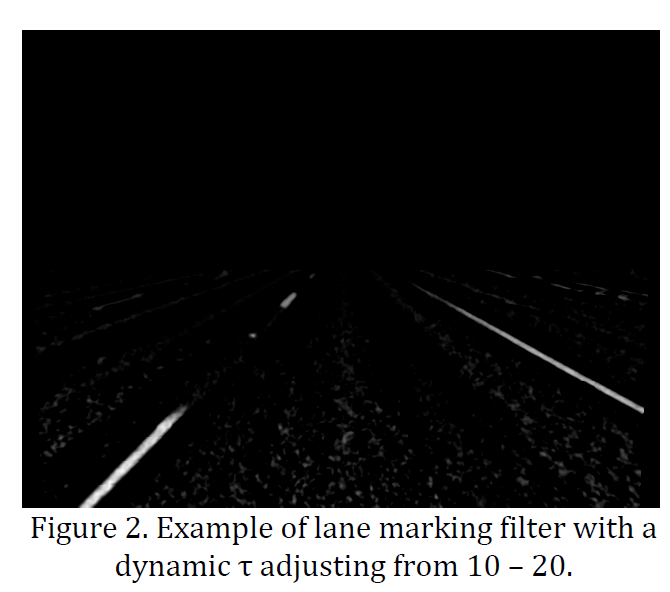


### Lane-marking filter

This filter is very similar to a 1-D horizontal differential filter. However, it is a customized filter that is design to specifically promote detection of lane-markings on pavement [1]. The idea is that lane-markings are high intensity compared to their neighbors. Also, pavement tends to be homogenous. So the low intensity pavement neighbor pixels should hold similar intensities. The formula for the lane-marking filter is below:



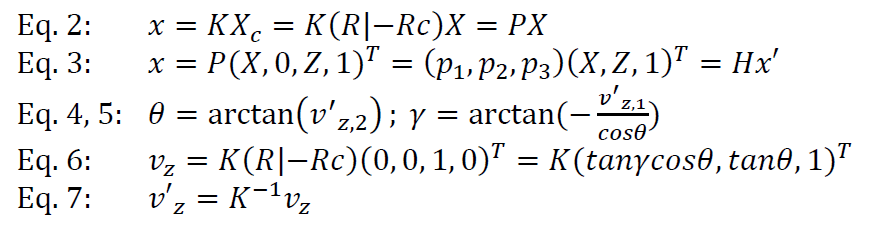
The first term relies on the high intensity of the lane feature. The second term penalizes pixels that do not have bright-neighbors at width distance τ. The third term discriminates against pixels whose neighbors are not similar intensity. Overall, this filter performs much better than a simple differential mask or Sobel filter.



Nieto mentions that to accommodate the perspective effect in the image, we can dynamically adjust τ while processing our way further down the image [1]. We take advantage of this for optimal detection. We also only process the lower half of the image to save time. For handling edge pixels, we crop them. Said another way, for the pixels near the edge of the frame that do not have the requisite neighbors for the calculation, we skip them and set their pixel in the output image to fully black, or 0 in unsigned 8-bit integer images.

### Vanishing point detection

By finding the intersection of at least two lines on the road plane, we can detect the point at infinity, or the vanishing point. This allows us to completely define the relationship, given by Nieto, of our world coordinates and our camera coordinates [1].



Equation 2 links the world coordinates, X, to our camera coordinates, Xc through the projection matrix P. The projection matrix constitutes the rotation matrix R, translation matrix c, and camera intrinsic matrix K. Equation 3 solves for the road plane where Y = 0. Using equation 7 we link the vanishing point in world coordinates, vz, to the camera coordinates v’z. We can then solve equation 2 using the definition of the vanishing point in homogenous coordinates, (0, 0, 1 0)T, yielding equations 3, 4, 5, and 6. We can directly solve for the angles θ and γ using equations 4 and 5. This reflects the equations and conclusion shown in Nieto [1].

Qualitatively, in this stage we pass the lane-marking features from the prior stage into a Canny edge detector, followed by a probabilistic Hough transform. This outputs pairs of points representing lines that were detected in the image. These lines predominately represent lane-marking features. By looking at all the line intersections in the image, and finding the most prominent grouping of them, we are able to estimate the vanishing point in any given image. We utilize a variant of the

RANSAC algorithm, called MSAC, in order to find the prominent grouping.

Most of the operations in this stage are canonical in computer vision: e.g. Canny,

Hough, RANSAC, and Kalman. The Canny, and Hough algorithms are implemented using OpenCV. We utilize a library for the MSAC algorithm that was written by Nieto himself (Available: <http://sourceforge.net/projects/vanishingpoint/>).

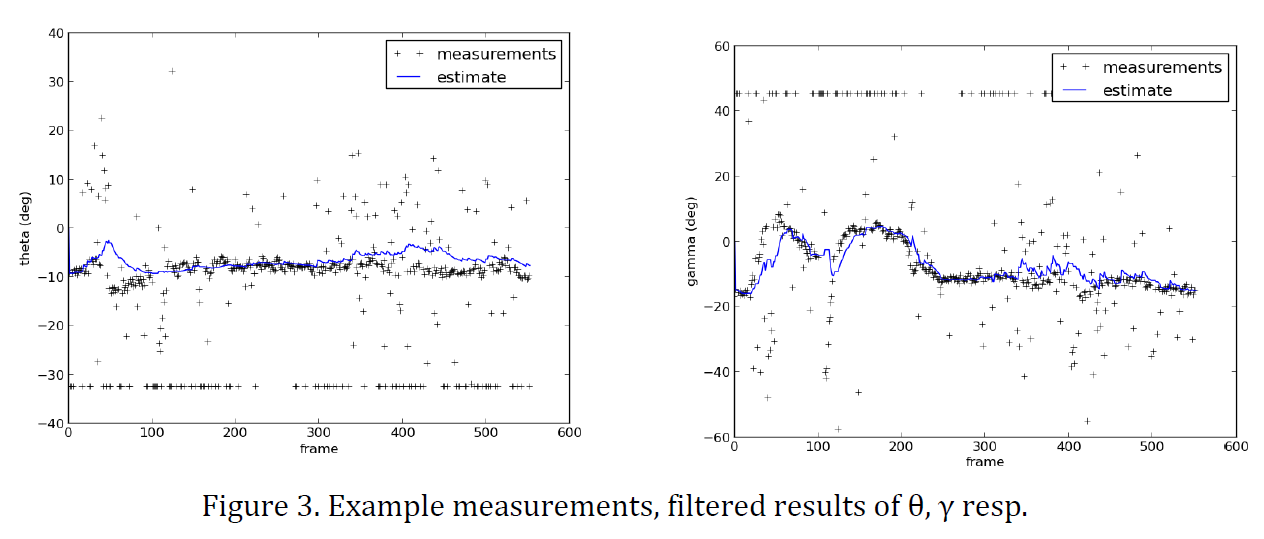


Figure 3. Example measurements, filtered results of θ, γ resp.

The Kalman filter is a custom 1-D filter that follows standard implementations. It is applied to the radian angle outputs of θ and γ separately. We initially tried implementing a 2-D filter, however the intrinsic motions that cause θ and γ to change are not inherently linked. The angle θ, or pitch, changes on breaking or acceleration.

The angle γ, or yaw, is from turning, or failing to turn in relation to the road. We ended up getting better results filtering them separately. We also tried to start with a state update model based on standard Newtonian velocity and acceleration.

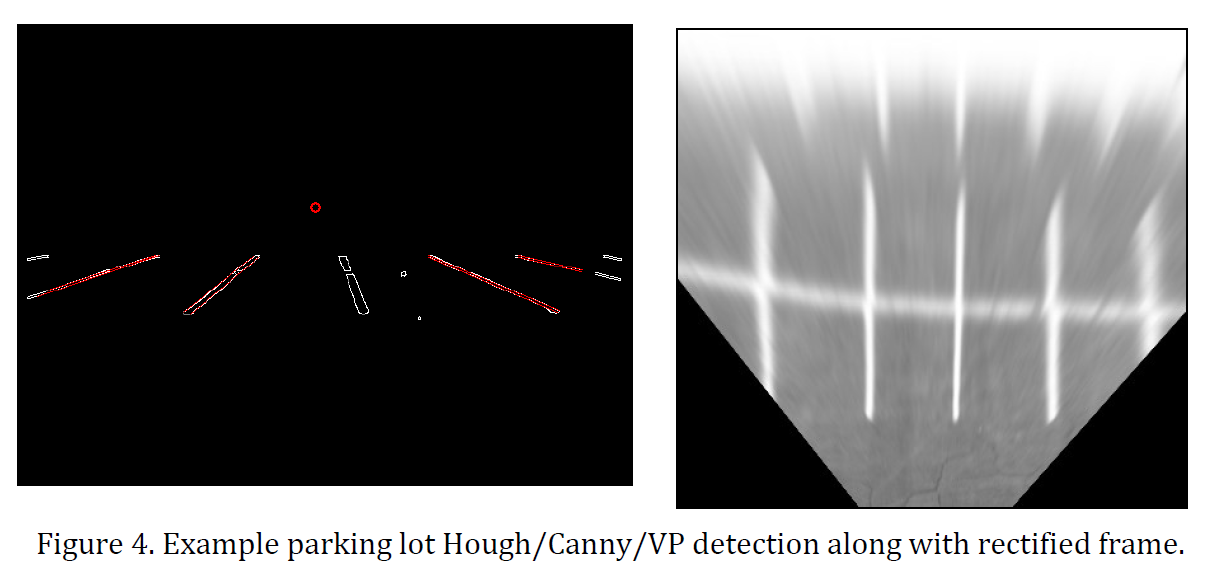
However, this did not work very well. Since our data is noisy and sudden – mimicking impulses, like from bump in the road – the velocity and acceleration of the angles would often poorly predict future movement. Currently we use a null state update model on the predict stage which works fairly well. We are working on better state update models for motion given our system’s constraints, and so hope to further improve this stage.

### Plane-to-plane homography

Given the angles θ and γ, we have completely defined our camera relationship to the world coordinates. This allows us to perform a transform from the image plane to the road plane, negating the effects of perspective in measuring distance to objects. Our model of the relationship between world, X, and camera coordinates, Xc, is given by the equation 2 in the vanishing point section [1].

We had difficulty in directly applying Nieto’s road-plane solution directly (where Y =

0). In order to rectify our problem, we performed a simple 90° rotation downwards offset by our calculated θ and γ. We also translate in order to have a view from far above. These values were mostly resolved experimentally. The camera intrinsic matrix, K, was found through the camera calibration process.



An ideal example of this transformation is shown above. The original image was taken from an empty parking lot.

### Bayesian segmentation

In our point of view, each frame of the input video has three main elements which are lane markers, pavement, and objects. We use the same definition for each element as

Nieto [1] paper, in which lane markers are set of pixels that have bright white color, pavement is a set of pixel represent the surface of the road, and objects category contains dark color pixels from the shadows of the cars, wheels, etc. In addition, we also use the undefined class as a regulation factor to maintain the convergence of the parameter estimation step explained later.

8

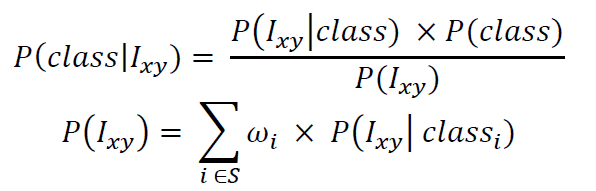
The Bayesian segmentation step uses the intensity of a pixel to calculate the probability of one category at that pixel. Different from the method in Nieto’s paper, we use only the intensity feature, eschewing the spatial feature information of a pixel. Therefore every pixel with the same intensity will be assigned the same probability of being a lane maker, pavement, object, and undefined element. Our method turns out to be a computational advantageous since we can implement it using a histogram based scheme. With this method, probability calculation time no longer grows with the number of pixels in a frame.

However, to simplify even more, we assume that each category follows a Gaussian distribution. Input frames are collected many different scenarios, with differing lighting conditions and pavement types, and noise in the desired features is ubiquitous. Although the color of a lane marker is generally constant, the added noise tends to make its color non-uniform with a bell-shape distribution. The same phenomenon applies to the other categories. In fact, our implementation of the method with this assumption works without any significant problem. This assumption mathematically eases the path for calculating the probability we need, since the Gaussian distribution is easy to represent by sigma and μ. Due to this, the maximization step in session 3 is much more direct.

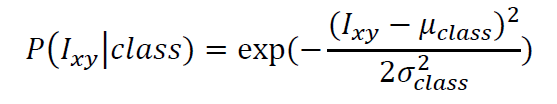
Bayesian theory

To begin the mathematical explanation, we borrow Nieto’s notations [1]. The set of four classes is S = {P, L, O, U}, where P is the pavement class, L is the lane marker class, O is the object class, and U is the undefined class. The feature 𝐼𝑥𝑦 is the intensity of the pixel at position (x, y) in the current frame.

To determine the representation of a class at a particular pixel, we calculate the posterior probability (𝑐𝑙𝑎𝑠𝑠|𝐼𝑥𝑦 ) at each given pixel as follows, given that 𝑃(𝑐𝑙𝑎𝑠𝑠) is the prior probability of a category, and 𝜔𝑖 is the weighting factor of how much a category contributes to an intensity value.



Given the assumption of Gaussian distributions, (𝐼𝑥𝑦 |𝑐𝑙𝑎𝑠𝑠) is calculated as follows.



In our method, the prior probability of a class is currently set to be equal the weighting factor without spatial information. In other words, each pixel has the same prior probability of each class. This simplification saves significant computational time.

Initializing parameters

The parameters that we need throughout the segmentation procedure are 𝜔, sigma and μ for each category.

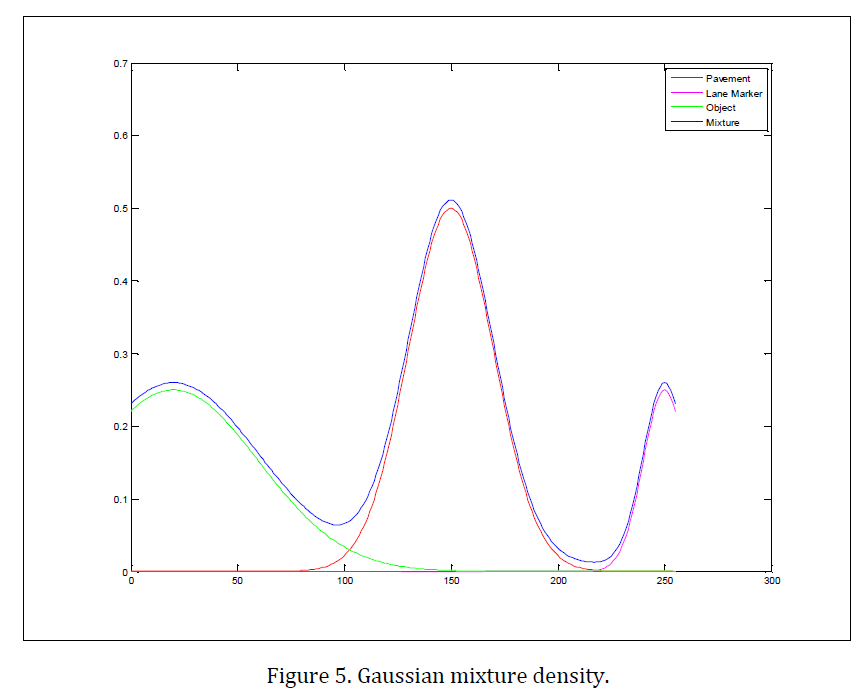
The initializing parameters step greatly affects the performance of the Bayesian segmentation. Our experiments show that non-suitable initial parameters cause miss-segmentation for every class. In contrast, appropriate initial values lead to a fast convergence to the true parameters. The explanation for this behavior is that the parameters are updated each frame through the Expectation Maximization algorithm, and poor initial estimates converge much more slowly to the true values, or even fail to converge.

To initialize the parameters, we first find the sigma and μ for the pavement class. We use the method described in section 4.2.3 of Nieto’s paper [1] to get the two values.

After that, sigma and μ for lane marker and object will be derived from them. In detail, 𝜇𝑂 ≈ 𝜇𝑃 − 3𝜎𝑃 and 𝜇𝐿 ≈ 𝜇𝑃 + 3𝜎𝑃.

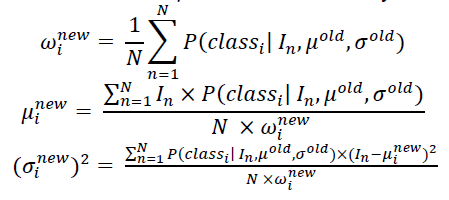
Updating parameters using Expectation Maximization

Figure 5 illustrates our assumption of Gaussian distributions for each class of elements, with the mixture being the observed image.



The EM algorithm is a general method of finding the maximum-likelihood estimate of the parameters of an underlying distribution from a given dataset when the data is incomplete or has missing values [2]. In our case, the application of EM algorithm falls to the category of “optimizing the likelihood function is analytically intractable but when the likelihood function can be simplified by assuming the existence of and values for additional but missing (or hidden) parameters”.

In summary, the parameters ω, σ and μ are EM estimated by:



Output of the Bayesian segmentation

For the purpose of object detection, the output of this step is the Object class posterior probability set which contains the probability at each pixel of a frame. We call this output the probability image. We threshold the probability image to eliminate weak pixels that have low chance for the presence of an object. The final output is a binary image with each pixel representing an object present, or no object present, encoded as 255, and 0, respectively.

During the Bayesian segmentation process, we make one exception in that pixels with intensity of 0 or 255 will be cut by 90% of the present before going to the EM step. The reason for this is that the noise present after the homography step is significant, and most of the noise is 0 or 255 in intensity. As a result, we improve the accuracy and the convergence time of the whole procedure.

## Tracking

### Morphology opening

Morphology opening consists of a morphology erosion step followed by a morphology dilation step. Basically, the erosion step using a structure element cleans up small processing noise after the Bayesian segmentation step. The erosion may break some weak connections between parts of an object. However, the dilation using the same structure element bridges those connections again to form solid blobs. A structure element in morphology is simple as a small solid circle that does AND / OR operators all over the subjecting image.

The following Figure x show how a morphology opening works in order to highlight blob area and eliminates small noise.

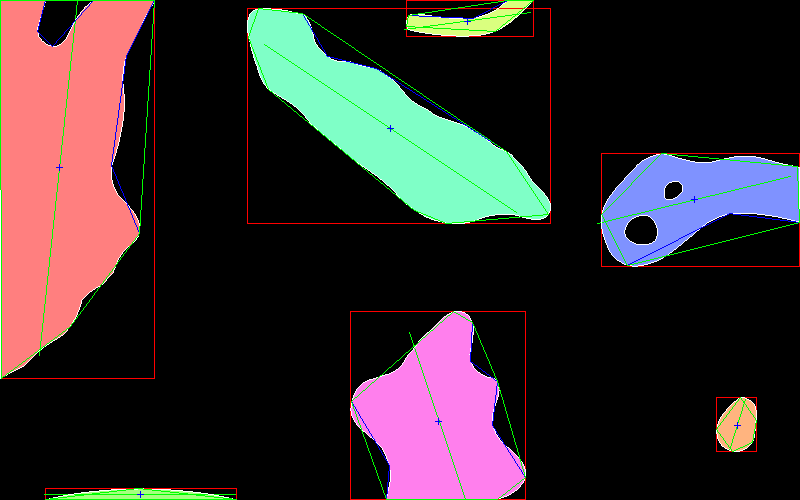
 

Original – dilation – erosion – final blob

### Blob detection

The blob detection step aims to find the central position of each blob. At first, we worked on finding the lowest point of each blob to represent for a blob. But later, the lowest point turned out to be quite unstable than the central point of a blob. Therefore, we decided to use the central point at the end.

We use the built in function in OpenCV. The idea of this function is tracing the contour of each solid blob, and then calculates the central position as the center of the outer-most rectangle. The Figure z below illustrates blobs, bounding rectangles (red boxes), and the central position (blue crosses).



<https://code.google.com/p/cvblob/>

The central position of each blob which is a pair of (x; y) pixel is then saved into a link list as an attribute of a node in the list. As observed by us, the central position of an object was affected by the processing noise coming from Bayesian segmentation, thresholds, and morphology steps. Therefore, the overall noise disturbed pretty much on the output of the blob detection. Not only that, the uneven distribution of light intensity on the street sometimes makes small parts of the pavement very dark and look like the shadow of an object.

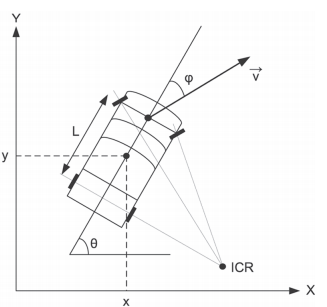
The second problem can be mitigated by setting a threshold for the width and the length of a ‘valid’ blob. In this manner, the meaningful blobs should not have very small width or very long length. We choose the threshold based on our observation of noise blob from the pavement and grass along the street. Currently, the length of a valid blob is less than 2/3 of the image’s length, and the width of a valid blob is X pixels. The first problem with processing noise will be address in the EKF position filtering session.

The Figure x below illustrate the before blobs and after threshold bobs detected.

### EKF position filtering

The previous work on vehicle tracking such as in Ess paper [citation] recommends using the Extended Kalman filter since it is the filter for non-linear system and a successful model for car were built in 1994 by Cameron and Proberdt. This model is so called the Ackermann steering model. The model employs the state vector [ , ]T and the update equation f where

=



Ess’s picture on illustrating the concept of Ackermann model

Variables in the Ackerman model are:

, : Position of a vehicle

: Direction of a vehicle

: Velocity and the turning angle of the vehicle from its previous direction

: Acceleration of the vehicle

Among these six attributes of the state vector, x and y attribute have the continuous input from the blob detection step. Other four attributes are used to balance the system and no correction is made on them. At first, we want to extract the velocity from EKF model as an estimation of true velocity. However, without any input to monitor its value, the velocity behaved un-realistic.

Like classic Kalman filter for linear system, the Extended Kalman filter reads measured positions from the blob detection step and predicts a true position. However, while KF uses the identical update function f for internal matrix A, EKF calculates A as the partial derivatives from f. In short, the following mathematically equations summary steps for updating positions using EKF.

EKF equations here

The most ambiguous factors in the EKF model are the process noise Q matrix and measurement noise R matrix. Q is a 6 x 6 matrix with two non-zero components and and R is a 2x2 matrix. The values of Q and R depend on specific system and have no general form to calculate. In our case, we estimated Q and R by tuning the matrices until we got a stable EKF system. We finalize Q and R with the following matrices:

R =

Q =

EKF performs excellent on predicting stables value for position (, ). The Figure x clearly shows the difference between the measured positions and the filtered positions of one particular blob.

EKF result image here + comments on the figure

### Direction and velocity estimation using linear fit

As the accuracy issue in the EKF session, the direction and velocity of a detected object need to be independently calculated from consecutives positions since the EKF does not provide a precise and . In our solution, we implemented a linear fitting algorithm to calculate the steering angle. The velocity is updated every dt amount of time by subtracting the position before and after this dt. In the current solution, both steering angle and velocity have a delay time of 1 second.

Picture of steering angle linear fitting + comments

Picture of velocity estimate + comments

### Vehicles management

In reality, a vehicle takes time to enter to our camera vision area and pavement noise blobs enter and exit the area in a short time. Therefore, we use a simple scheme that a blob presents long enough will be count as a valid object,; otherwise, ignore the short time blobs.

## Collision prediction

* Definition of dangerous situations
* Elements to predict dangerous situations
* Accuracy of those elements from previous estimation
* Decision on alarming situations
* Remaining unsolved dangerous situations

## Audio alert system

**References**

[1] NHTSA’s National Center for Statistics and Analysis, “TRAFFIC SAFETY FACTS 2011 Data.” U.S. Department of Transportation, May-2013.

[2] “In-depth investigations of accidents involving powered two wheelers.” MAIDS, Sep-2004.

[3] Leo J. Gugety, “Situation Awareness during Driving: Explicit and Implicit Knowledge in Dynamic Spatial Memory,” *Journal of Experimental Psychology*.

[4] “Toyota | Toyota Develops New Pedestrian Safety Technology.” [Online]. Available: <http://pressroom.toyota.com/releases/toyota+new+pedestrian+safety+technology.htm>. [Accessed: 15-Oct-2013].

[5] “DMM 22BUC03-ML Monochrome Board Camera.” The Imaging Source, LLC, 02-Aug-2013.

[6] “DARPA Grand Challenge (2005),” *Wikipedia, the free encyclopedia*. 16-Sep-2013.

[7] Adithya Naresh, “Light Detection And Ranging,” Oct. 2009.

[8] Marcos Nieto, Jon Arróspide Laborda, and Luis Salgado, “Road environment modeling using robust perspective analysis and recursive Bayesian segmentation.”

[9] “Mobileye - Our Vision. Your Safety.” [Online]. Available: <http://www.mobileye.com/>. [Accessed: 14-Oct-2013].

[10] “BMW launch 2013 - Mobileye.” [Online]. Available: <http://www.mobileye.com/markets/oem/oem-launches/bmw/bmw-launch-2013/>. [Accessed: 17-Oct-2013].

[11] “Rear Camera Applications - Mobileye.” [Online]. Available: <http://www.mobileye.com/technology/applications/rear-camera-applications/>. [Accessed: 17-Oct-2013].

[12] Andreas Ess, Konrad Schindler, Bastian Leibe, and Luc Van Gool, “Object Detection and Tracking for Autonomous Navigation in Dynamic Environments.” [Online]. Available: <http://ijr.sagepub.com/content/29/14/1707.short>. [Accessed: 9-Oct-2013].

[13] “Aptina Advances global Shutter Technology For High Speed Surveillance, Inspection And Scanning,” 28-Mar-2012. [Online]. Available: <http://www.aptina.com/news/press/aptina_advances_global_shutter_technology_for_high_speed_surveillance,_inspection_and_scanning/>. [Accessed: 12-Oct-2013].

[14] “Micro Air Vehicle Cameras.” [Online]. Available: <https://pixhawk.ethz.ch/electronics/camera#pixhawk_camera_for_gumstix_overo_com>. [Accessed: 16-Oct-2013].

[15] “MT9V032 LVDS camera board | [danstrother.com](http://danstrother.com/),” 14-Jan-2011. [Online]. Available: <http://danstrother.com/2011/01/14/mt9v032-lvds-camera-board/>. [Accessed: 10-Oct-2013].