Cornell Cup
Team SAFE

Interim Report, 3/24/2014

Nikolas Davis
Cuc Duong
Tuan Hoang
Matt Stehr

Table of Contents

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 system's – and so our image plane's – 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.

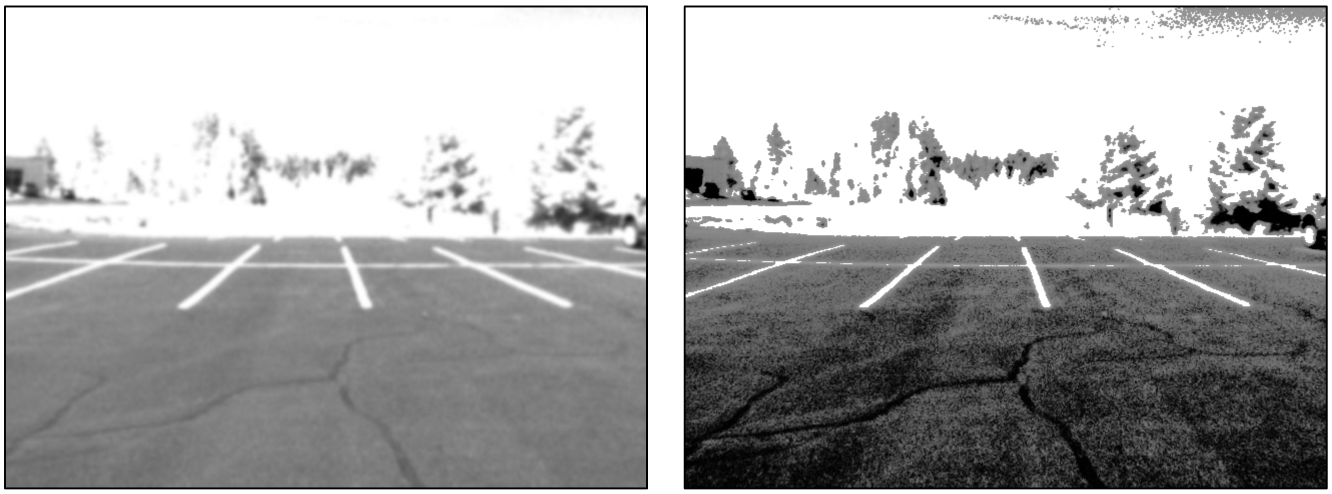
Figure 1. Example before and after of histogram equalization.

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:

$$y_i = 2_{x_i} - (x_{i-\tau} + x_{i+\tau}) - |x_{i-\tau} - x_{i+\tau}|$$

Eq. 1 from Nieto [1]

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.

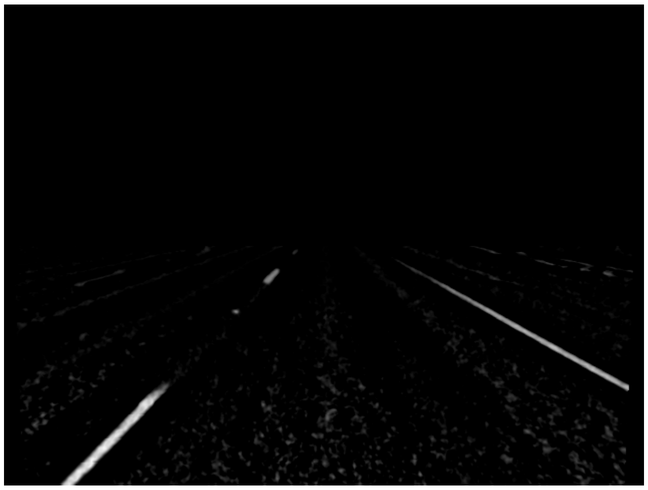
Figure 2. Example of lane marking filter with a dynamic τ adjusting from 10 – 20.

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].

Eq. 2: $x = KX_c = K(R|-Rc)X = PX$

Eq. 3: $x = P(X, 0, Z, 1)^T = (p_1, p_2, p_3)(X, Z, 1)^T = Hx'$

Eq. 4, 5: $\theta = \arctan\left(v'_{z,2}\right)\,;\ \gamma = \arctan(-\frac{v'_{z,1}}{cos\theta})$

Eq. 6: $v_z = K(R|-Rc)(0, 0, 1, 0)^T = K(tan\gamma cos\theta, tan\theta, 1)^T$

Eq. 7: $v'_z = K^{-1}v_z$

Equation 2 links the world coordinates, X, to our camera coordinates, X_c 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,

v_z, 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/).

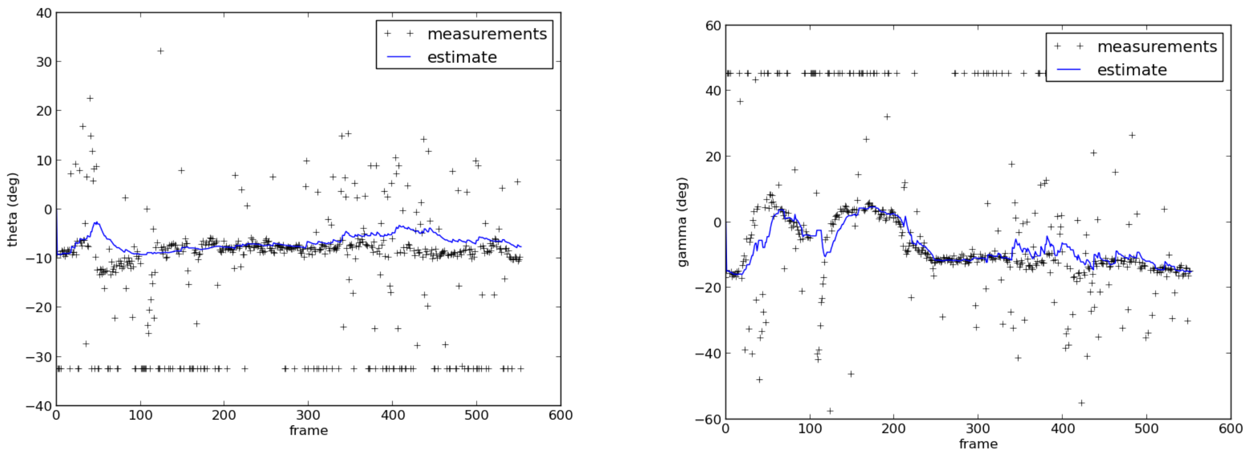

Figure 3. Example measurements, filtered results of θ, γ resp.

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, X_c, 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.

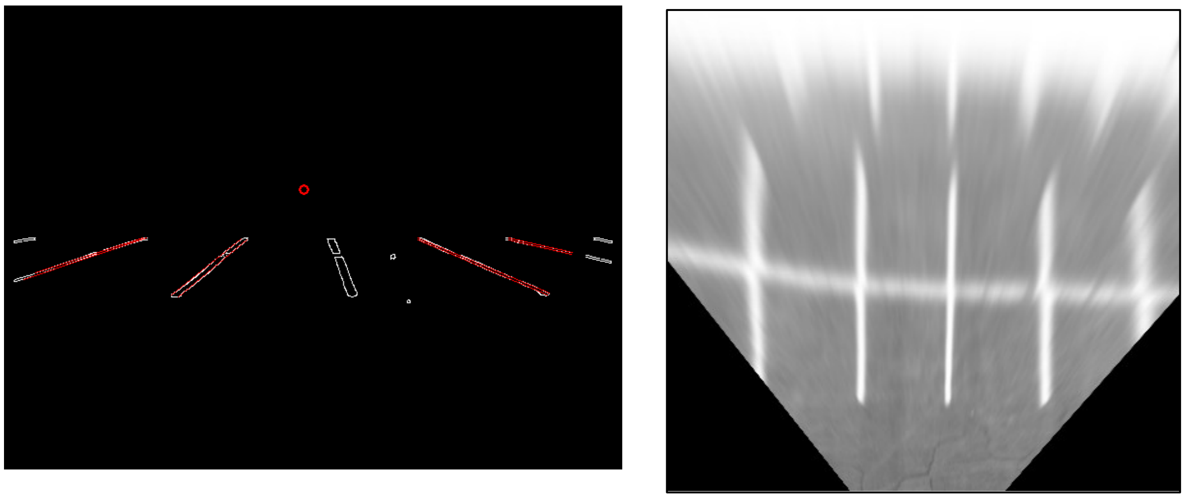
Figure 4. Example parking lot Hough/Canny/VP detection along with rectified frame.

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.

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 I_{xy} 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 $P(class|I_{xy})$ at each given pixel as follows, given that $P(class)$ is the prior probability of a category, and ω_i is the weighting factor of how much a category contributes to an intensity value.

$$P(class|I_{xy}) = \frac{P(I_{xy}|class) \times P(class)}{P(I_{xy})}$$

$$P(I_{xy}) = \sum_{i \in S} \omega_i \times P(I_{xy}|\, class_i)$$

Given the assumption of Gaussian distributions, $P(I_{xy}|class)$ is calculated as follows.

$$P(I_{xy}|class) = \exp(-\frac{(I_{xy} - \mu_{class})^2}{2\sigma_{class}^2})$$

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, $\mu_O \approx \mu_P - 3\sigma_P$ and $\mu_L \approx \mu_P + 3\sigma_P$.

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.

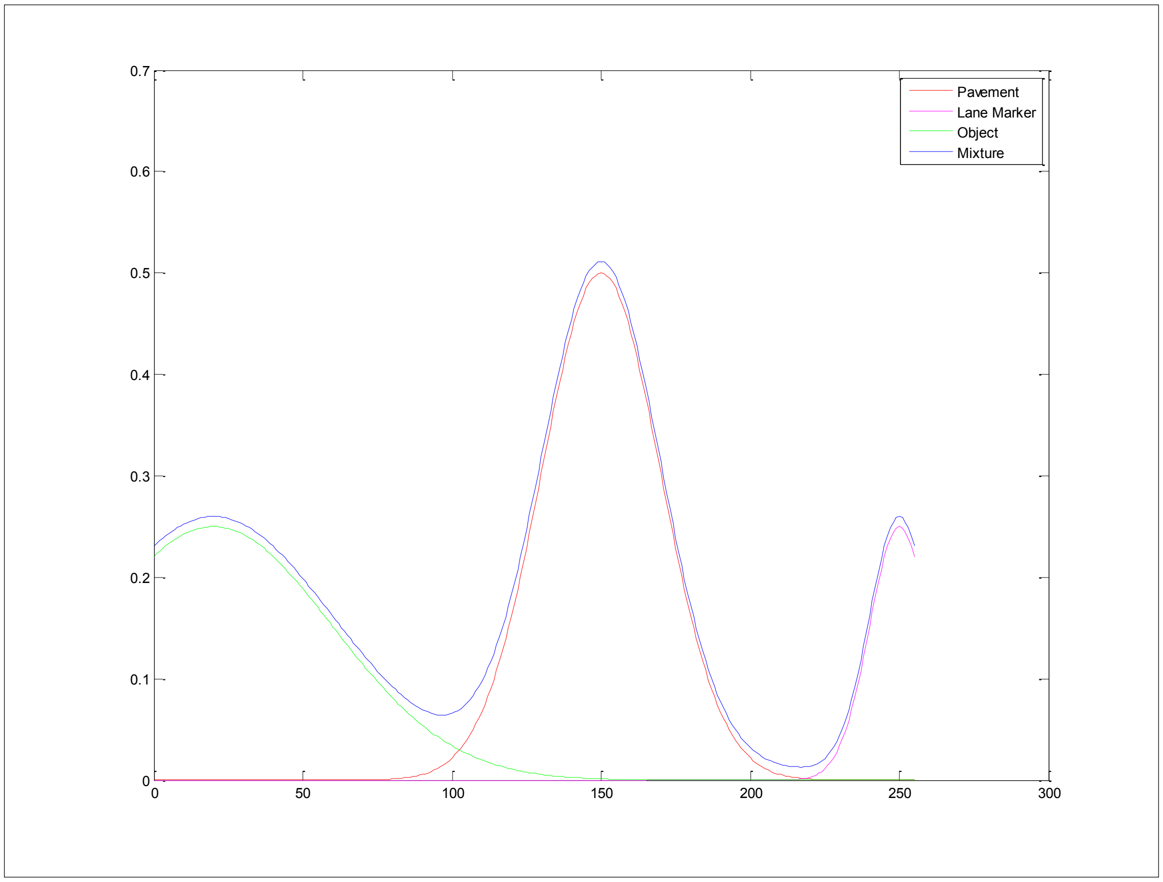

Figure 5. Gaussian mixture density.

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:

$$\omega_i^{new} = \frac{1}{N}\sum_{n=1}^{N} P(class_i |\, I_n, \mu^{old}, \sigma^{old})$$

$$\mu_i^{new} = \frac{\sum_{n=1}^{N} I_n \times P(class_i |\, I_n, \mu^{old}, \sigma^{old})}{N \times \omega_i^{new}}$$

$$(\sigma_i^{new})^2 = \frac{\sum_{n=1}^{N} P(class_i |\, I_n, \mu^{old}, \sigma^{old}) \times (I_n - \mu_i^{new})^2}{N \times \omega_i^{new}}$$

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.

Blob detection

This has yet to be integrated, along with the higher-level tracking and alert blocks.

Works cited

[1] M. Nieto *et al.*, “Road environment modeling using robust perspective analysis and recursive Bayesian segmentation,” *Machine Vision and Applications*, vol. 22, no. 6, pp. 926-945, Nov. 2011.

[2] J. A. Bilmes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," *Int. Computer Science Institute*, vol. 4, no. 510, pp. 126-140, Apr., 1998.