Localization on Freeways using the Horizon Line Signature

Nghia Ho[1] and Punarjay Chakravarty[2]

[1]nghiaho.com
[2]KU Leuven, ESAT-PSI, jaychakravarty.com

Abstract— **We explore the horizon line as a viable signature for vision-based localization on a freeway. Our proposed system uses a single forward facing camera mounted on the dash of a vehicle. Preliminary testing shows successful localization for 10 video sequences conducted on a freeway, travelling up to 100km/h, under varying weather conditions (sunny/cloudy/rainy). A particle filter is employed for localization. For the motion model, we use a purely vision-based method that counts the number of lane markings per frame. Overall, we conducted over 240 km of localization experiments.**

I. Introduction

Localization is a fundamental requirement in robotics. From indoor mobile robots wandering university labs [1] to experimental autonomous vehicles that traverse long distances without human intervention ([2], [3]), any time a robot needs to navigate between places in an environment, it needs to solve the localization problem of determining where it is in the world.

Whilst the Google car and the DARPA Grand Challenge vehicles used an array of sensors including GPS, inertial sensors, radar, laser range finders and cameras, recent research ([4], [5], [6], [7], [8], [9]) has focused on localization using vision alone.

Vision sensors are attractive due to their relatively low cost and passive operational nature. Images capture very rich information about the environment, albeit requiring more complex processing. Vision-based localization has the potential to operate with higher accuracy in GPS denied environments like urban canyons and tunnels. Google, through its Street View service, provides a vast database of mapped urban streets, which can be used as reference sequences for visual localization. Vision is also the modality with which we humans localize. For these reasons, we explore vision-based localization and the focus of this paper is visual localization in outdoor urban environments, along previously driven routes.

Any vision-based localization is essentially an image matching problem. Illumination differences, seasonal changes, view point variations, man-made environmental modifications and occlusion make this a non-trivial problem. The ideal algorithm for extracting a signature that can be matched between images should be able to cope with such issues and yet be highly discriminative at the same time.

Techniques for vision-based localization using image features can be divided into local and global features. With the former, many local features such as SURF, SIFT and FAST ([4] and [5]) are extracted and matched between images, whereas with the latter, one global signature is matched between images ([7] and [8]).

Despite some robustness of local features to viewpoint and scale changes, they can fail when faced with extreme seasonal change. This problem is addressed by [6] by a brute-force method, with imagery from each location over many seasons. [7] and [8] use global features on panoramic imagery for outdoor mobile robot localization. [8] uses patch normalized images, matched in the frequency (Fourier) domain, whereas [7] extracts a Haar wavelet signature describing each panoramic image. Patch normalized images are also used by [9] to demonstrate localization across challenging image sets that span both day and night.

A different way to solve the visual localization problem is to take a higher level, semantic scene-understanding approach to localization.

Scene understanding could be at a pixel level, where each pixel is classified into one of many classes ([10], [11], [12]), or at an object level, where a classifier is able to recognize an object using image features and draw a bounding box around it [13].

Pixel level road scene classification is described by [10], who use a Convolutional Neural Network trained on weak, machine labelled ground truth to classify the scene into sky, road and building facades, and by [11], who in addition to classifying individual pixels into road, building, sky, tree, sidewalk, etc. based on motion and appearance features, also use a CRF to model pairwise and higher order relationships between neighbouring pixels. [12] used road scenes classified at a pixel level for localizing a vehicle in urban Japan.

Traffic signs dotting streets and highways at frequent intervals could be used for object level classification to aid localization systems, and [13] describes such a system, that classifies traffic signs and localizes them on 3D maps of the road created while driving through them.

Another possible feature found on residential streets are street numbers, and Google looked at recognizing house numbers to improve Street View localization [14].

We look at using the horizon line as a signature because it is stable as long as the environment does not drastically change structure. It is robust to daytime illumination changes and being a small one-dimensional signature (we work on downsampled images), is fast to compute and compare.

The horizon line has previously been used by [15] and [16] to match skylines in real images to synthetically generated

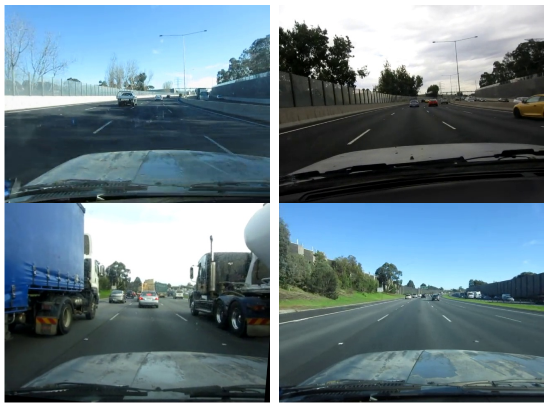


Fig. 1: Challenges of our datasets: weather and seasonal changes, occlusion from vehicles and inconsistent camera extrinsics. Each row corresponds to a matching pair.

ones from 3D models of the landscape. [15] uses panoramic images (taken with cameras with fisheye lenses) to outperform GPS localization in urban canyons, while [16] matches mountain skylines from tourist photos in Switzerland to synthetically generated skylines from a digital elevation model of the country. Both these techniques involved relatively clear views of the horizon with little occlusion: one because of high altitude and another because of specialist hardware: a fisheye lens looking straight up at skyscraper-rich urban cityscapes.

Our application uses a regular forward-looking camera mounted behind the windshield of a car, of the sort used by driver-aid systems like lane departure warning and pedestrian detection, fitted in high-end vehicles and after-market systems [17] available today. The view of the horizon from a forward looking view is not unimpeded and is often occluded by trucks and the use of such a signature for the localization of a vehicle along a freeway, to the best of our knowledge, hasn't been attempted before.

The horizon signature is fed into a temporal smoothing algorithm able to handle a multi-modal distribution of the location of the vehicle, the particle filter.

Lane markers along highways have a characteristic dot-dash-dot pattern, and the speed of travel along them determines the apparent frequency of this signal. We investigate the use of this frequency as a motion model for our particle filter. The use of lane markings for odometry is, to the best of our knowledge novel.

Some of the challenges we face are highlighted in figure 1. They include weather and seasonal changes , occlusion from vehicles and inconsistent camera extrinsics (yaw/pitch/roll).

The remainder of this paper is organized as follows. Section II gives an overview of our proposed method, describing the hardware and algorithms involved. Section III describes how we detect the horizon and match it to reference images. Section IV describes the lane marker counting algorithm that is used as the motion model for the particle filter. Section V covers our implementation of the particle filter. Experimental results are presented in section VI. Finally, conclusions and future work are discussed in section VII.

II. Proposed method

We now give an overview of how our localization system works. To collect our data, we used a Canon S100 point and shoot camera initially attached to the passenger-side sun visor with a custom-built hook and later to the windscreen using a suction mount. Video was collected over a span of 9 months, using 2 different vehicles, resulting in small discrepancies in viewing angles. The videos were recorded at 640x480 resolution at 30 fps. All video was processed offline on a laptop with an Intel i7-4700MQ CPU running at 2.40GHz and equipped with 16GB of RAM.

The localization system uses a particle filter to track the vehicle. The particle filter has two phases:

1) Motion Model: propagate the particles when new odometry data has arrived
2) Sensor Model: update the particle weights when new sensor data has arrived

The sensor data in this case is the detected horizon line, represented as a 1D feature vector. For odometry, we detect and count the number of lane markings over time in a bird's eye view/ground projection image and use this information to propagate the particles.

The pseudo code for our localization system is given in algorithm 1. The lane counting module runs every frame while the particle updates are done every 25 frames.

Algorithm 1 Overview of localization system
```
initializeParticles()
while grabVideoFrame() do
    runLaneCounting()
    if mod(frameCount, 25) = 0 then
        speed = speedFromLaneCounting()
        signature = detectHorizonLine()
        applyMotionModel(speed)
        applySensorModel(signature)
        estimateVehiclePosition()
        resampleParticles()
    end if
end while
```

All the code is written in C++ using OpenCV on Linux.

III. Horizon detection

The algorithm works on the assumption that the sky is always of a lighter intensity than the ground and scans downwards along each column from the top of the image and finds the first pixel that has a blue intensity value higher than a threshold value or has a vertical gradient higher than a threshold, which is marked as the horizon pixel. We found using only the blue channel to be adequate for all the datasets.

A pre-processing step of color reduction and downsamping is performed prior to the horizon line detection. The horizon detection algorithm is given in algorithm 2. The image coordinate convention we use throughout the paper for y=0 is

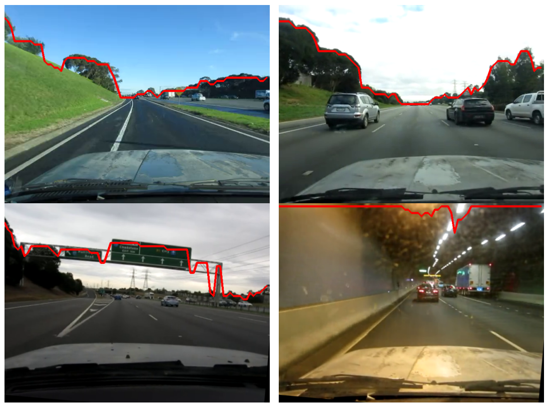


Fig. 2: Examples of horizon lines detected.

at the bottom of the image. Examples of detected horizons are shown in Figure 2. In general, the horizon detection algorithm has been found to work well in all parts of our datasets except when obscured by a truck or inside tunnels, where there is no sky-ground horizon to be detected.

Algorithm 2 Horizon detection and signature creation

```
 1: function horizonSignature(img : [640x480 RGB])
 2: horizon : integer array[0..79]
 3: tmp = colourReductionAndDownSize(img) // 80x60
 4: skyColor = findSkyColorAboveHorizon(tmp)
 5: for x = 0 to 79 do
 6:     for y = 59 to 30 do
 7:         horizon[x] = y
 8:         edgeStrength = |tmp(y,x,blue) - tmp(y-1,x,blue)|
 9:         if img(y,x,blue) < 100 OR edgeStrength > 80 then
10:             break
11:         end if
12:     end for
13: end for
14: leftSig = horizon[0..51] // two third overlapping
15: rightSig = horizon[28..79] // two third overlapping
16: return leftSig, rightSig
```

The horizonSignature function returns two vectors of 52 integers. Each vector is 2/3 the length of the original horizon line and overlap each other, starting from the far left and right. We refer to them as the left and right signature. This helps to improve matching when we are driving in different lanes and provides some robustness to vehicles occluding one of the lanes. The scores from the left and right signature are then averaged together as follows

$$score = (\boldsymbol{horizonScore}(leftSigA, leftSigB) + \boldsymbol{horizonScore}(rightSigA, rightSigB))/2$$

where A and B are the two images whose signatures need to be compared.

A. Matching

We use mean-sum-of-absolute-difference (MSAD) with shift correction to match the signatures. The shift correction is necessary due to the slightly different viewing angles between the datasets. To correct for both x and y shifts, we opted for a simple brute force 2D search of possible shifts within a window and find the best match. To speed the task up we used a pyramidal scheme. Using the pyramids we can constrain the search to ± 1 pixel at each level. Using a 3 level pyramid this translates to a maximum shift correction of ± 7, which was found to be adequate due to the heavy downsampling (8x) of the horizon signature in the pre-processing step. Since the signature is 52 integers, it divides nicely for the pyramid: 52, 26, 13.

The value returned from MSAD with shift correction is then fed into a sigmoid function to return a value between [0,1] for the particle filter. The weights and bias of the sigmoid were trained using ground truth for dataset #2. Dataset #2 contains 94 manually aligned images with dataset #1. 279 (93x3) positive and negative samples were generated from the ground truth and trained using logistic regression found in Python's scikit-learn package.

The matching function is described in algorithm 3. The WEIGHT and BIAS parameters used in our experiments are -2.10899 and 5.12172, respectively.

Algorithm 3 Horizon matching with shift correction

```
 1: function horizonScore(sigA : [0..52], sigB : [0..52])
 2: pyramidA[3] = generatePyramid(sigA, levels=3)
 3: pyramidB[3] = generatePyramid(sigB, levels=3)
 4: best = 0, estShiftX = 0, estShiftY = 0
 5: for level = 0 to 2 do
 6:     bestDiff = 0, bestShiftX = 0, bestShiftY = 0
 7:     for shiftY = -1 to 1 do
 8:         for shiftX = -1 to 1 do
 9:             diff = sumAbsDiff(sigA, sigB, estShiftX +
                shiftX, estShiftY + shiftY)
10:             if diff < bestdiff then
11:                 bestDiff = diff
12:                 bestShiftX = shiftX
13:                 bestShiftY = shiftY
14:             end if
15:         end for
16:     end for
17:     best = bestDiff
18:     estShiftX = (estShiftX + bestShiftX) x 2 // upscale
19:     estShiftY = (estShiftY + bestShiftY) x 2 // upscale
20: end for
21: x = (double)best / 52 // average
22: score = 1/(1 + exp(-x*WEIGHT + BIAS)) // sigmoid
23: return score
```

The function generatePyramid recursively downsamples the horizon signature by two using averaging. Since the signature represents a line in a 2D image the downsampling needs to be done for both x and y direction. So the averaging becomes

$$avg[i] = (A[i \times 2] + A[i \times 2])/4$$

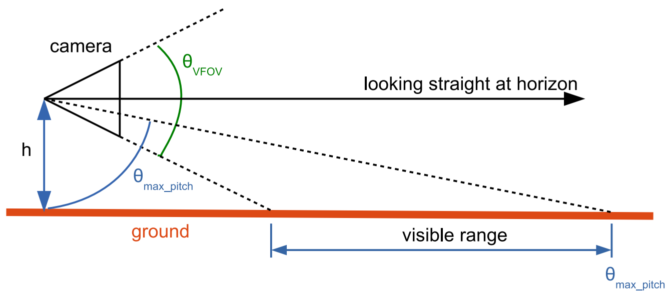


Fig. 3: Ground projection model

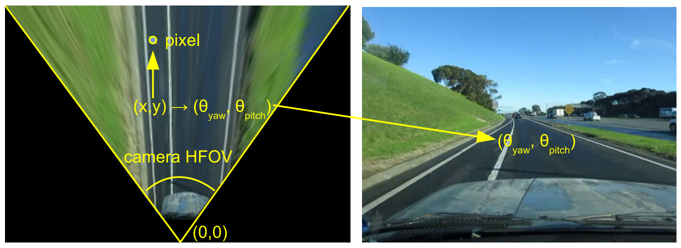


Fig. 4: Ground projection mapping

The sumAbsDiff function handles border issues by using the nearest valid array index.

IV. Vehicle Speed using Lane Marker Counting

The dot-dash-dot pattern of highway lane markers is utilized by our algorithm to determine the virtual speed of the vehicle in terms of lane markers per frame. This speed is then used as an odometer, to propagate the particles in the particle filter described in the next section. The lane marker detection and counting can be divided into the following steps, which are elaborated on next.

1) Ground view image generation
2) Shadow removal and lane marker detection
3) Lane marker counting for odometry

The following sub-sections describe these steps in more detail.

A. Ground View Image Generation

The ground view image is generated from the forward facing camera by applying a pre-computing mapping based on ray projection trigonometry. The model we use is shown in figure 3 and the mapping between forward and ground views is shown in figure 4. We assume the ground is flat and the camera is looking directly at the horizon.

B. Shadow Removal and Lane Marker Detection

The grayscale ground image is patch normalized to remove shadows and highlights from it. Patch normalization subtracts from each image pixel the mean value of pixels in a square patch around it and divides it by the standard deviation of pixels in that patch.

Lane markings are characterized as regions of high intensity surrounded by regions of low intensity and are extracted by a filter that searches for such an intensity profile [18].

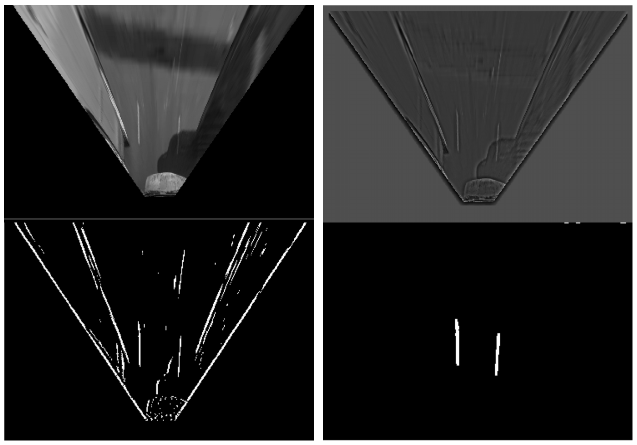


Fig. 5: Steps in the detection of lane markings. From top left, in raster scan order: Ground view projection, patch normalization to remove shadows and highlights, lane marker filtering, connected components analysis to get lane markings in regions of interest immediately in front of the vehicle, on either side of it.

The output of patch normalization is passed through this filter, which scans each row of the image and performs the following operation on each input pixel $x[i]$ to give the output pixel $y[i]$ of the filtered image as follows:

$$y[i] = 2x[i] - (x[i-\tau] - x[i+\tau]) + abs(x[i-\tau] - x[i+\tau])$$

where τ is the lane width. The filter is applied for a set of different τ values.

The output of this image is thresholded to give a binary image with the lanes shown in white. This binary output is dilated and connected components analysis is then used to filter out elements that are too small. Two rectangular region of interest masks (roughly the length of the lane markings) are then used to further filter out the components that are not likely to be part of the lane. These masks make the assumption that the vehicle is roughly centered in the lane.

Figure 5 illustrates the steps of the lane marker detection algorithm.

C. Lane Marker Counting for Odometry

The number of *on* pixels under these masks in the binary image are counted in each frame, and this number is stored in a rolling buffer of 100 frames. This 100-length buffer, containing the local appearance history of lane markings is used to find continuously varying maximum and minimum thresholds (80% and 20% of the maximum value in this buffer) for the presence of a lane marker. If the number of *on* pixels under the lane mask in the current frame exceeds the maximum value, a lane is considered to be present. The lane is considered to continue to be present until this number goes back under the minimum threshold.

The presence/absence of lane markings over time is a square wave signal, whose peaks tell us the virtual speed of the vehicle in terms of lane markers per frame.

In our experiments, dataset #1 is used as the reference sequence, and its lane marker speed is recorded for every 100th frame of the sequence. When matching other sequences to this reference sequence, the currently estimated lane marker speed is compared against the reference lane marker speed to find out the relative speed, which is then used to propagate each particle along the sequence. An important point to note is that we are comparing the frequency of the lane markers between the sequence to be localized and the reference sequence. We are not depending on the absolute frequency at any point in the sequence, which might change not just because of speed, but also because of lane type; for instance, the physical separation between the dots along entrance/exit lanes, main freeway lanes and tunnel lanes are all different.

V. Particle Filter

Our implementation of the particle filter is identical to the one described in [19].

A fixed number of 10,000 particles was used throughout the localization. Alternatively, we could use an adaptive method like KLD-sampling [19] but we found a fixed number to work well while still being fast. Most of the datasets have roughly 32,000 frames so this represents about 30% coverage.

The particle with the highest score is taken to be the best estimate of the vehicle's position at that given time. We tried a weighted average but found it was not as accurate.

VI. Experimental Results

We collected a total of 11 datasets on the Monash Freeway (Melbourne, Australia) for a single direction of travel. The distance for each run, estimated by Google Maps is 24.6 km. The data collected spans across a 9 month period, with the first 6 sets collected in July (winter) and the remaining 5 in March (autumn). The weather conditions varied from bright and sunny, to overcast and rainy. The freeway is three laned for the most part, with short sections having four. We mainly drove between the first and the second lane, with later datasets having brief instances of driving in the right-most overtaking lane. Two different vehicles were used for the winter and autumn datasets. A Google Map of the route taken with localization results overlayed is shown in figure 6.

To get a qualitative measure of the localization accuracy, ground truth data was created manually by hand for each dataset at every 300th frame. We use the very first dataset collected as the reference for all experiments. Each dataset typically required 100 ground truth points marked. For positions in between the marked points we used linear interpolation. This was found to be accurate enough because 300 frames equates to 10 seconds of travel, which is short enough that we can safely assume near-constant speed. All localization error values stated are in terms of frame, that is the absolute difference with the ground truth frame.

We define the following measures for each dataset:

1) percentage of localization within ±30 frames of the ground truth

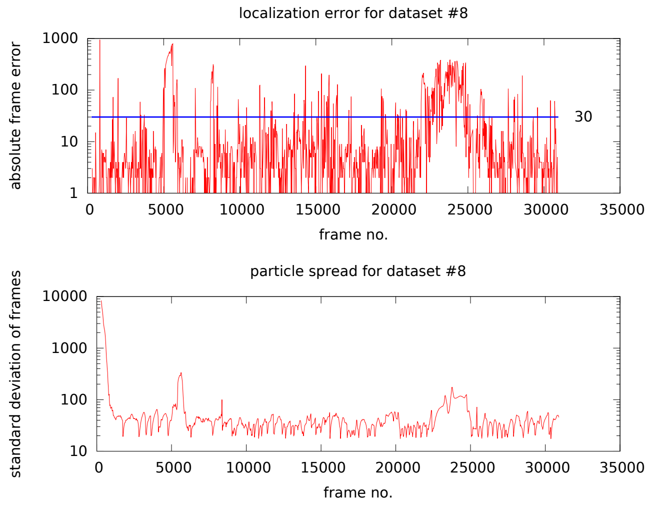


Fig. 7: Localization errors and particle spread for best performing dataset (#8).

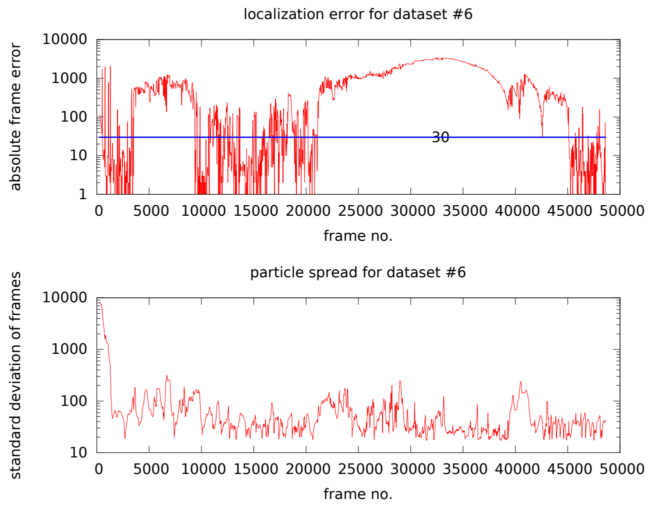


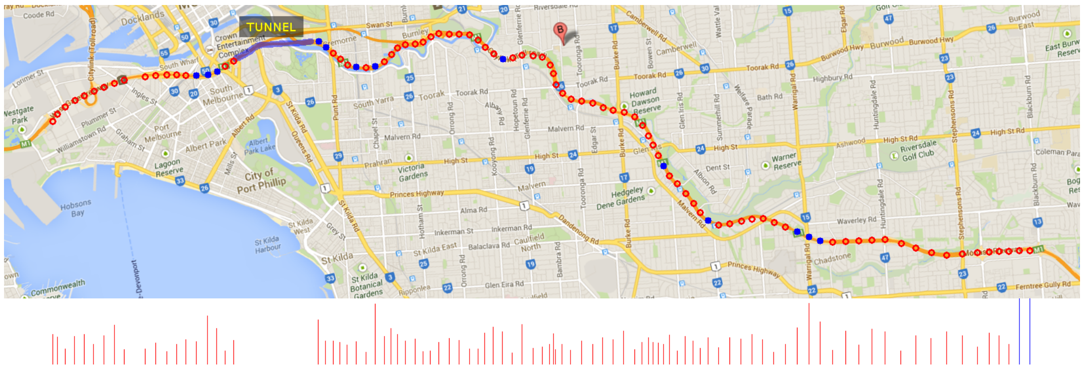
Fig. 8: Localization errors and particle spread for the worst performing dataset (#6). The particle filter loses track between frames 22,000 and 45,000.

2) median localization error
3) localization error of the last frame

(1) measures the number of particle updates that have localized *successfully*. We define a successful localization to be within ±30 frames of the ground truth, or one second in real life, which we believe is an acceptable measure of success. At 100 km/h, this is equivalent to $\pm 27.8m$.

(2) uses the median localization error instead of the mean value to better reflect our data. The mean would be skewed by the initial global localization stage, during which time the particles are spread all over the sequence. We could have removed the global localization stage from this analysis, but its length varies between datasets and would have required manual selection.

(3) is used to asses whether the localization has succeeded



Particle spread for each marked location directly above. The highest value of the bar is $\sigma = 100$ frames and occurs typically during global localization at the beginning. A blue line indicates a $\sigma > 100$.

Fig. 6: Google Map of Monash Freeway (Melbourne, Australia) with the result from dataset #8 overlayed on top. Localized positions are marked every 300 frames, with red circles indicating a correct match (within ±30 frames of the ground truth) and blue circles indicating an incorrect match. No ground truth was marked for the tunnel section. The estimated travel distance is 24.6 km.

Dataset	weather	date	video length (mins)	correct matches %	median frame error	last frame error
#1 (ref)	sunny	15/07/13	17:30			
#2	cloudy	16/07/13	17:03	0.77	5	2
#3	cloudy	18/07/13	17:55	0.69	9	3
#4	cloudy	20/07/13	17:55	0.75	8	2
#5	cloudy	23/07/13	19:00	0.60	14	75
#6	cloudy	25/07/13	27:10	0.27	450	1
#7	sunny	29/07/13	17:36	0.66	8	2
#8	sunny	30/07/13	17:15	0.82	5	1
#9	sunny	14/03/14	17:52	0.73	8	10
#10	dark overcast	15/03/14	18:00	0.75	8	13
#11	cloudy	17/03/14	18:22	0.68	9	1

TABLE I: Localization results for all datasets. Dataset #1 is used as the reference video. Dataset #6 has localization errors for a majority of the sequence, due to very heavy traffic causing problems for the image signature, but recovers towards the end.

all the way to the end of the freeway, or failed somewhere in between.

The experimental results are summarized in table I. The worst final frame localization error of 75 frames was for dataset #5 and the worst performing dataset in terms of percentage of correct matches was 6, where trucks occluded the horizon often. A still image from this dataset is shown in the bottom left image of figure 1. Trucks on both sides of the vehicle completely occlude the horizon line. The poor quality of horizon signature matching is also reflected in the cyan ROC curve for dataset #6 (figure 11).

The tunnel section exhibits an interesting behavior. When the vehicle enters the tunnel, there is no horizon and the horizon signature ends up being a flat line. The reason the particle filter is able to relocalize fairly quickly upon exiting the tunnel is because all the matches have very similar scores. Thus, the particles end up spreading themselves over the tunnel to reflect their uncertainty.

The localization errors (on a logarithmic scale) and the particle spreads over time are plotted for the best and worst performing datasets in figures 7 and 8. The blue line indicates the threshold for good localization (within 30 frames of ground truth).

Figure 9 shows some examples of matches and mismatches for dataset #10 (captured 9 months after the reference dataset #1) from the particle filter.

Our system takes about 36 ms to update the particle filter for 10,000 particles, which includes horizon extraction and matching. The lane counting takes about 7 ms. The system runs at 23 frames per second. Times are quoted for a single core as we have not multi-threaded our code. A video of the localization is available at http://youtu.be/ylMbMA8COu0.

A. *Horizon signature analysis*

We analyze the characteristics of the horizon signature by generating confusion matrices and ROC curves for all the datasets using the ground truth, which is marked every 300 frames excluding the tunnel section.

The confusion matrix for dataset #2 (obtained by plotting the match scores between images in #2 on the y axis and images in #1 on the x axis) is shown in figure 10.

It displays a strong diagonal response (white), implying that a majority of the ground truth images match between sequences. Low matches on the diagonal typically correspond to points along the sequence where the horizon is

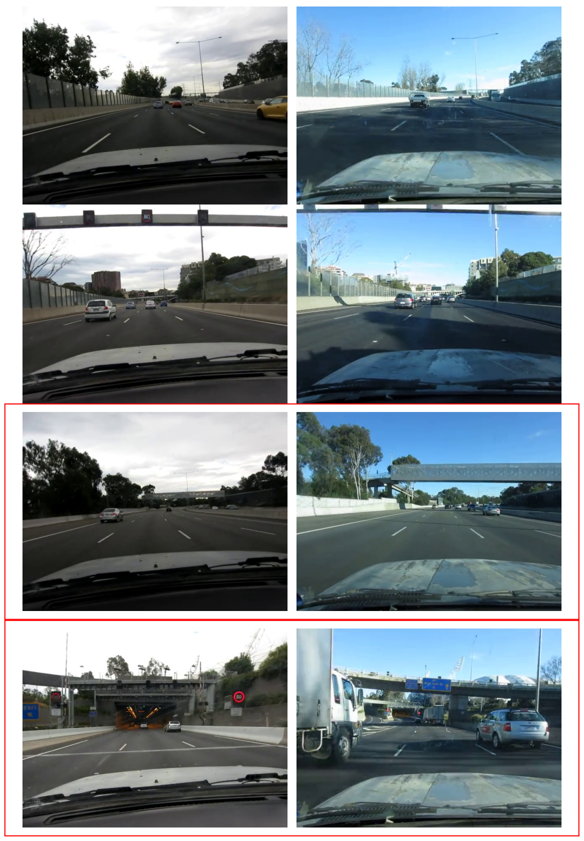


Fig. 9: Examples of matches and mismatches from dataset #10 (left column) taken 9 months after the reference dataset #1 (right column), using localization output from the particle filter. Mismatches are highlighted in a red box. The top rows highlights seasonal changes. Second row shows man made changes. The final two rows show the mismatches that seem close together, but are still 84 and 131 frames apart (approx. 62m and 97).

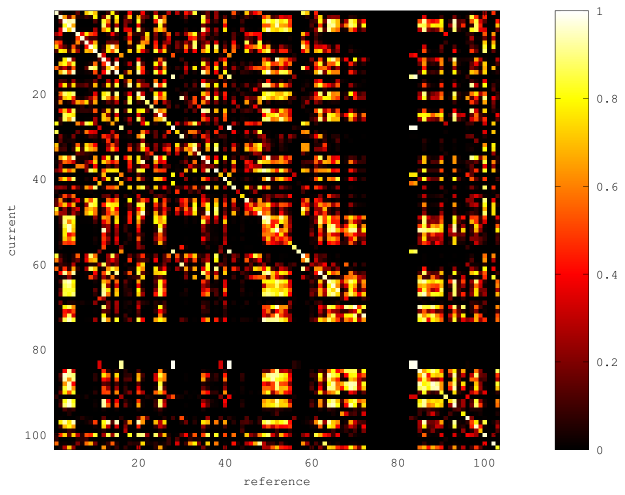


Fig. 10: Confusion matrix (94x94) for dataset #2

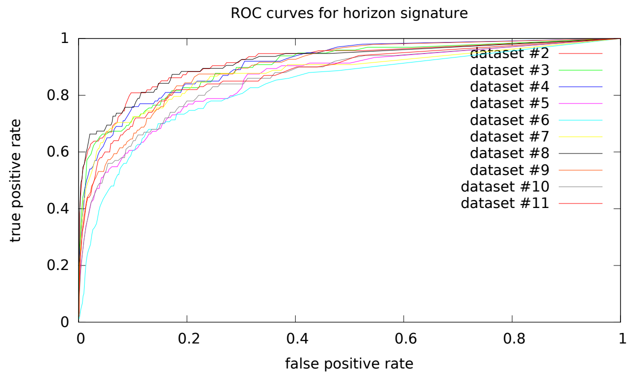


Fig. 11: ROC curves for all datasets

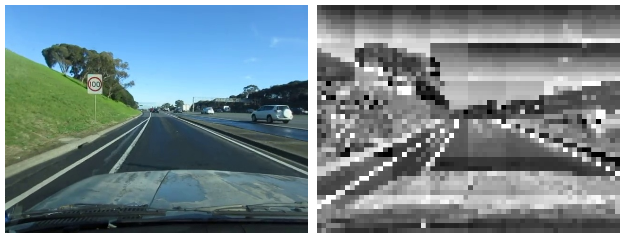


Fig. 12: Example of patch normalization using 8x8 patches. The right image is a downsized 64x48 image of the left image.

obscured by trucks. There are only a small number of white pixels off the diagonal, suggesting that the horizon is fairly distinctive for the environment, atleast outside the tunnel section (74-82 on the y axis), which is missing ground truth and is black in the figure. The confusion matrices for the remaining datasets (omitted given space constraints) have similar diagonal responses.

The ROC curves (figure 11) are similar between the datasets with a 70-80% true positive rate returning a 10-20% false positive rate.

We also compared the horizon signature with patch normalized image signature, matched using sum of absolute differences as used by [9] in their sequence SLAM algorithm. For a fairer comparison, only the top half of the image is used, which aims to eliminate some of the occlusion from vehicles. This gave slightly better results than using the entire image.

Patch normalization (figure 12) comprises of cropping and downsampling the image (to 64x24), and then normalizing each 8x8 patch in the image by subtracting its mean and dividing by its standard deviation.

A comparison of the ROC curves is shown in figure 13. The precision-recall curves of the two signatures are shown in 14. For the sake of clarity, we only plot the best performing dataset for each signature.

It is evident that the horizon signature out performs patch normalization overall (atleast in these freeway environments), even after cropping the top half of the image.

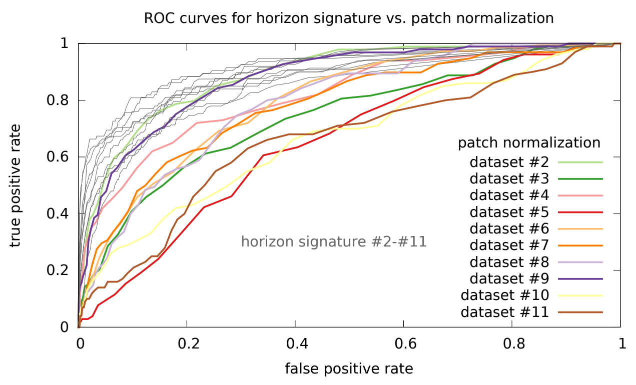


Fig. 13: Horizon signature vs. patch normalization for all datasets

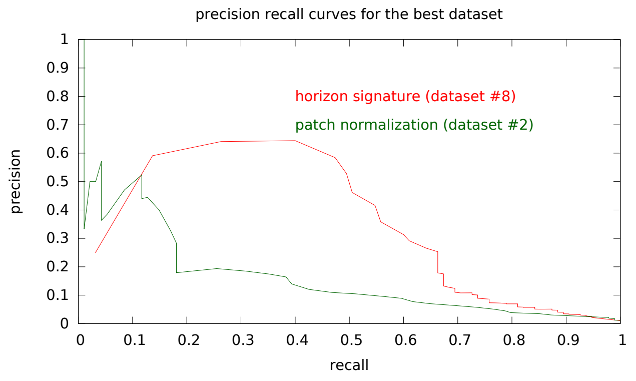


Fig. 14: Precision-recall curves for the two signatures on their respective best performing datasets

VII. Conclusions and Future Work

We have shown that the horizon signature coupled with a particle filter for temporal smoothing is a viable feature for localization in freeway environments. It is robust to daytime illumination changes, weather and seasonal changes, and to most dynamic objects in the scene. Trucks are problematic because they occlude a large part of the horizon, analogous to a lost GPS signal. Horizon-less tunnels also present a challenge to the system. Currently, the system's localization loses accuracy in the 1.6 km long tunnel in the datasets, does not completely lose track and locks on to the correct position soon after emergence. A different feature will need to be employed for more accurate localization within tunnels. Another drawback is that the horizon is only detectable during the day time.

We have also used the frequency of the lane markers as an odometry measure/motion model for propagating the particles in the particle filter.

The freeway, and built-up environments in general, have many other high level semantic features that we can use to augment our horizon signature. For example, we have done some preliminary investigation into using road signs as a feature. These have shown to be promising because they are unlikely to change over time and have strong colour contrast with the surroundings.

In addition to adding new features, we plan to test the system in non-freeway environments, specifically inner city streets and localize datasets we collect against Google's pre-existing Street View images.

References

[1] D. Fox, S. Thrun, F. Dellaert, and W. Burgard, "Particle filters for mobile robot localization," in *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and N. Gordon, Eds. New York: Springer Verlag, 2000, to appear.

[2] How google's self-driving car works. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works

[3] M. Buehler, K. Iagnemma, and S. Singh, Eds., *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic, George Air Force Base, Victorville, California, USA*, ser. Springer Tracts in Advanced Robotics, vol. 56. Springer, 2009. [Online]. Available: http://dblp.uni-trier.de/db/conf/darpa/urban2009.html

[4] H. Badino, D. Huber, and T. Kanade, "Visual topometric localization," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, June 2011, pp. 794–799.

[5] T. Botterill, S. Mills, and R. D. Green, "Bag-of-words-driven, single-camera simultaneous localization and mapping." *J. Field Robotics*, vol. 28, no. 2, pp. 204–226, 2011.

[6] C. Valgren and A. J. Lilienthal, "Sift, surf and seasons: Long-term outdoor localization using local features." in *EMCR*, 2007.

[7] N. Ho and R. Jarvis, "Vision based global localisation using a 3d environmental model created by a laser range scanner," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, Sept 2008, pp. 2964–2969.

[8] A. M. Zhang and L. Kleeman, "Robust appearance based visual route following in large scale outdoor environments."

[9] M. Milford and G. Wyeth, "Seqslam : visual route-based navigation for sunny summer days and stormy winter nights," in *IEEE International Conferece on Robotics and Automation (ICRA 2012)*, N. Papanikolopoulos, Ed. River Centre, Saint Paul, Minnesota: IEEE, 2012, pp. 1643–1649. [Online]. Available: http://eprints.qut.edu.au/51538/

[10] J. M. Alvarez, T. Gevers, Y. LeCun, and A. M. Lopez, "Road scene understanding from a single image," in *European Conference on Computer Vision (ECCV)*, 2012.

[11] P. Sturgess and K. Alahiri, "Combining appearance and structure from motion features for road scene understanding," in *British Machine Vision Conference (BMVC)*, 2009.

[12] J. Miura and K. Yamamoto, "Robust view matching-based markov localization in outdoor environments," in *IEEE/RJS Int. Conf. on Intelligent Robots and Systems*, 2008.

[13] R. Timofte, K. Zimmermann, and L. J. V. Gool., "Multi-view traffic sign detection, recognition, and 3d localisation." in *IEEE Workshop on Applications of Computer Vision (WACV)*, 2009.

[14] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," *CoRR*, vol. abs/1312.6082, 2013.

[15] S. Ramalingam, S. Bouaziz, P. F. Sturm, and M. Brand, "Skyline2gps: Localization in urban canyons using omni-skylines," in *IROS*, 2010.

[16] G. Baatz, O. Saurer, K. Kser, and M. Pollefeys, "Large scale visual geo-localization of images in mountainous terrain." in *ECCV (2)*, ser. Lecture Notes in Computer Science, A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds., vol. 7573. Springer, 2012, pp. 517–530.

[17] Mobileye. [Online]. Available: http://www.mobileye.com/

[18] L. S. M. Nieto, J. Arrospide, "Road environment modeling using robust perspective analysis and recursive bayesian segmentation," *Machine Vision and Applications*, vol. 22, no. 6, pp. 927–945, 2011.

[19] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.