Traffic Light Mapping and Detection

Nathaniel Fairfield Chris Urmson
{nfairfield, curmson}@google.com
Google, Inc.

***Abstract*— The outdoor perception problem is a major challenge for driver-assistance and autonomous vehicle systems. While these systems can often employ active sensors such as sonar, radar, and lidar to perceive their surroundings, the state of standard traffic lights can only be perceived visually. By using a prior map, a perception system can anticipate and predict the locations of traffic lights and improve detection of the light state. The prior map also encodes the control semantics of the individual lights. This paper presents methods for automatically mapping the three dimensional positions of traffic lights and robustly detecting traffic light state onboard cars with cameras. We have used these methods to map more than four thousand traffic lights, and to perform onboard traffic light detection for thousands of drives through intersections.**

I. Introduction

More than 50 million new cars are manufactured each year [Plunkett Research, Ltd., 2009]. Advanced onboard sensors, such as sonar, radar, and cameras, are becoming commonplace as part of commercial driver-assistance systems. With power steering, power brakes, standardized feedback from the CAN bus, cars are basically robots – they only need a brain. A key component of these augmented vehicles is the perception system, which allows the vehicle to perceive and interpret its surroundings.

Humans have engineered the driving problem to make it easier. For example, lanes are delineated by lines painted on the road, traffic lights indicate precedence at intersections, brake lights show when other vehicles are decelerating, and turn signals indicate the driver's intentions; all these cues are intended to simplify the perception task. Perception systems can use these driving aids, but in many cases they are able to use alternative sensing modalities, such as radar or lidar, instead of vision. In addition to these other sensing modalities, vehicles can often leverage prior maps to simplify online perception. Using a prior map that includes stop signs, speed limits, lanes, etc., a vehicle can largely simplify its onboard perception requirements to the problem of estimating its position with respect to the map (localization), and dealing with dynamic obstacles, such as other vehicles. In this paper, we used a localization system that provides robust onboard localization accuracy of $<$15 cm, and focus on the problem of perceiving traffic lights.

Traffic lights are a special perception problem. Efforts have been made to broadcast traffic light state over radio [Audi MediaInfo, 2008], [Huang and Miller, 2003], but this requires a significant investment in infrastructure. A prior map can be used to indicate when and where a vehicle should be able to see a traffic light, but vision is the only way to detect the state of the light, which may include detecting which sub-elements of the light are illuminated (Figure 1). Although any vision task may be challenging due to the variety of outdoor conditions, traffic lights have been engineered to be highly visible, emissive light sources that eliminate or greatly reduce illumination-based appearance variations: a camera with fixed gain, exposure, and aperture can be directly calibrated to traffic light color levels.

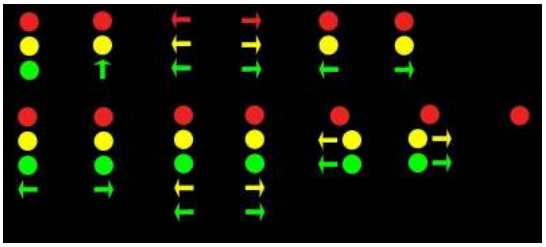

Fig. 1. General traffic light types that are detected by our system.

Safety is of paramount importance in the automotive field. The most common failure conditions in a traffic light detection system are either visual obstructions or false positives such as those induced by the brake lights of other vehicles. Happily, both of these are fail-safe conditions. By using a map of traffic lights the vehicle can predict when it *should* see traffic lights and take conservative action, such as braking gradually to a stop while alerting the driver, when it is unable to observe any lights. False positives from brake lights are also safe, since the vehicle should already be braking for the obstructing object. Rarer false positives, including false greens, may arise from particular patterns of light on a tree, or from brightly lit billboards. However, for the car to take an incorrect action, it must both fail to see all the red lights in an intersection (several false negatives), and falsely detect a green light (a false positive). Using a prior map, the vehicle can also predict a precise window where the light should appear in the camera image. Tight priors on the prediction window, strict classifiers, temporal filtering, and interaction with the driver can help to reduce these false positives. When there are multiple equivalent lights visible at an intersection, the chance of multiple simultaneous failures is geometrically reduced. Thus, detecting traffic lights using a prior map usually fails safely, when it fails.

In this paper we describe methods for building maps of traffic lights, and methods for using these maps to perform online traffic light detection. We present results from large-scale mapping, and analyze the performance of the detector.

II. Related Work

There has been a large amount of research in the general problem of detecting traffic signs (stop signs, speed limits, etc) [Fang et al., 2004], [Fleyeh, 2005], [Prieto and Allen, 2009]. Our work largely avoids the need to detect traffic signs through the use of an annotated prior map.

[Hwang et al., 2006] use a visually-derived estimate of intersection locations to aid the detection of signalling lights (turn signals, brake lights, etc). [Chung et al., 2002] perform traffic light detection using videos of intersections taken from a stationary camera, allowing them to subtract background images. Others, [Kim et al., 2007], [de Charette and Nashashibi, 2009], attempt to detect and classify traffic lights just from onboard images.

In the most closely related work, [Lindner et al., 2004] describe a system that can use the vehicle's motion and a map to aid traffic light detection. Although their detection system is very similar to ours, they do not discuss how to build the traffic light map.

Many of these systems struggle to provide the reliability required to safely pass through intersections.

III. Traffic Light Mapping

Traffic light mapping is the process of estimating the 3D position and orientation, or pose, of traffic lights. Precise data on the pose of traffic lights is not generally available, and while it is possible to make estimates of traffic light locations from aerial or satellite imagery, such maps are typically only registered to within a few meters and do not provide estimates of the traffic light altitude (although it might be possible to use the image capture time and ephemeris data to roughly estimate the altitude from the length of shadows, if they are present).

However, it is easy to drive a mapping car instrumented with cameras, GPS, IMU, lasers, etc., through intersections and collect precisely timestamped camera images. The traffic light positions can then be estimated by triangulating from multiple views. All that is needed is a large set of well-labeled images of the traffic lights. The accuracy of the traffic light map is coupled to the accuracy of the position estimates of the mapping car. In our case online position estimates of the mapping car can be refined by offline optimization methods [Thrun and Montemerlo, 2005] to yield position accuracy below 0.15 m, or with a similar accuracy onboard the car by localizing with a map constructed from the offline optimization.

A. Camera Configuration

We use a Point Grey Grasshopper 5 mega-pixel camera facing straight ahead and mounted to the right of the rear-view mirror, where it minimally obstructs the driver's field of view. Using region-of-interest selection, we use only a 2040×1080 pixel region of the camera. We use a fixed lens with a 30 degree field of view, which we selected with the goal of sufficient resolution to be able to detect traffic lights out to 150 m, a reasonable braking distance when traveling at 55 MPH. Since our detector depends primarily on color because no structure is visible at night, we also fix the gain and shutter speeds to avoid saturation of the traffic lights, particularly bright LED-based green lights.

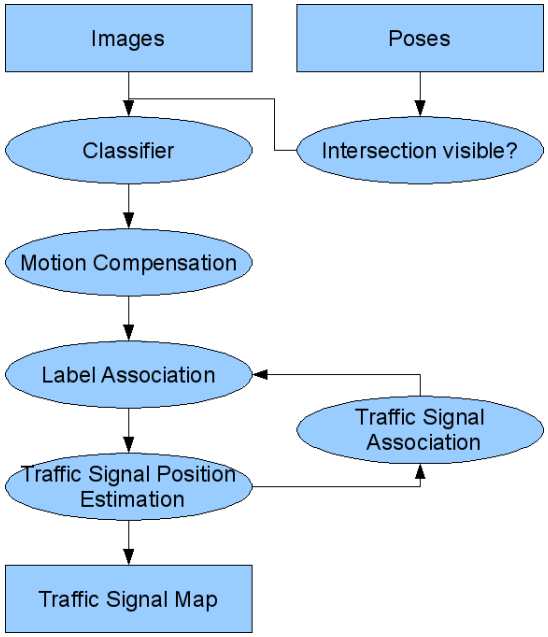

Fig. 2. A diagram of the mapping pipeline as described in the text.

B. Automatic Labeling

One method for generating labeled images is to use a team of human labelers. In our experience, a labeler can label a one-way pass through a typical four-light intersection in about five minutes, not including verification and quality control. This rate is acceptable for small maps, but does not scale well to the high density of lights in a typical urban environment. Furthermore, even the best human labelers are not always accurate in their placement of labels – while they are capable of pixel-level accuracy, the trade-off is a reduction in overall speed and increase in cognitive fatigue. An obvious alternative is to use an automatic labeling system.

The input to the automatic mapping system is a log file that includes camera images and the car's precise pose (Figure 2 shows the mapping pipeline). Generally, traffic lights will only occur at intersections, so we use geo-spatial queries (through the Google Maps API) to discard images taken when no intersections are likely to be visible. Unfortunately, Google Maps only includes intersections and not whether there is a traffic light at the intersection, which would make this winnowing process even more precise.

C. Classification

After winnowing the set of images to those that were taken while the car was approaching an intersection, we run a traffic light classifier over the entire image. The classifier finds brightly-colored red, yellow, and green blobs with appropriate size and aspect ratios, and these blobs are then

used as tentative traffic light labels for the position estimation process.

D. Position Estimation

The output of the labeling step is a large number of labels, but with no information about which labels belong to which traffic lights. To estimate the position of an object in 3D via triangulation, at least two labels in different images are needed, and the position estimate will generally improve if more labels are available. Association of labels to objects can be done between labels in image sequences, or between 3D objects once position estimation has been performed. We use an iterative approach that combines both types of association.

1) Image-to-Image Association: As a first step, the full size of the traffic light is inferred by assuming that the light has the standard vertical red-yellow-green structure, which is the most common configuration in our area. These full-sized traffic light labels make it easier to associate labels that come from traffic lights that have changed color.

There are several possible ways to do label association between images. In the case of near-affine motion and/or high frame rates, template trackers can be used to associate a label in one image with a label in the next image. In our case, the camera frame rate is low (4 fps), and the object motion may be fully projective, so we implemented direct motion compensation.

The precise car pose is known for each image, and the apparent motion of objects due to changes in roll, pitch, and yaw are straightforward to correct using the camera model, but some estimate of the object's position is necessary to correct the apparent motion of objects due to the car's forward motion. The object's apparent position in the image restricts its position to somewhere along a ray, and a rough estimate of the distance of the object can be made by assuming that the object is a traffic light element, and exploiting the fact that most traffic light elements are about 0.3 m in diameter. The distance d to an object with true width w and apparent width $\tilde{w}$ in an image taken by a camera with focal length f_u is

$$d \approx \frac{w}{2\tan(\frac{\tilde{w}}{2f_u})},$$

the direction vector $\mathbf{x} = [u, v]^\top$ can be computed by using the camera model to correct for radial distortion, and the rough 3D position of the object in the camera coordinate frame is

$$\begin{aligned} y &= \sin(\arctan(-u))d, \\ z &= \sin(\arctan(-v))d, \\ x &= \sqrt{d^2 - y^2 - z^2}. \end{aligned}$$

If T_1 and T_2 are the 4×4 transformation matrices for two different times from the car's frame to a locally smooth coordinate frame, we can correct for the relative motion of the object from one image to another as

$$\hat{x}_2 = CT_2T_1^{-1}C^{-1}x_1,$$

where C is the transform from the vehicle frame to the camera coordinate frame, and then compute the distorted image coordinates by using the camera's intrinsic model. Two labels are associated if they fall within an association distance of each other: we use the sum of the object radii. In this way, long sequences of labels are associated, meaning that they are inferred to arise from a single traffic light object in the real world.

If, in fact, the label corresponds to some other type of object, then the rough distance estimate and the subsequent motion compensation will be incorrect. This has the effect of reducing the likelihood of label associations between objects incorrectly classified as traffic lights and helps filter out spurious labels. On the other hand, if the motion-corrected label overlaps another label, then it is likely that they correspond to the same object. The 3D position of the object can then be estimated from a sequence of such corresponding labels.

2) Least Squares Triangulation: Using a set of object labels in two or more camera images, we estimate the pose of the 3D object using linear triangulation and the Direct Linear Transform [Hartley and Zisserman, 2004], a least-squares method. There is an optimal triangulation method described by [Hartley, 1997], but the optimal method can only be applied to up to three labels (or views), while there are often many labels for the same traffic light.

For each provisionally classified image label, we have the image coordinates $\bar{\mathbf{x}}$. These image coordinates are converted into a direction vector $\mathbf{x}$ using the camera's intrinsic model to correct for radial distortion, etc.

We estimate the 3D point $\mathbf{X}$ such that for each label direction vector $\mathbf{x}_i$ and the camera projection matrix P_i,

$$\mathbf{x}_i = P_i\mathbf{X}.$$

These equations can be combined into the form

$$A\mathbf{X} = 0,$$

which is a linear equation in $\mathbf{X}$. In order to eliminate the homogeneous scale factor inherent in projective geometry, we assemble the $3n \times 4$ matrix A from the cross product of each of the image labels $\{\mathbf{x}_1, \mathbf{x}_2, \dots\}$ for a particular object.

$$A = \begin{bmatrix} [\mathbf{x}_1]_\times HP_1 \\ [\mathbf{x}_2]_\times HP_2 \\ \cdots \end{bmatrix},$$

where cross-product matrix

$$[\mathbf{x}]_\times = \begin{bmatrix} 0 & -1 & -u \\ 1 & 0 & -v \\ u & v & 0 \end{bmatrix}, \text{ and } H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

We then perform SVD on A, where $A = U\Sigma V^\top$, and the solution for $\mathbf{X}$ is the de-homogenized singular vector that corresponds to the smallest singular value of A, or the right-most column vector of V. This vector is the least squares estimate of the 3D object position.

The orientation of the traffic light is estimated as the reciprocal heading of the mean car heading over all the image labels that were used to estimate the light position.

3) Camera Extrinsic Calibration: The accuracy of the mapping pipeline is very sensitive to the extrinsic parameters of the camera: the transform that represents the camera's position and orientation relative to the car's coordinate frame. Assuming a reasonable initial estimate of the extrinsic parameters, we precisely calibrate them by minimizing the reprojection error of the traffic lights using coordinate descent:

$$e^* = \arg\min_e \sum_i (\textit{RadialLensDistortion}(PX_{i,e}) - x_i)^2$$

where X_e are the traffic light positions estimated by the mapping pipeline using extrinsic parameters e. A similar process is used to estimate the timing delay between when the image is taken by the camera and when it is transmitted to the computer, although we now also have the capability to use precise hardware timestamps. This timing delay varies depending on the camera frame rate and Firewire bus scheduling allocation, but is stable to within a few hundredths of a second for a given configuration. The camera's intrinsic parameters, which determine the lens distortion, are calibrated with a standard radial lens model.

E. Traffic Light Semantics

Even human drivers can be confused by the semantics of new and complex intersections. In particular, drivers need to know which lights are relevant to their current lane and to their desired trajectory through the intersection. Since we have a detailed prior map, this information can be represented as an association between a traffic light and the different allowed routes through an intersection. We use simple heuristics based on the estimated traffic light orientation and the average intersection width to make an initial guess as to these associations, but then manually verify and these guesses. This is particularly necessary for complex multi-lane intersections.

IV. Traffic Light Detection

Traffic lights are detected onboard the car as follows:

a) Prediction: With the car pose and an accurate prior map of the traffic light locations, we can predict when traffic lights should be visible, and where they should appear in the camera image. Our implementation uses a kd tree to find nearby lights, and a simple visibility model that includes the car orientation as well as the traffic signal orientation. We use the car position estimate from our lidar localization system, which also includes accurate elevation. The predicted positions are then projected using the camera model into the image frame as an axis-aligned bounding box. To account for inaccuracy in the prediction, this bounding box is made three times larger in each axis than the actual prediction.

b) Classification: The color blob-segmentation classifier used in the mapping process above is used to find appropriately-sized brightly colored red and green blobs within each of the predicted bounding boxes. From the prior map, the type of the expected light elements (round or arrow) to impose additional constraints on the blob geometry: for example, the ratio of bounding box to blob pixels is much higher for a round light than for an arrow. This generates a set of one or more classifications for each traffic signal.

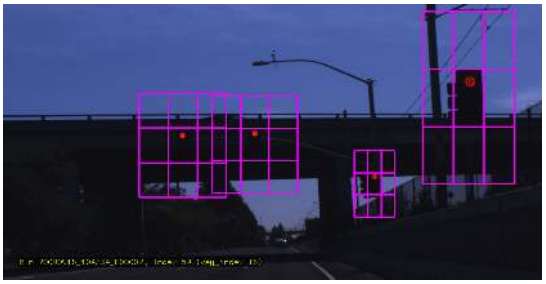

Fig. 3. Negative example neighbors are generated from each human-labeled positive example. The newly generated examples are discarded if they overlap with a positive example, as is the case when two traffic lights are very close together. This figure also shows how the camera settings have been configured to give a dark image, even during the day.

c) Geometric Constraints: From the map, we also know the type of the traffic signal (Figure 1). We can use the structure of the light to apply geometric constraints on the low-level blobs and decide which elements are actually illuminated. For example, for simple 3-stack lights we use a very simple rule that prefers the geometrically highest classification within a prediction window. This deals with the common case of a geometrically low red light detection arising from orange pedestrian cross-walk lights that are frequently just below green lights, well within the predicted bounding box.

d) Temporal Filtering: In the absence of recent measurements of the light state, the light is assumed to be yellow. Temporal filtering smoothes the output of the detector by assuming that if there is no new classification then the light state has not changed. The temporal filtering degrades the confidence in unobserved lights over time, and defaults to the prior assumption that the light is yellow within a second.

The vehicle can decide whether a particular path through an intersection is allowed by applying the rule that out of all the traffic signals associated with the path no red or yellow lights may be detected and at least one green light must be detected.

V. Classifier Optimization

We have assembled a repository of thousands of human-labeled traffic lights, and we continue to add more labels as we encounter new situations, such as brightly colored billboards or inclement weather. We also generate negative examples using regions around labeled traffic lights (Figure 3). This repository is used to optimize the classifier and to perform regression tests.

Since the repository may grow indefinitely, we have implemented optimization methods that scale well to large-scale computing resources. We omit the details of the large-scale computing architecture, except to mention that when using large numbers of machines, the optimization architecture must be robust to temporary faults or complete failure of

any particular machine, not excepting the central control machine.

Since each image is 2040×1080 pixels, loading and decompressing the sequence of images from disk is more time-intensive than evaluating the classifier itself. Instead, we distribute the images across a set of machines such that each machine loads a small set of images into RAM and then repeatedly evaluates the classifier according to the current optimizer state on those cached images. A particular classifier configuration can be evaluated over a set of over 10000 images (with several lights per image) by a few hundred computers in under a second. This allows us to use iterative hill-climbing approaches, such as coordinate ascent, to optimize the classifier in under an hour. Further parallelization is possible by batching up all the states to be evaluated in a single optimizer step. Since we have many more negative examples than positive examples, we weight the positive examples to prevent convergence to the local minima of no detections.

VI. Results

The true metric for success for the traffic light mapping and detection system is whether it provides reliable and timely information about the state of the intersection. Intersections are usually engineered with redundant traffic lights, and it is almost always safe to simply slow down and continue looking, so the actual performance of the system is difficult to characterize with static plots such as Figure 5. For example, when the traffic lights are obscured, they are assumed to be yellow and the system signals to reduce speed. As the car approaches the intersection, the probability of detection of at least one of the usual set of three or four traffic lights increases, and the system outputs the correct state.

The mapping pipeline automatically generates large-scale maps (Figure 4) of traffic light positions and orientations with low reprojection error, as verified by multiple passes with different cars (with different camera calibrations). Depending on the area and traffic conditions, the mapping pipeline misses between 1 and 5 % of the traffic lights. Although there is a significant amount of human effort involved in verifying and tweaking the traffic light positions, shapes, and lane associations, the automatic pipeline does the bulk of the most tedious work: placing the traffic lights in the world.

Figure 5 shows the intersection state (go/stop) detection rate for a drive down El Camino Real and various side routes. The drive as 52 miles long, with about 220 intersections controlled by traffic lights. We controlled for obstructions, curves, etc. by only counting intersections that a human could classify from the camera images. Note that while there are almost always multiple semantically identical traffic lights in an intersection, it is only necessary for the system to see one of these lights to determine the intersection state. The sharp knee in the detection rate above 190 m is likely due to sampling effects: at ∼15 m/s, the vehicle travels several meters in between the 4 Hz camera frames. Since the system only starts to predict traffic lights at 200 m, this means that

Fig. 4. This map shows some of the traffic lights we have mapped in the San Francisco - San Jose area, in total we have mapped about a thousand intersections and over 4000 lights (though not necessarily all the lights in a given intersection).

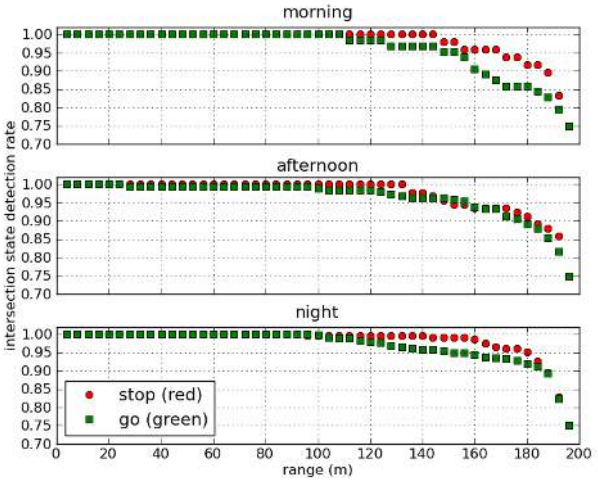

Fig. 5. Plot of detection range of the state of the intersection (red = stop, green = go) for a 52 mile drive on El Camino Real at three different times of day. The traffic light detector begins to predict that it should see traffic lights at 200 m. The illumination at different times of day most heavily impacts the detection of green lights, with the best performance at night.

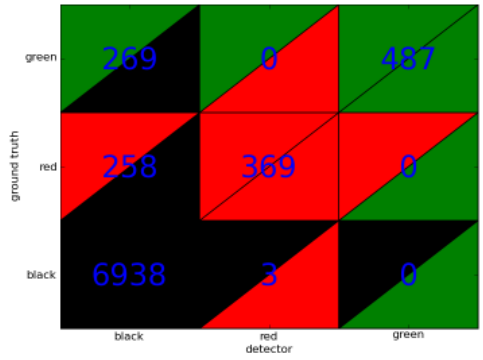

Fig. 6. Traffic light confusion matrix for a typical drive. The ground truth includes negative examples generated from the eight neighbors of each positive example, except when these negative examples overlap positive examples. Yellow light examples were not included, since the detector assumes all lights are yellow by default.

missing just a frame or two drops the detection range to around 190 m.

The confusion matrix for the training dataset is shown in Figure 6. These scores include detections at all ranges from 0 - 200 m. We can map the confusion matrix into a simple table containing true positives (tp), true negatives (tn), false positives (fp), and false negatives (fn):

	Detector Positive	Detector Negative
Positive	856 (tp)	527 (fn)
Negative	3 (fp)	6938 (tn)

Using this mapping, the detector's precision was $856/859 = 0.99$, while recall was $856/1378 = 0.62$. Part of the reason that the recall is low is that we insist on no false-positive green lights.

The traffic light detection system has $\sim$0.2 s latency, primarily due to the latency for the transmission of the image from the camera to the computer, which is $\sim$ 0.12 s. Similarly, camera bandwidth limitations determined the frame rate of the detection pipeline, which was 4 Hz. The processor load is less than 25% of a single CPU, primarily due to the high resolution of the images rather than any complexity in the algorithm.

While the system works well at night and with moderate rain (the camera is mounted behind an area swept by the windshield wipers), glare from the sun or heavy rain obscure the camera's view and degrade performance. There are also especially dim traffic lights, particularly green lights with fish-eye lenses, that are difficult for the system to detect. Depending on the specific case, we may annotate the map with a "dim" tag, in which case our system assumes that the light is green (as opposed to the standard case, in which our system assumes that the light is yellow).

VII. Conclusions

Cars must deal with traffic lights. The two main tasks are detecting the traffic lights and understanding their control semantics. Our approach to solving these two tasks has been to automatically construct maps of the traffic light positions and orientations, and then to manually add control semantics to each light. We then use this map to allow an onboard perception system to anticipate when it should see and react to a traffic light, to improve the performance of the traffic light detector by predicting the precise location of the traffic light in the camera image, and to then determine whether a particular route through an intersection is allowed. Our system has been deployed on multiple cars, and has provided reliable and timely information about the state of the traffic lights during thousands of drives through intersections.

We are now experimenting with a secondary camera with a wider field of view but a shorter detection range, that allows the car to detect nearby lights when it is very close to the intersection. We are also experimenting with the robust detection of flashing lights: we can annotate the map with lights that always flash or lights that flash at certain times of day, but during construction or an emergency lights may unpredictably be switched to flash. Finally, there are cases where the lights, frequently arrows, are so dim, it appears that the only way to detect the state is to watch for relative changes in intensity of the light elements.

Thanks to Anna Osepayshvili for her analysis of the human QC time required to build the traffic light maps.

References

[Audi MediaInfo, 2008] Audi MediaInfo (2008). Travolution promotes eco-friendly driving. Available at `http://www.audiusanews.com/newsrelease.do?id=1016&mid=76`.

[Chung et al., 2002] Chung, Y.-C., Wang, J.-M., and Chen, S.-W. (2002). A vision-based traffic light detection system at intersections. *J. Taiwan Normal University: Mathematics, Science & Technology*, 47(1):67–86.

[de Charette and Nashashibi, 2009] de Charette, R. and Nashashibi, F. (2009). Traffic light recognition using image processing compared to learning processes. In *Proc. IROS 2009*, pages 333–338.

[Fang et al., 2004] Fang, C. Y., Fuh, C. S., Yen, P. S., Cherng, S., and Chen, S. W. (2004). An automatic road sign recognition system based on a computational model of human recognition processing. *Comput. Vis. Image Underst.*, 96(2):237–268.

[Fleyeh, 2005] Fleyeh, H. (2005). Road and traffic sign color detection and segmentation – a fuzzy approach. In *IAPR Conference on Machine Vision Applications*.

[Hartley, 1997] Hartley, R. (1997). Lines and points in three views and the trifocal tensor. *IJCV*, 22(2):125–140.

[Hartley and Zisserman, 2004] Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition.

[Huang and Miller, 2003] Huang, Q. and Miller, R. (2003). The design of reliable protocols for wireless traffic signal system. Technical report, Communications Review.

[Hwang et al., 2006] Hwang, T., Joo, I., and Cho, S. (2006). Detection of traffic lights for vision-based car navigation system. In *PSIVT06*, pages 682–691.

[Kim et al., 2007] Kim, Y., Kim, K., and Yang, X. (2007). Real time traffic light recognition system for color vision deficiencies. In *ICMA 2007.*, pages 76 –81.

[Lindner et al., 2004] Lindner, F., Kressel, U., and Kaelberer, S. (2004). Robust recognition of traffic signals. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 49–53.

[Plunkett Research, Ltd., 2009] Plunkett Research, Ltd. (2009). Automobiles and trucks overview. Available at `http://www.plunkettresearch.com/Industries/AutomobilesTrucks/AutomobileTrends/tabid/89/Default.aspx`.

[Prieto and Allen, 2009] Prieto, M. and Allen, A. (2009). Using self-organising maps in the detection and recognition of road signs. *Image Vision Comput.*, 27(6):673–683.

[Thrun and Montemerlo, 2005] Thrun, S. and Montemerlo, M. (2005). The GraphSLAM algorithm with applications to large-scale mapping of urban structures. *IJRR*, 25(5/6):403–430.