Transitioning Control and Sensing Technologies from Fully-autonomous Driving to Driver Assistance Systems

Humberto Gonzalez[♣] Esten I. Grøtli[♢] Todd R. Templeton[♣]
Jan O. Biermeyer[♣] Jonathan Sprinkle[♠] S. Shankar Sastry[♣]

[♣]Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720, U.S.A.
phone: +1 (510) 643-9783, fax: +1 (510) 643-2356, email: {hgonzale,ttemplet,janb,sastry}@eecs.berkeley.edu.
[♢]Department of Engineering Cybernetics, Norwegian University of Science and Technology, N-7491 Trondheim, Norway. grotli@itk.ntnu.no
[♠]Department of Electrical and Computer Engineering, University of Arizona, Tuscon, AZ 85721, U.S.A. sprinkle@ece.arizona.edu

Abstract

Based on our experience in the DARPA Urban Challenge and on current trends in consumer automobiles, we believe that driver assistance systems can be significantly improved by new techniques in control and sensing that have been developed for fully-autonomous driving.

In particular, from the control community, real-time Model Predictive Control (MPC) can be used as the next generation of cruise control for automobiles, offering a principled method for robustly incorporating information from automobiles' existing sensing systems, such as GPS and odometry, as well as from additional sensors that will be used in future, complimentary driver assistance systems, such as visible-light cameras, infrared (IR) cameras and laser scanners.

From the sensing community, we believe that obstacle-detection systems using passive (hence, noninterfering) and cost-effective visible-light cameras and thermal IR cameras, originally developed for fully-autonomous driving, are also a valuable addition to the driver assistance toolbox, offering the ability to warn drivers about moving or heat-producing obstacles, including pedestrians and other automobiles.

In this paper we will discuss methods, derived from fully-autonomous vehicle research, for real-time Model Predictive Control (MPC), segmentation of moving (relative to the ground) obstacles using visible-light cameras, and detection of heat-producing objects using thermal infrared (IR) cameras, as well as their application to driver assistance systems.

1 Introduction

Driver assistance systems range from the well-known — in-vehicle navigation and guidance systems with aid from GPS, cruise control and anti-lock braking systems (ABS) — to the less-well-known [1]:

- *vision enhancement*: suitable for reduced-sight conditions due to inadequate lighting, night driving, fog, rain or snow

- *object detection*: warn the driver about surrounding vehicles during driving and parking
- *adaptive cruise control*: maintain a driver-set speed or, in the case of a following vehicle, maintain a certain distance to the vehicle in front
- *intelligent speed control*: limit the speed of the vehicle via a signal from the infrastructure
- *roll stability control*: throttle or brakes are activated in a potential roll-over situation
- *lane-keeping assistance*: the driver is warned or a correcting steering action is taken when an imminent lane departure (without the use of a turn signal) is detected

These technologies are not just research topics. Conventional cruise control has been available for about two decades, and premium vehicles such as the Mercedes-Benz S-Class, Jaguar XKR and Lexus LS430 have had *adaptive cruise control* as an option for several years [2, 3, 4]. Recently, Nissan has announced that the *intelligent cruise control* in their new 2008 Fuga, released in Japan, combines route information and a navigation system to decelerate when approaching a curve, and then accelerate to the original speed when the road straightens [5].

Recently, the third DARPA Grand Challenge, named the 'Urban Challenge' because it took place in a cluttered urban environment, demonstrated that driverless cars can detect obstacles, negotiate with other traffic and make sophisticated interactions with each other [6, 7]. Although such fully-autonomous vehicles are far from perfect, developments such as the successful completion of the Urban Challenge course by five autonomous vehicles beg the following question: By equipping the vehicle with appropriate sensors (such as video cameras, radars, laser scanners or ultrasound devices), can we use technologies developed for autonomous vehicles to increase safety, efficiency and convenience in driver assistance systems?

In this paper, we develop important pieces of a next-generation cruise control scheme based on technologies originally designed for autonomous vehicles. In particular, we develop a real-time driver assistance vehicle controller using Model Predictive Control (MPC), as well as real-time methods for providing sensor information about vehicles and other obstacles to such a system using passive (hence, noninterfering) and cost-effective visible-light and thermal-infrared cameras.

2 Definitions: Model Predictive Control

Model Predictive Control (MPC) is a particular case of optimal control where, instead of solving the problem for all time up to infinity, we only solve it for a finite window of time in the future, using the current state as the initial state of the model. After a fixed time we solve a similar problem again, using the new state as the initial state.

Let $x(t_0)$ be the current state and $u(k)$ be the input of the dynamical system f at time $t = kT$, with sampling time T. Then, given the control input sequence $\{u(k)\}_{k=0}^{N}$, the future

states $\hat{x}(\cdot)$ up to time $t = (N+1)T$ can be predicted as:

$$\begin{aligned} \hat{x}(k+1) &= f(\hat{x}(k), u(k)) \quad k = 0, \ldots, N \\ \hat{x}(0) &= x(t_0) \end{aligned} \tag{1}$$

Now, given a lower-bounded cost function $J(\hat{x}, u)$, we want to solve

$$\tilde{u}_{t_0} = \arg\min_{\{u_k\}} \; J(\hat{x}, u) \tag{2}$$

subject to (1). This same approach can be extended to values of $\hat{x}$ and u in properly-defined sets (such as constraint-defined sets).

Let $t_r = k_r T$ such that $k_r \leq (N+1)$, then we will apply the control

$$u(t) = \tilde{u}_{t_0}(k) \quad t \in [t_0 + kT, t_0 + (k+1)T) \tag{3}$$

for $k = 0, \ldots, k_r - 1$. At time t_r we solve problem (2) again, updating the values of t_0 and $x(t_0)$. This MPC formulation allows us to develop a multi-threaded implementation of the algorithm, where one thread solves the optimization problem and another thread applies the proper input in the window of time $[0, k_r T]$. Thus, the parameter k_r is directly related to the time required for the optimization thread to finish.

We choose MPC as our control strategy because of the flexibility that it allows in the definition of the control objective. Whereas most control strategies use objectives such us "regulation" or "tracking" of a signal, we can define objectives such as "obstacle avoidance" or "better performance" by means of an optimization problem with constraints. Although solving an optimization problem in real-time is difficult in general, we will exploit some properties of this particular problem to achieve our real-time goal.

Real-time implementations of MPC have been developed in [8, 9, 10] (among many others), often avoiding the explicit use of constraints in the optimization of the control input because, as we mentioned above, it is difficult to numerically solve constrained optimization problems in real time. Since the problem of driving in an unknown environment is primarily one of avoiding obstacles, we must add constraints to our formulation; thus, our work is focused on solving the constrained optimization problem in real time.

3 Assisted Highway Driving Scheme

Until now, cruise control has been considered to be a speed regulation problem, where we are only interested in automating the behavior of the accelerator pedal given a particular speed reference. With state-of-the-art sensors and computing technologies, this paradigm can be moved one layer up, including information from the highway and from the local neighborhood of the car to control not only the accelerator, but also the steering and the brake pedal in order to follow

the lane and to provide a simple collision-avoidance scheme.

3.1 Model of the Car

We will use the following model for the position and course of the car:

$$\begin{aligned}\dot{x}_1(t) &= x_3(t) \cdot \cos(x_4(t)) \\ \dot{x}_2(t) &= x_3(t) \cdot \sin(x_4(t)) \\ \dot{x}_3(t) &= u_1(t) \\ \dot{x}_4(t) &= u_2(t)\end{aligned} \tag{4}$$

where x_1 is the x-coordinate, x_2 is the y-coordinate, x_3 is the speed, x_4 is the course, u_1 is the acceleration and u_2 is the rate of change of the steering wheel angle. We will also use x_P to describe the position of the car, i.e. $x_P = [x_1\ x_2]^T \in \mathbb{R}^2$. The reference frame chosen for the coordinates is north-east.

We will use the forward Euler discretization algorithm to obtain a discrete-time model, like the one described in (1), from (4).

3.2 Obstacles

Let us describe the local neighborhood as a set of M obstacles $\{O_j\}_{j=1}^{M}$, where $O_j = (H_j, v_j, \theta_j)$, H_j is a set of Cartesian coordinates such that the obstacle is a subset of the convex hull of H_j (denoted co $\{H_j\}$), $v_j \in \mathbb{R}_+$ is the speed of the obstacle with respect to the ground and θ_j is the course of the obstacle with respect to north. Also, let $\widehat{H}_j(k)$ be the set of predicted positions of the obstacle's points at time $t = kT$ assuming that it follows the same speed and course. Thus, $\forall x_k \in \widehat{H}_j(k), \exists x_0 \in H_j$ such that

$$x_k = kTv_j\hat{d}_j + x_0 \tag{5}$$

where $\hat{d}_j = [\cos\theta_j\ \sin\theta_j]^T$ is the course vector of the obstacle.

Let $d(x_P, H)$ be the signed distance from x_P to ∂co $\{H\}$, where ∂co $\{H\}$ is the boundary of the convex hull of $H \subset \mathbb{R}^2$, e.g. let

$$d(x_P, H) = \begin{cases} +\inf\{\|x_P - y\|_2 : y \in \partial\text{co}\{H\}\} & \text{if } x_P \in \text{co}\{H\} \\ -\inf\{\|x_P - y\|_2 : y \in \partial\text{co}\{H\}\} & \text{if } x_P \notin \text{co}\{H\} \end{cases} \tag{6}$$

In this case, we now want to solve the following improved version of (2):

$$\begin{aligned} u_{opt} = \arg \min_{\{u_k\} \in \mathcal{U}} \; & J(\hat{x}, u) \\ \text{s.t. } & d(\hat{x}_P(k), \widehat{H}_j(k)) < -\varepsilon \quad \forall k, j \\ & v_{\min} \leq \hat{x}_3(k) \leq v_{\max} \quad \forall k \end{aligned} \tag{7}$$

subject to (1), where $\mathcal{U}$ is a box constraint set and $\varepsilon > 0$ is a safety parameter at least equal to the length of the car.

3.3 Objective Function

Our choice of objective function must include two important elements:

1. maintain the reference speed under the obstacle constraints
2. follow the current lane, even under curves

Thus, we consider the following cost function:

$$J(x, u) = \sum_{k=0}^{N} \left[K_1 J^k_{\text{speed}}(x) + K_2 J^k_{\text{input}}(u) + K_3 J^k_{\text{lane}}(x) \right] + J^{N+1}_{\text{speed}}(x) + J^{N+1}_{\text{lane}}(x) \tag{8}$$

where

$$J^k_{\text{speed}}(x) = \left(x_3(k) - v_{\text{ref}}\right)^2 \tag{9}$$

$$J^k_{\text{input}}(u) = u(k)^T R\, u(k) \tag{10}$$

$$J^k_{\text{lane}}(x) = \left[\left(\hat{d}^T_{\text{lane}} x_P(k) \right) \hat{d}_{\text{lane}} - x_P(k) \right]^2 \tag{11}$$

Here, $K_i \in \mathbb{R}_+ \; i = 1, 2, 3$, v_{ref} is the speed reference, R is a weight matrix, $\hat{d}_{\text{lane}} = [\cos\beta \;\; \sin\beta]^T \in \mathbb{R}^2$ and β is the angle of the lane with respect to north $(N+1)T$ seconds ahead (at the current speed). The parameters $\{K_i\}_{i=1}^3$ allow us to tune the dynamical behavior of the closed loop in order to achieve both objectives simultaneously.

3.4 Solver

The problem of driving a vehicle in a dynamic environment is one of finding feasible inputs for the car such that it will not collide with any obstacles. If a feasible input set exists, then we should choose the input from that set that minimizes our cost function. With this idea in mind, we propose the following method to solve the optimal control problem.

Numerical optimization under constraints is generally a computationally-expensive task, but if we can simplify the problem to an unconstrained one then there is a greater chance of making it work in real time.

The main idea in this case is to replace the obstacle constraints by (cost) penalty functions of the form:

$$J^k_{\text{obs}}(x) = \sum_{j=1}^{M} h_j(k) \tag{12}$$

where

$$h_j(k) = \begin{cases} \frac{1}{|d(x_P(k),\widehat{H}_j(k))+\varepsilon|} & d(x_P(k),\widehat{H}_j(k)) < -\varepsilon \\ +\infty & \text{otherwise} \end{cases} \tag{13}$$

The speed and box constraints can be formulated as linear constraints in the discrete-time case, so it is not necessary to formulate them as penalty functions because they are not as computationally expensive as the nonlinear ones.

Since the penalty function goes up to infinity when we approach an obstacle, we can ensure that the solution will be feasible as long as the initial condition is feasible. Under assumptions such as proper behavior of other vehicles in the local neighborhood, e.g. all vehicles follow the driving rules of the highway, then finding a feasible initial input $\{u_k\}$ is not a difficult task, since when we start the system we can begin without obstacles and after that we can use the previous optimal solution as an initial condition. If the previous solution is not feasible under the new set of obstacles then we can use heuristics to modify that input to make it feasible, such as choosing a slower speed in order to ensure that we will not collide with an obstacle.

Our real-time requirements make a first-order algorithm most suitable for this kind of problem. In this case, we are not looking for an optimal solution, rather we want to compute a "more optimal" solution than our first guess; therefore, we can stop the numerical optimization loop at any time and use that solution as the input for the next time window. In particular, a Projected Gradient method with Armijo step size satisfies all of the requirements, namely a simple first-order algorithm with linear and box constraints [11].

4 Moving Obstacle Detection

The goal of the algorithm described in this section is to detect, and estimate the trajectories of, mobile obstacles such as other vehicles, in the interest of providing these obstacles to the next-generation cruise control described in the previous section. Note that, in this scenario, we wish to detect both objects in close proximity to our vehicle (for example, the car in front of us), and also objects that are farther away (for example, oncoming vehicles), so we cannot make assumptions about whether the objects in the scene appear to be close to planar. Our solution must also be able to run in real time.

4.1 Comparison to Previous Work

Much effort has been expended in solving the two-view single-camera moving object detection problem but, in the case of a real-time, high frame-rate image sequence, applying such an algorithm to each pair of consecutive images would not take into account the fact that points on the same object undergoing rigid motion remain part of the same object over time — the algorithm would have to rediscover this set every time. Moreover, additional computation would be required to determine correspondence between objects in successive frames. Applying a monocular two-view algorithm to each pair of consecutive images would also not take advantage of the range estimates available with a stereo pair; although range estimates from stereo can be inaccurate in many cases, we are in effect using these estimates as one of several measurements for classification (so absolute errors, and small relative errors, are less important than in many applications), and the use of a stereo pair allows us to remove the planarity (or non-planarity) assumption that plagues existing algorithms such as [12, 13, 14, 15].

In this application, we are interested in determining 3D motion in the scene, rather than the more-studied case of 2D motion in the image. In particular, we want to find the motion of objects in the scene relative to each other; if we have vehicle state data available, or if we can assume that the majority of the scene is not moving, we can determine which motion-segmented region corresponds to the background and remove its perceived motion from the perceived motion of the other objects, obtaining the motion of the other objects in a global frame. As discussed in [15], prior work in this area includes many methods that concentrate on more specific problems such as multiple points moving linearly with constant speed [16, 17] or in a conic section [18], multiple points moving on a plane [19], multiple translating planes [20], multiple objects viewed through an affine camera [21, 22, 23, 24, 25, 26, 27], and segmentation of two objects using two perspective views [28]. The case of multiple moving objects seen from two perspective views has been studied in [12, 13, 14, 15], but all of these are either specialized to the case of estimating fundamental matrices (assuming non-planar objects), or are also capable of estimating homographies (assuming planar objects) but cannot choose between the two. Torr in [29] considers the case of both planar and non-planar objects, but his algorithm fits both a fundamental matrix and a homography and includes the complexity of the model in its scoring system, instead of using an underlying method that is invariant to planarity, and his algorithm is not suitable for real-time use. The approach closest to our algorithm is that of Feng and Perona in [30], in that they first cluster the features and then fit a motion model to each cluster, but they only consider essential matrices and hence their algorithm is not suitable for planar or near-planar obstacles. To our knowledge no previous work has taken advantage of a stereo rig and multiple frames available at a high frame rate.

4.2 Proposed Algorithm

Instead of using a feature-based approach, in effect attempting to differentiate independently-moving objects based on only a few (often incorrectly tracked, when using a real-time feature tracker) features on each object, we propose a dense algorithm that takes advantage of all of the information available in the image sequence.

Although we do not expect an entire object to appear to move identically in the image sequence because of differences in range, we do expect neighboring pixels on the same object to move much more similarly than those on the border between independently-moving objects. This immediately suggests a graph segmentation approach (as opposed to traditional image segmentation, which attempts to find homogeneous regions under some metric), to which end we construct an undirected graph with edges between neighboring pixels that have weights given by the similarity between the pixels' (image) movements over a period of time.

As in Geyer et al.'s Recursive Multi-Frame Planar Parallax (RMFPP) algorithm [31], we use a reference frame and a sequence of approximately 10 subsequent frames to estimate a value (here, independently-moving-object membership) for each pixel in the reference frame. The difference between neighboring pixels is defined as a linear combination, summed over all N frames in the sequence:

$$\delta_{p_1,p_2} = \sum_{i \in 1 \ldots N} ||f_{i,p_1} - f_{i,p_2}||_1 \tag{14}$$

$$\text{where} f_{i,p_j} = \begin{pmatrix} u_{i,p_j} \\ v_{i,p_j} \\ z_{i,p_j} \\ \dot{z}_{i,p_j} \\ h_{i,p_j} \end{pmatrix}.$$

Here, u_{i,p_j} and v_{i,p_j} are the optical flow components, z_{i,p_j} is the range, $\dot{z}_{i,p_j}$ is the range velocity (z_{i,p_j} and $\dot{z}_{i,p_j}$ are computed using stereo vision), and h_{i,p_j} is the appearance (hue) in image i of pixel p_j in the reference image.

To construct the similarity graph, we define the similarity between the motion of two adjacent pixels in the reference image as

$$s_{p_1,p_2} = \left(\max_{j \in 1 \ldots P} \max_{k \in n(p_j)} \delta_{p_j,p_k} \right) - \delta_{p_1,p_2}, \tag{15}$$

where P is the number of pixels in each image and $n(p_j)$ is the set of pixels adjacent to pixel p_j in the reference image, i.e. the difference between the maximum over all differences in motion between adjacent pixels, and the difference in motion between the neighboring pixels of interest.

Once we have built the adjacent-pixel motion similarity graph, we use Flake et al.'s graph segmentation algorithm [32] to determine the independently-moving objects. This method

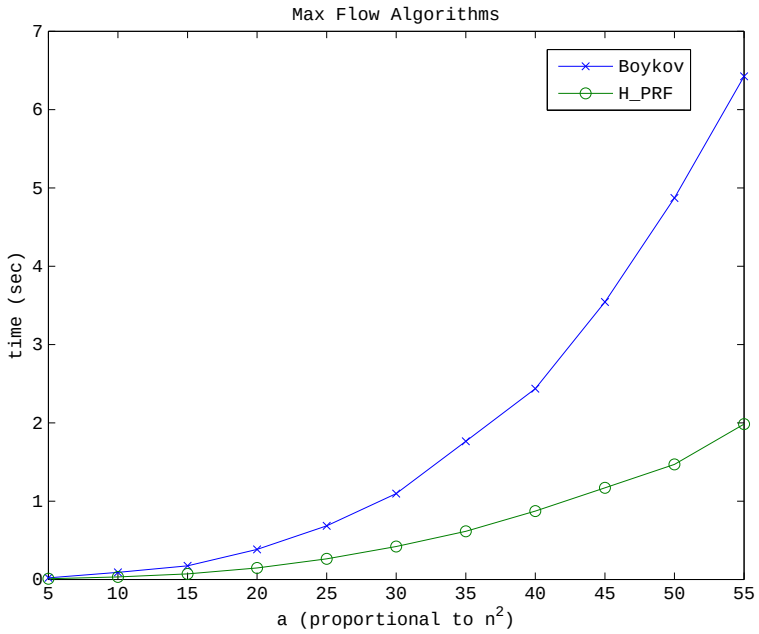
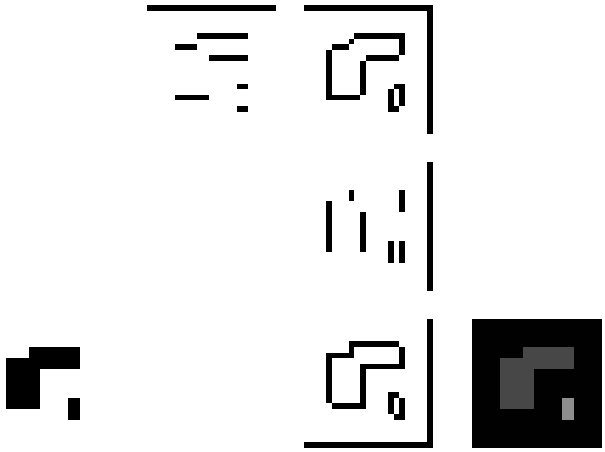


Figure 1: Runtime comparison (average of five trials) of two max-flow/min-cut implementations on general graphs with number of nodes proportional to a^2. (Green 'o') H_PRF, an algorithm that is considered to be one the best for general graphs, (Blue 'x') Boykov and Kolmogorov's algorithm.

makes no assumption about the number of independently-moving object groups, and is controlled by a single parameter α, which is an upper bound on the inter-group edge capacity (motion similarity) and a lower bound on the intra-group edge capacity (motion similarity). We use the heuristic version of this method, which provides the same guarantee but that in practice reduces the number of required max-flow/min-cut operations to the number of groups (observed independently-moving objects).

For the max-flow/min-cut subroutine of the graph segmentation, we use Boykov and Kolmogorov's algorithm [33]. We choose this algorithm because it does not restart the search for an augmenting path from the source and sink at each iteration, which removes the theoretical runtime guarantee of many max-flow/min-cut algorithms, but which is shown in [33] to make it run an order of magnitude faster than the best theoretically-guaranteed algorithms for many practical computer vision problems. In addition, as is illustrated in Figure 1, our tests have shown that this algorithm does not scale significantly differently from the algorithm that is considered to be one of the best for general graphs [33], on general graphs generated by Goldfarb and Grigoriadis' GENRMF [34] maximum-flow random-graph generator.

Finally, we estimate the linear and angular velocities of each independently-moving object. Following Longuet-Higgins in [35], we envisage the vehicle as being in motion relative to the object in the scene (we will imagine each object in the scene to be stationary and find the motion of the view relative to that object; we will find multiple relative motions between the vehicle and



(a) (b) (c)

Figure 2: Graph segmentation result: (a) original image, (b) directional similarities, (c) result of graph segmentation using directional similarities.

the scene if the scene is not all moving together). Consider a world point $\boldsymbol{P} = \begin{bmatrix} X & Y & Z \end{bmatrix}^T$, and let the view move with angular velocity $\boldsymbol{W} = \begin{bmatrix} W_x & W_y & W_z \end{bmatrix}^T$ about each axis, and linear velocity $\boldsymbol{V} = \begin{bmatrix} V_x & V_y & V_z \end{bmatrix}^T$ along each axis. In this case we have that

$$\begin{bmatrix} xy & -(1+x^2) & y & \frac{-1}{z} & 0 & \frac{x}{z} & -u \\ 1+y^2 & -xy & -x & 0 & \frac{-1}{z} & \frac{y}{z} & -v \end{bmatrix} \begin{bmatrix} W_x \\ W_y \\ W_z \\ V_x \\ V_y \\ V_z \\ 1 \end{bmatrix} = \mathbf{0}. \tag{16}$$

Note that we always know the image coordinates (x, y), and that we have approximated (u, v) and z using optical flow and stereo reconstruction, respectively, so this system is linear in $(\boldsymbol{W}, \boldsymbol{V})$. Determining the linear and angular velocities of a given independently-moving object simply requires solving this linear equation with a "stacked" version of the left-hand matrix that includes many frame-to-frame motions of pixels associated with that object.

4.3 Preliminary Result

Although work on this algorithm is in the early stages, we give a preliminary result: We take a simple image, compute the similarities between all neighboring pixels (for each pixel, the similarity betwen it and its top, top right, right, and bottom right neighbors), and then use only these similarities to segment the image (with no infomation about the number of segments).

Figure 2 shows the steps, and the successful result.

5 Heat-producing Obstacle Detection

The algorithm described in this section complements the one described in the previous section in that its goal is to detect, and estimate the trajectories of, *heat-producing* obstacles such as other vehicles, in the interest of providing these obstacles to the next-generation cruise control described above.

While visible light computer vision (VLCV) can aid the driver-assistance system in good light conditions, it naturally fails in bad-light or low-light situations; thermal imaging (FIR) can substantially extend the range of VLCV *and human vision* in these situations. Nighttime driver assistance is not a problem to overlook: the National Highway Traffic Safety Administration (NHTSA) found in data from 2000 that, although nighttime driving represents only 28% of total driving time, 55% of all traffic fatalities occur when it is dark [36]. Admittedly, the same study showed that the rate of alcohol involvement in fatal crashes is more than three times as high at night as it is during the day, but it is clear that reduced vision also accounts for heightened fatalities, especially in collisions involving pedestrians and deer.

We believe that FIR cameras are also essential to a successful driver assistance system during the day, since the heat signatures of all active cars, largely due to hot tail pipes and brakes, stand out in FIR imaging. Although an FIR system cannot detect all obstacles that a visible-light system can detect during the day, it can detect hot obstacles at a longer range and with higher accuracy.

Luxury cars from the 2000 Cadillac DeVille [37] to the BMWs equipped with 'Night Vision' [38] technology feature FIR cameras and in-car monitors or projection methods, mainly marketed for night-time driving in rural areas; here, we would like to add some intelligence to the use of such sensors.

5.1 Algorithm

For each grayscale white-heat FIR image, we first downsample to 320x240 pixels using linear interpolation and then remove the artificial crosshair that is added by our FIR camera (a bitwise AND of the crosshair region with its negation). We then threshold the image at a particular heat level, followed by segmentation via compressing horizontal, vertical, and diagonal segments. Now that we have isolated a set of heat-producing blobs, we compute the image area, average brightness, and first-order moments in both the x and y image directions of each blob. Some blobs, such as very small and such as the elongate sunlight reflections on light posts, are rejected immediately, and all other candidates are numbered for further evaluation.

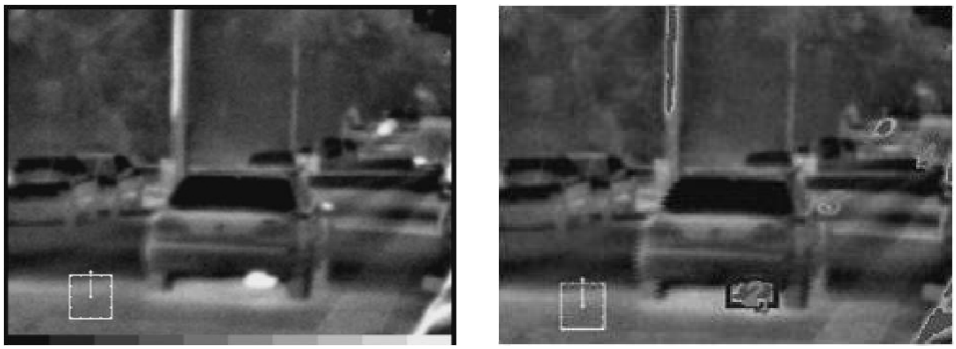
While, at first glance, a classification of blobs based on image area and average brightness seems most promising, this approach turns out to be useful primarily for rectangular heat sig-

natures, such as the ones observed under the sides of oncoming cars. We have found that, for round heat signatures, such as those from tires and especially those from exhaust pipes, the strongest indication that a blob is part of an active-car obstacle is that its first-order x and y moments have different signs.

Although it at first seems that we can increase the accuracy of this scheme over multiple frames by modeling spatial or temporal attributes of blobs using Hidden Markov Models (HMMs) similarly to how they are used in face recognition in [39] or by using a feature tracking algorithm such as the Kanade-Lucas-Tomasi (KLT) tracker [40, 41], we have found that this is generally not feasible. The heat signatures of cars are often unstable; for example, the size of the tail-pipe heat signature changes dramatically with acceleration. Also, most cars have only one visible heat signature from the front or the back, and not only are these heat signatures unstable, but they are often ambiguous; the heat blobs from two tires observed in the side view of one car look very similar to two exhaust pipes on one car, or to two cars next to each other from the back.

5.2 Results

We evaluate the algorithm using a dataset of around 30,000 images, captured from a thermal IR camera mounted on top of a running car parked on the sides of several city streets during rush hour.



(a) sample input FIR image (b) resulting output image

Figure 3: Results of the proposed algorithm. Note that the parked cars on the side of the street do not show any heat signatures.

Figure 3 shows a sample input and output. The heat-producing obstacle detection algorithm correctly identifies each active car in the input images with a success rate of about *75%* manually compared against hand-labeled ground truth. With this dataset, it avergages *52* frames per second when reading images from the hard disk of an off-the-shelf notebook computer with a dual-core 2 GHz processor.

5.3 Future Work

Since the heat-producing obstacle detection algorithm is capable of running at more than twice the framerate of the FIR camera, we are currently experimenting with using the above algorithm as a first stage to isolate regions of interest (ROI), and then using more-sophisticated machine learning methods in a second stage to eliminate most false positives and, ideally, all false negatives. We also plan to extend the approach to non-vehicle obstacles such as pedestrians and deer.

6 Conclusion

In this paper, we have shown how technology and lessons learned from autonomous driving can be used to increase safety, efficiency and convenience in driver-assistance systems. In particular, we have demonstrated why and how MPC, with its ability to handle multiple objectives and multiple constraints while maintaining real-time criteria, may be the preferred control algorithm in the next generation of cruise control systems. We have also described methods for providing sensor information about vehicles and other obstacles to such a system using passive (hence, noninterfering) and cost-effective visible-light and thermal-infrared cameras.

References

[1] "Intelligent Transportation Systems, U.S. Department of Transportation," Online: http://www.itsoverview.its.dot.gov/DAS.asp, downloaded 10. January 2008.

[2] U. Franke, D. Gavrila, A. Gern, S. Göorzig, R. Janssen, R. Paetzold, and C. Wöhler, *From door to door: principles and applications of computer vision for driver assistant systems*. Butterworth Heinemann, Oxford, 2001, ch. 6, pp. 131–188.

[3] W. D. Jones, "Keeping cars from crashing," *IEEE Spectrum*, vol. 38, no. 9, pp. 40–45, 2001.

[4] O. Carsten, "Beyond cruise control," *The Economist*, 2001.

[5] "Nissan press release," Online: http://www.nissan-global.com/EN/NEWS/2007/_STORY/071220-01-e.html, 2007, downloaded 10. January 2008.

[6] "DARPA Grand Challenge home page," Online: http://www.darpa.mil/grandchallenge/index.asp, downloaded 10. January 2008.

[7] L. Grossman, "Building the best driverless robot car," Time [online]: http://www.time.com/time/magazine/article/0,9171,1684543,00.html, November 2007, downloaded 10. January 2008.

[8] G. Sutton and R. Bitmead, "Performance and computiational implementation of nonlinear model predictive control on a submarine," in *Nonlinear Model Predictive Control*, ser. Progress in Systems and Control Theory, F. Allgöwer and A. Zheng, Eds. Birkhäuser, 2000, pp. 461–471.

[9] H. Kim, D. Shim, and S. Sastry, "Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles," in *Proceedings of the American Control Conference*, 2002, pp. 3576–3581.

[10] J. Eklund, J. Sprinkle, and S. Sastry, "Implementing and testing a nonlinear model predictive tracking controller for aerial pursuit/evasion games on a fixed wing aircraft," in *Proceedings of the American Control Conference*, 2005, pp. 1509–1514.

[11] E. Polak, *Optimization: Algorithms and Consistent Approximations*. Springer, 1997.

[12] R. Vidal, Y. Ma, S. Soatto, and S. Sastry, "Two-view multi-body structure from motion," *International Journal of Computer Vision*, vol. 68, no. 1, 2006.

[13] R. Vidal and S. Sastry, "Optimal segmentation of dynamic scenes from two perspective views," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2003, pp. 281–286.

[14] R. Vidal, S. Soatto, Y. Ma, and S. Sastry, "Segmentation of dynamic scenes from the multibody fundamental matrix," in *Proceedings of ECCV Workshop on Visual Modeling of Dynamic Scenes*, 2002.

[15] R. Vidal and Y. Ma, "A unified algebraic approach to 2-d and 3-d motion segmentation," *Journal of Mathematical Imaging and Vision*, vol. 25, no. 3, pp. 403–421, 2006.

[16] M. Han and T. Kanade, "Reconstruction of a scene with multiple linearly moving objects," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2000, pp. 542–549.

[17] A. Shashua and A. Levin, "Multi-frame infinitesimal motion model for the reconstruction of (dynamic) scenes with multiple linearly moving objects," in *Proceedings of IEEE International Conference on Computer Vision*, vol. 2, 2001, pp. 592–599.

[18] S. Avidan and A. Shashua, "Trajectorytriangulation: 3d reconstruction of moving points from a monocular image sequence," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 4, pp. 348–357, 2000.

[19] P. Sturm, "Structure and motion for dynamic scenes - the case of points moving in planes," in *Proceedings of European Conference on Computer Vision*, 2002, pp. 867–882.

[20] L. Wolf and A. Shashua, "Affine 3-d reconstruction from two projective images of independently translating planes," in *Proceedings of IEEE International Conference on Computer Vision*, 2001, pp. 238–244.

[21] T. Boult and L. Brown, “Factorization-based segmentation of motions,” in *Proceedings of IEEE Workshop on Motion Understanding*, 1991, pp. 179–186.

[22] K. Kanatani, “Motion segmentation by subspace separation and model selection,” in *Proceedings of IEEE International Conference on Computer Vision*, vol. 2, 2001, pp. 586–591.

[23] K. Kanatani and C. Matsunaga, “Estimating the number of independent motions for multibody motion segmentation,” in *Proceedings of European Conference on Computer Vision*, 2002, pp. 25–31.

[24] K. Kanatani and Y. Sugaya, “Multi-stage optimization for multi-body motion segmentation,” in *Proceedings of Australia-Japan Advanced Workshop on Computer Vision*, 2003, pp. 335–349.

[25] R. Vidal and R. Hartley, “Motion segmentation with missing data by PowerFactorization and Generalized PCA,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2004, pp. 310–316.

[26] Y. Wu, Z. Zhang, T. Huang, and J. Lin, “Multibody grouping via orthogonal subspace decomposition,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2001, pp. 252–257.

[27] L. Zelnik-Manor and M. Irani, “Degeneracies, dependencies and their implications in multi-body and multisequence factorization,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2003, pp. 287–293.

[28] L. Wolf and A. Shashua, “Two-body segmentation from two perspective views,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2001, pp. 263–270.

[29] P. Torr, “Geometric motion segmentation and model selection,” *Phil. Trans. Royal Society of London*, vol. 356, no. 1740, 1998.

[30] X. Feng and P. Perona, “Scene segmentation from 3d motion,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1998, pp. 225–231.

[31] C. Geyer, T. Templeton, M. Meingast, and S. S. Sastry, “The recursive multi-frame planar parallax algorithm,” in *Proceedings of Third International Symposium on 3D Data Processing, Visualization, and Transmission*, 2006.

[32] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis, “Graph clustering and minimum cut trees,” *Internet Mathematics*, vol. 1, no. 4, pp. 355–378, 2004.

[33] Y. Boykov and V. Kolmogorov, “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[34] D. Goldfarb and M. D. Grigoriadis, “A computational comparison of the Dinic and Network Simplex methods for maximum flow,” *Annals of Operations Research*, vol. 13, no. 1, pp. 81–123, 1988.

[35] H. C. Longuet-Higgins, “The visual ambiguity of a moving plane,” in *Proceedings of the Royal Society of London, Series B, Biological Sciences*, 1984.

[36] *Traffic safety facts 2000. A compilation of motor vehicle crash data from the Fatality Analysis Reporting System and the General Estimates System*. National Highway Traffic Safety Administration, Publication no. DOT HS 809 337. Washington, DC: NHTSA, 2002.

[37] “General motors press release,” Online: http://media.gm.com/division/cadillac/products/00cadillac/deville/00nightvision.htm, 2000, downloaded 10. January 2008.

[38] “BMW technology guide: BMW night vision,” Online: http://www.bmw.com/com/en/insights/technology/technology_guide/articles/bmw_night_vision.html, 2007, downloaded 10. January 2008.

[39] A. Nefian and M. Hayes, “Hidden markov models for face recognition,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1998, pp. 2721–2724.

[40] B. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” in *International Joint Conferences on Artificial Intelligence*, 1981, pp. 674–679.

[41] C. Tomasi and T. Kanade, “Detection and tracking of point features,” Carnegie Mellon University, Tech. Rep. CMU-CS-91-132, April 1991.