2008 IEEE Intelligent Vehicles Symposium
Eindhoven University of Technology
Eindhoven, The Netherlands, June 4-6, 2008

HyHOPE: Hybrid Head Orientation and Position Estimation for Vision-based Driver Head Tracking

Erik Murphy-Chutorian and Mohan Manubhai Trivedi
Laboratory for Intelligent and Safe Automobiles
Department of Electrical and Computer Engineering
University of California, San Diego
{erikmc,mtrivedi}@ucsd.edu

Abstract— **Driver distraction and inattention are prominent causes of automotive collisions. To enable driver assistance systems to address these problems, we require new sensing approaches to infer a driver's focus of attention. In this paper, we present a new 3D tracking algorithm and integrate it into HyHOPE, a real-time (30fps) Hybrid Head Orientation and Position Estmation system for driver head tracking. With a single video camera, the system continuously tracks the head in six degrees-of-freedom, initializing itself automatically with separate modules for head detection and head pose estimation. The tracking module provides a fine estimate of the 3D motion of the head, using a new appearance-based algorithm for 3D-model tracking by particle filtering in an augmented reality environment. We describe our implementation, which utilizes OpenGL-optimized graphics hardware to efficiently compute particle samples in real-time. To quantitatively evaluate the accuracy of our system, we compare its estimation results to a marker-based cinematic motion capture system installed in an automotive testbed. We evaluate the system on real daytime and nighttime drives with drivers of varying ages, race, and sex.**

I. Introduction

Vehicular safety relies on the ability of people to maintain a constant awareness of the environment as they drive. As new vehicles and obstacles move into the vicinity of the car, a driver must be cognizant of the change and ready to respond as necessary. Although people have an astounding ability to cope with these changes, a driver is fundamentally limited by the field-of-view that he can observe at any one time. When a driver fails to notice a change to his environment, there is an increased potential for a life-threatening collision. It is reasonable to assume that this danger could be mitigated if the driver was notified when these situations arise. As evidence to this effect, a recent comprehensive survey on automotive collisions demonstrated a driver was 31% less likely to cause an injury-related collision when he had one or more passengers who could alert him to unseen hazards [1]. Consequently, there is great potential for driver assistance systems that act as virtual passengers, alerting the driver to potential dangers. To design such a system in a manner that is neither distracting or bothersome, these systems must act like real passengers, alerting the driver only in situations where he appears to be unaware of the possible hazard. This requires a context-specific system that is not only aware of the environment, but one that is also capable of actively interpreting the behavior of the driver and fusing this information from inside and outside the vehicle. [2], [3].

With consideration for future driver assistance systems, we concentrate on one of the integral processes for monitoring driver awareness: estimation of the position and orientation of a driver's head. Head pose is a strong indicator of a driver's field-of-view and current focus of attention. It is intrinsically linked to visual *gaze estimation*, the ability to characterize the direction in which a person is looking. Intuitively, it might seem that looking at the driver's eyes might provide a better estimate of gaze direction, but in practice eye-gaze systems often require that the head pose is known in advance. In addition, a vision system that focuses on a driver's eyes requires multiple high-resolution cameras (to view the eye from all head positions) and cannot operate when the driver is wearing sunglasses, for instance. In our application we desire a single camera solution that estimates gaze direction without these restrictions.

Computational head pose estimation remains a challenging vision problem without inexpensive and widely-available solutions. Amongst the research thrusts and commercial offerings that can provide a real-time estimate of head pose, most require multiple cameras to obtain correspondence-based depth information, and none have been rigorously and quantitatively evaluated in an automobile. In a car, ever-shifting lighting conditions cause heavy shadows and illumination changes and as a result, techniques that demonstrate high proficiency in stable lighting often will not work in this challenging environment. In this paper, we address these concerns with three novel contributions. First, we introduce a new algorithm for identity and lighting invariant head pose tracking in an augmented reality. Second, we combine the algorithm with head detection and static head pose estimation modules to create HyHOPE, a fully-automatic Hybrid Head Orientation and Position Estimation system. In this implementation, we use only a single video camera and provide real-time (30fps) operation by performing the calculations with the parallel processing capabilities available from a consumer-level graphics processor. Third, we quantitatively demonstrate the success of our system on the road, comparing our system to ground truth obtained with a professional cinematic motion capture system that we have configured for a vehicular testbed. To ensure a wide variety of driving conditions, we perform these experiments with 14 drivers of varying age, race, and sex spanning daytime and nighttime drives.

II. Prior Work

Head tracking algorithms operate on continuous video, estimating head pose by inferring the change in pose between consecutive frames of a video sequence. They exploit temporal continuity and smooth motion constraints to provide a visually appealing estimate of pose over time and demonstrate much higher levels of accuracy than static pose estimation methods that operate on individual images. In bottom-up tracking systems, low-level features can be followed from frame to frame. From this local motion, a

978-1-4244-2569-3/08/$20.00 ©2008 IEEE.

global pose transformation can be estimated, for example, by assuming the human face is a planar surface and using weighted least-squares to determine the best affine transformation between any two frames [4]. In top-down systems, a global transformation is found best describes the new observations. With stereo imagery, for instance, the head pose can also be obtained with affine transformations by finding the translation and rotation that minimize the discrepancy in grayscale intensity and depth [5].

Besides finding the transformation that minimizes the appearance between the model and the new camera frame, systems can also incorporate prior information about the dynamics of the head. Particle filters provide an approximation of the optimal track by maximizing the posterior probability of the movement from a simulated set of samples. Variations on particle filtering have been applied to head accurate real-time head pose tracking in varying environments, including low-resolution video with adaptive PCA subspaces [6], and near-field stereo with affine approximations [7]. In this paper, we introduce a new dual-state particle filter to explicitly model the nonlinear motion of a driver. This motion model is able to simultaneously account for the observed jitter of a driver's head and the driver's intentional head movements. Compared to other particle tracking approaches, we have overcome many simplifications and limitations such that

1) we require only monocular video, satisfying our design criterion and preventing the need for periodic stereo calibration;
2) we compute full projective transformations of the objects, rather than affine approximations, improving performance by removing an artificial source of distortion;
3) we use a full textured-mapped 3D model instead of a series of point samples, allowing a more complete comparison between the model and the observation;
4) we provide a real-time implementation, satisfying our design criterion for 30fps tracking.

The head pose estimation system that we propose combines a static head pose estimator [8] with our new head tracker. In this sense, it is a hybrid approach that combines the initialization and stability properties of a static pose estimator with the high-accuracy and real-time capabilities of a tracking approach. This provides a solution to the important requirement that tracking be initialized from a known head position. It also provides a solution for reinitialization after drifting or losing track. Although very different in composition and scope, other works have espoused the advantages of hybrid systems as well [5], [9], [10].

The system that we present in this paper is a novel software engine that advances the state-of-the-art in fully-automatic head pose estimation. This system has practical utility for many applications, but we have focused our efforts on the automotive domain. To demonstrate the capability of our system, we have compared it to a marker-based cinematic motion capture system on a wide range of natural driving situations. Although there have been other head pose estimation systems that have been applied to automotive imagery [11]–[16], they have been evaluated when the car is moving in specific scenarios only in situations where the car is not moving (indoors). It is not clear whether or not these approaches would require substantial modification to become viable options for real automotive use. In contrast, the data collection and evaluation procedure that we describe in this paper is the first its kind, and we are able to demonstrate that our system attains a high level of accuracy during real-world driving.

The remainder of this paper is structured as follows: Section III introduces our augmented-reality head tracking algorithm. Section IV describes our hybrid head pose system and the real-time implementation of the tracker using optimized consumer-grade graphics hardware. Section V introduces our automotive testbed and presents an evaluation of our methods, and Section VI contains our concluding remarks.

III. Head Pose Tracking in Augmented Reality

We present a new procedure to track the driver's head in six degree-of-freedom at 30fps from a single video camera. Our approach uses an appearance-based particle filter in an *augmented reality*, a virtual environment that mimics the view-space of a real camera. Using an initial estimate of the head position and orientation, the system generates a texture-mapped 3D model of the head from the most recent video image and places it into the environment. The model is subsequently rotated, translated, and rendered in perspective projection to match the view from each subsequent video frame. It would be computationally inefficient to exhaustively search for the best transformation, so instead we introduce an appearance-based particle filter framework to generate a set of virtual samples that together provide an optimal estimate of this transformation. The virtual samples are perspective projections of the head model at a specific rotation and translation, and resemble small perturbations of the driver's face set against a solid background.

Although the 3D construction and evaluation of these samples is a daunting computational challenge for a conventional computer processor, we show that it can be highly optimized for Graphics Processing Units (GPUs), and in the following section will present a real-time implementation that utilizes the 3D virtualization and processing capabilities of a consumer-level GPU. In this section, Part A describes our dual-state motion model, and Part B details our particle filtering approach to update this model.

A. State Model

We represent the driver's head as rigid object constrained to six degrees-of-freedom in a 3D world. This can be represented with respect to a fixed Cartesian coordinate system by the position, (x, y, z) and Euler angles, (α, β, γ). To model the system using linear dynamics, we define the state, $\boldsymbol{x}_t = [\boldsymbol{\theta_t} \;\; \boldsymbol{\omega_t}]^T$, where $\boldsymbol{\theta}_t$ represents the position and angle of the object at time t, and $\boldsymbol{\omega}_t$ represents the respective linear and angular velocity. In our head tracking application, however, motion is not well described by a linear system. Consider the typical motion of a person's head bobbling about in an automobile. For the most part, the subject is focused on a single location in the world, and his head is essentially static, subject only to small perturbations that can appear to be instantaneous when viewed at a sampling rate defined by a video camera. Only when the person conscientiously moves their head from one position to another can linear dynamics provide a good temporary approximation of the motion. The first situation can be modeled with a zero-velocity state model,

$$\boldsymbol{x}_t^{(ZV)} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \boldsymbol{x}_{t-1} + \begin{bmatrix} \boldsymbol{\nu}_t \\ \mathbf{0} \end{bmatrix}, \qquad (1)$$

where $\boldsymbol{\nu}_t$ is a vector-valued random sample from an i.i.d.[1] stochastic sequence that accounts for small instantaneous displacements of the head. The second situation can be described by a constant-velocity model,

$$\boldsymbol{x}_t^{(CV)} = \begin{bmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \boldsymbol{x}_{t-1} + \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\eta}_t \end{bmatrix}, \tag{2}$$

where $\boldsymbol{\eta}_t$ is a vector-valued sample from another i.i.d. stochastic sequence that accounts for any change in velocity of the head. At a practical level, we do not need to estimate whether the head is in a zero-velocity or constant-velocity mode, since we are only interested in the position and orientation of the head. Instead, these two models simultaneously constitute a mixed prior probability for the motion of the head.

To accommodate both of these motion models, we define the augmented object state, $\boldsymbol{y}_t = \{\boldsymbol{x}_t, \xi_t\}$, where ξ_t is a binary variable $\{\xi_t : \xi_t \in 0, 1\}$ that specifies the object motion model at time t,

$$\boldsymbol{x}_t = (1 - \xi_t)\boldsymbol{x}_t^{(ZV)} + \xi_t \boldsymbol{x}_t^{(CV)}. \tag{3}$$

We can model ξ_t as a Markov Chain, drawing each new sample ξ_t from a probability distribution $f(\cdot)$ that depends only on the previous state,

In a classical tracking problem, the object's state, $\boldsymbol{y}_t$, is observed at every time step, but the observation is assumed to be noisy, and the optimal track can be found by maximizing the posterior probability of the movement given the previous states and observations. For a Markovian system that is perturbed by non-Gaussian noise, a Sampling Importance Resampling (SIR) particle filter offers a practical approach that approximates the optimal track as a weighted sum of samples. These samples are drawn from the state transition density $p(\boldsymbol{y}_t|\boldsymbol{y}_{t-1})$, and the weight is set proportional to the posterior density of the observation given the samples. In our vision-based tracking problem, instead of observing a noisy sample of the object's state, we observe an image of the object. The observation noise is negligible, but the difficulty lies in inferring the object's state from the image pixels. The solution to this problem can be estimated using a similar SIR construction. We generate a set of state-space samples and use them to render virtual image samples using the fixed-function pipeline of a GPU. Each virtual image can be directly compared to the observed image, and these comparisons can be used to update the particle weights.

Given the existence of a set of N samples with known states, $\{\boldsymbol{y}_t^{(l)} : l \in 0, \cdots, N-1\}$, we can devise the observation vector,

$$\boldsymbol{z}_t = \begin{bmatrix} d(\boldsymbol{y}_t, \boldsymbol{y}_t^{(0)}) \\ \vdots \\ d(\boldsymbol{y}_t, \boldsymbol{y}_t^{(N-1)}) \end{bmatrix}, \tag{4}$$

where $d(\boldsymbol{y}, \boldsymbol{y}')$ is an image-based distance metric. As with a classical SIR application, we are required to maintain and update a set of samples with a known state at every time step. We use these samples to update our observation vector.

As a potential image-comparison metric, normalized cross-correlation (NCC) provides an appealing approach for comparing two image patches, having the desirable property that it is invariant to affine changes in pixel intensity in either patch. Given two image patches specified as M-dimensional vectors of intensity, $\boldsymbol{\phi}$ and $\boldsymbol{\phi}'$, we can specify a NCC-based distance metric as follows:

$$d_{NCC}(\boldsymbol{\phi}, \boldsymbol{\phi}') = 1 - \frac{1}{\sqrt{\sigma_\phi^2 \sigma_{\phi'}^2}} \sum_{i=0}^{M-1} (\phi_i - \mu_\phi)(\phi_i' - \mu_{\phi'}), \tag{5}$$

where μ_ϕ is the mean intensity and σ_ϕ^2 is the variance of the intensity. The unit constant and minus sign are introduced in (5) provide a distance measure in the range $[0, 2]$.

When the lighting variation is non-affine (e.g., specular reflections, shadowing, etc.), NCC performs poorly as a global image metric. If the image patches are small enough, however, it is likely that they will be locally affine. Therefore, better invariance to globally non-uniform lighting can be gained by using the average of a series of P small image patch NCC comparisons spread out over the object of interest. This is the basis of the mean normalized cross-correlation (MNCC) metric that we use in our tracking system,

$$d(\boldsymbol{y}, \boldsymbol{y}') = \frac{1}{P} \sum_{p=0}^{P-1} d_{NCC}(\boldsymbol{\phi}_p, \boldsymbol{\phi}_p'). \tag{6}$$

We can directly relate these comparisons to the conditional observation probability if we can model the distribution such that it depends only on the current sample,

$$p(\boldsymbol{z}_t|\boldsymbol{y}_t^{(l)}) \propto h(z_{t,l}, \boldsymbol{y}_t^{(l)}), \tag{7}$$

where $z_{t,l}$ is the lth component of $\boldsymbol{z}_t$, and $h(\cdot, \cdot)$ is any valid distribution function. In our head tracking system, we model the observation probability as a truncated Gaussian envelope windowed by the displacement between the current sample state and the sample with the smallest MNCC distance.

A Sampling Importance Resampling particle filter is a Monte Carlo estimation method based on stochastic sampling [17] that, regardless of state model, converges to the Bayesian optimal solution as the number of samples increases towards infinity.

To update our state model with every new image frame, we use the following steps that constitute a full iteration of the SIR filter:

1) Update samples : $\boldsymbol{y}_t^{(l)} \sim p(\boldsymbol{y}_t|\overline{\boldsymbol{y}}_{t-1}^{(l)})$

2) Calculate weights : $c_t^{(l)} = \dfrac{p(\boldsymbol{z}_t|\boldsymbol{y}_t^{(l)})}{\sum_{l=0}^{N-1} p(\boldsymbol{z}_t|\boldsymbol{y}_t^{(l)})}$

3) Estimate state : $\widehat{\boldsymbol{x}}_t = \sum_{l=0}^{N-1} c_t^{(l)} \boldsymbol{x}_t^{(l)}$

4) Resample : $\overline{\boldsymbol{y}}_t^{(l)} \sim \rho(\overline{\boldsymbol{y}}_t|c_t^{(0:N-1)}, \boldsymbol{y}_t^{(0:N-1)})$

B. 3D Model

We represent the driver's head in our augmented reality framework as a texture-mapped 3D model. The model consists of 3D vertices that define a set of convex polygons approximating the surface of the object. Each vertex is assigned a texture coordinate that corresponds to a position in a 2D image texture.

To create a new model and place it in the environment, we require a new set of polygons and an image texture. For our approach, we use a rigid anthropometric head model

[1] i.i.d. - independent and identically-distributed

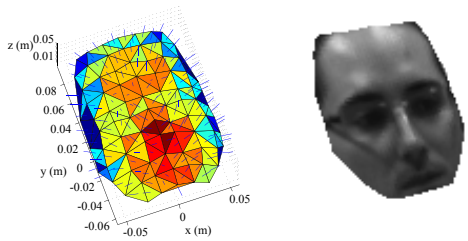

Fig. 1. (left) The rigid facial model using for initialization and tracking. (right) An example of the model as rendered by the tracking system.

shown in Fig. 1. This model was created from a person excluded from the driving experiments, and although this single model is only an approximation of the facial shape of each driver, the texture-based tracking approach does not require a highly-accurate fit. We place this model in the virtual environment with an inverse projection that puts it at the depth that corresponds to the observed width of the detected face. The static pose estimate is used to assign the initial model Euler angles. To ensure a symmetric view of the head, we only initialize the model if the estimated head pose is within 25° of the center; otherwise the initialization is skipped until this constraint is satisfied.

IV. Hybrid Head Pose Estimation

Our proposed system uses a hybrid pose estimation scheme, combining a static head pose estimator with a real-time, 3D model-based tracking system. The static estimator initializes the tracker from a single image frame and, as the head is tracked, continues to run in parallel, providing a periodic consistency check. If the tracking confidence falls below a threshold or the consistency check fails, the static estimator automatically reinitializes the tracker. The static head pose estimation algorithm and head detection algorithm have been described in our prior published work [8]. Briefly, the head is detected with three cascaded-Adaboost [18] face detectors applied to the grayscale video images. For static head pose estimation, a Localized Gradient Orientation histogram is passed to three Support Vector Regressors trained for pitch, roll, and yaw [8]. The face detection and static head pose estimation modules are run once to initialize the tracker and periodically repeated to check the consistency of the tracking estimate.

The tracking system has been optimized to run on a GPU. First we use the intrinsic parameters from our camera to model the perspective projection in the augmented reality. Many software packages are available to estimate the intrinsic parameters and remove any spherical lens distortion. We refer the reader to [19] for more information. In OpenGL, this entails a conversion from world coordinates to *normalized view volume* coordinates.

After each new video frame is captured, it is copied into a texture object on the GPU. Then for each generated sample in the particle filter, the following steps are performed to calculate the sample weight. First, the 3D head model vertices are rotated and translated as described by the sample state. Next, the object is rendered to an off-screen framebuffer object using the fixed-function GPU pipeline, (i.e., the basic procedure for rendering an object with the graphics API).

Computing the MNCC distance metric described in (6) requires many computationally intensive pixel calculations. To perform this operation at 30fps, we implement the operation using the programmable functionality of the GPU, using the

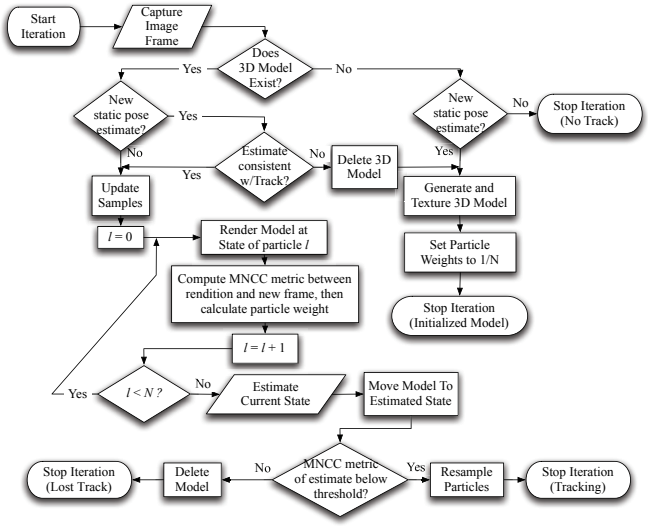

Fig. 2. Flowchart illustrating one iteration in HyHOPE. There are four potential results of each iteration, denoted with the phase: "Stop Iteration".

standards-approved OpenGL Shading Language. To begin, the head model is rendered a second time with a few notable exceptions. First, texture-mapping is disabled, causing the GPU to run in a mode where each vertex is assigned an RGB color. Secondly, rather than drawing polygons that make up the head model, we render the vertices as individual points that compute NCC of a local image patch using the programmable pipeline. More specifically, we have created a *vertex shader*, which operates on each vertex as is passes through the GPU pipeline.

A vertex shader has the ability to change the position, texture coordinates, and color of the vertex, and it can sample from textures and perform a variety of computational tasks. Modern GPUs contain as many as 128 shaders capable of parallel operation. In our application, the shader first calculates the 2D projection of the 3D vertex position. Then it performs the NCC of a 9×9 pixel image patch centered on this location in both the new image frame texture and the newly rendered texture. Computationally, this requires looping through the pixels of each patch twice – first to establish the mean and standard deviation of each patch and then to calculate the NCC as described in (5). A vertex shader is limited to standard graphical outputs, but we overcome this restriction by coding the NCC into the RGB color output of the vertex,

$$RGB_v = \left[\; d_{NCC}(\boldsymbol{\phi}_v, \boldsymbol{\phi}'_v) \quad 1.0 \quad 0.0 \; \right]^T . \qquad (8)$$

By setting the G component equal to 1.0 and ensuring a 32-bit floating-point framebuffer, a summation of the output from multiple shaders will effectively count of the number of times that the procedure is performed. Any pixels in the correlation window that lie outside the rendered object are skipped during the NCC calculation, and if fewer than half of the pixels remain, the calculation is terminated and the color is set to $RGB_v = [0.0, 0.0, 0.0]^T$.

For the last step in the shader program, the output position of the vertex is set to the lth framebuffer pixel, where l is the particle number. This will make every fragment in the particle overlap, and with the use of an alpha blending function, this will compute the summation of all the localized NCC calculations. This two-pass GPU process is repeated for every particle sample, and afterward, the MNCC filtering

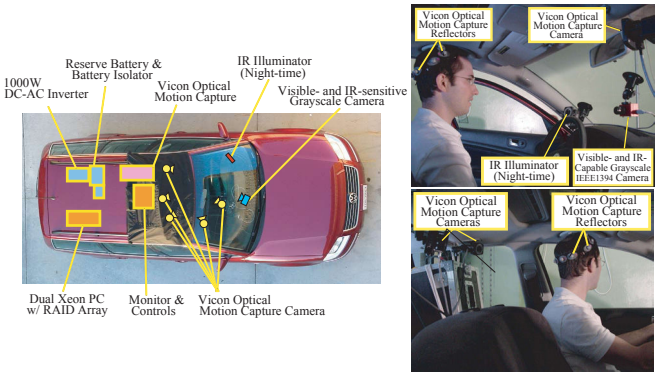

Fig. 3. The LISA-P experimental testbed is a modified Volkswagen Passat equipped with a mobile computing platform and sensors for motion and video capture [20]. These images depict the experimental setup used for to simultaneously capture video of the driver while collecting accurate head pose information.

observation vector, $\boldsymbol{z}_t$, can be derived by decoding the result from each framebuffer pixel, $\boldsymbol{\psi}_t(l)$. This is accomplished by dividing the NCC summation stored in the pixel's R component by the count stored in the G component,

$$\boldsymbol{z}_t = \begin{bmatrix} d(\boldsymbol{y}_t, \boldsymbol{y}_t^{(0)}) \\ \vdots \\ d(\boldsymbol{y}_t, \boldsymbol{y}_t^{(N-1)}) \end{bmatrix} = \begin{bmatrix} \frac{\psi_{t,R}(0)}{\psi_{t,G}(0)} \\ \vdots \\ \frac{\psi_{t,R}(N-1)}{\psi_{t,G}(N-1)} \end{bmatrix}. \quad (9)$$

To ensure that these calculations run as fast as possible, only one image upload (glTexSubImage2D) is necessary to copy the new image frame, one download (glReadPixels) to get the sample weights, and one download (glReadPixels) to obtain the confidence score. Our code makes heavy use of display lists, allowing us to cache all of the vertices and drawing calls that are required to render the model, as well as the state changes between the two fixed-function and programmable-function steps.

V. Data and Evaluation: LISAP-14

To evaluate our approach, we created the LISAP-14 dataset consisting of 14 drives with varying environmental conditions. The LISA-P experimental testbed, as seen in Fig. 3, was used to collect real-world test data. An IEEE1394 camera mounted on the windshield captures face data as seen in Fig. 3. This camera provides a 640x480 pixel grayscale video stream at 30fps, and like most CCD imagers, it is naturally sensitive to both visible and near-infrared light. For our specific camera, we were required to physically remove an infrared filter installed in front of the imager.

For the purpose of illuminating the driver's face and stabilizing the lighting conditions at nighttime, a near-infrared illuminator is attached to the left-most part of the windshield. Since the emitted light is of not part of the visible spectrum, it does not serve as a distraction or cause any glare for the driver. In addition, the vehicle is instrumented with a Vicon optical motion capture system, with multiple sensors placed behind the driver's head. This marker-based system is used to gather precise ground truth head pose data for evaluation. To prevent the reflective markers from appearing in the video, we created a small headpiece for the subjects to wear on the back of their head as shown in Fig. 3.

For the LISAP-14 dataset, we asked 14 subjects to drive the LISA-P while wearing the motion capture headpiece. The subjects consisted of 11 males and 3 females, spanning Caucasian, Asian and South-Asian descent. The subjects ranged from 15 to 53 years of age, and five of them wore glasses.

Each of the subjects drove the vehicle on different round-trip routes through the University of California, San Diego campus at different times, including drives from the morning, afternoon, dusk and night. The cameras were set to auto-gain and auto-exposure, but these adjustments have to compete with ever-shifting lighting conditions and dramatic lighting shifts (e.g. sunlight diffracting around the driver or headlights from a neighboring vehicle) that on occasion completely saturate the image. All of these situations remain part of our evaluation, as they are typical phenomena that occur in natural driving.

The automobile was set up to collect data during two periods, half during the summer and half during the winter. The placement of the cameras varies mildly between these two setups. The drives averaged 8 minutes in duration, and we analyzed our full hybrid system on all of the data, amounting to approximately 200,000 video frames in all. To train the static head pose estimator, we extracted a uniform sampling of the pose space for pitch, yaw, and roll separately, (approximately 300 images from each 10-degree interval where available, and all of the data from intervals with fewer than 300 images), and used a cross-validation scheme to train with the data from 13 of the subjects, leaving the remainder for evaluation. This was repeated for every all-but-one combination.

We evaluate our tracking system on the full LISAP-14 video footage obtained from all the drivers. In each video sequence, the tracker will follow the driver's head, and if the track is lost, the system automatically reinitializes based on the static pose estimator. To evaluate the steady-state error of our tracking algorithm separately from the entire hybrid system, we isolate the tracking error from the initialization error. This is accomplished by subtracting the mean position and orientation from the ground truth and observed track before calculating the mean absolute error between the two. We also exclude the 3.59% of frames in which the tracker has suffered catastrophic error due to a clearly lost track. We quantify these as the frames where the tracked head orientation deviates more than 30° from the true pitch, yaw, or roll. The results of these experiments are presented in Table I. The table includes the mean absolute error and the standard deviation of error position and head orientation. Since this is the first system to be evaluated on this data, we cannot direct compare these results to other systems. Nevertheless, our results on this challenging data are within one or two degrees of error of published results from much simpler datasets (i.e. indoors with only a few subjects) [5], [7]. In addition, these comparative systems require calibrated stereo cameras, while our system uses a single camera.

To evaluate the combined error from detection, initialization, and tracking, we compare the output of the system directly to the motion capture ground truth. These results are presented in Table II. This table contains only angular

TABLE I

Isolated Tracking Error on LISAP-14

Statistic	Translation (cm)			Rotation (degrees)		
	x	*y*	*z*	*Pitch*	*Yaw*	*Roll*
Mean absolute error	0.92	0.88	1.44	3.39	4.67	2.38
Std. deviation of error	1.87	1.22	2.37	4.71	6.89	3.74

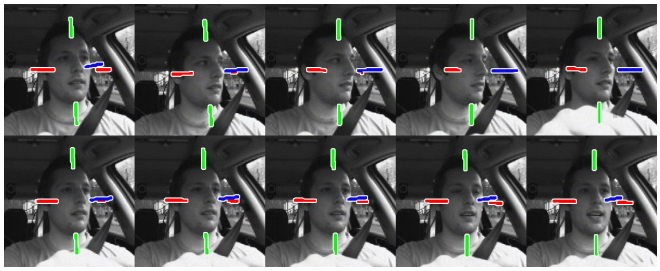

Fig. 4. Example images of tracking in a LISAP-14 video sequence. The images have been cropped around the driver's face to highlight the tracking estimate, which is indicated by the overlaid 3D axes.

TABLE II

COMBINED INITIALIZATION AND TRACKING ERROR ON LISAP-14

	Rotation (degrees)		
Statistic	*Pitch*	*Yaw*	*Roll*
Mean absolute error	8.57	11.24	8.29
Standard deviation of error	16.43	16.90	15.27

evaluation, since the ground truth is ambiguous as to the exact position of the face, which is not the same as the position of the motion capture headpiece.

A tracking example is shown in Fig. 4. We also include an example video of the running system as supplementary material on our website at `http://cvrr.ucsd.edu/LISA/hyhope.html` We encourage the readers view this material as it provides a better visualization of the system than is possible with the images alone.

VI. CONCLUSIONS

Robust systems for observing driver behavior will play a key role in the development of advanced driver assistance systems. In combination with environmental sensors, cars can be designed with the ability to supplement the driver's awareness, preempting and preventing hazardous situations. In this work, we focused on an automotive head pose estimation and tracking, since head pose is a strong indicator of a driver's field-of-view and current focus of attention. The HyHOPE system satisfies all of our design criteria, as it requires only monocular video for autonomous, real-time, identity-invariant, and lighting-invariant driver head pose estimation. It is an advancement in the state-of-the-art, providing fine head pose estimation and ease of use.

Further extensions to this system could focus on model augmentation, since the initial model represents only the slice of the head that was visible from the perspective of a single camera when the model was created. As the head rotates, this region shifts out of view until there is very little texture remaining to continue the tracking. As a result, the track becomes less reliable as the yaw of the head approaches 90° in either direction. As a possible solution, the model can be augmented by adding additional sets of polygons and textures. During tracking, if the rotation angle between the sample and the initial model exceeds a threshold and the MNCC score is sufficiently large to indicate an accurate track, the initialization step can be repeated to add new polygons with a new texture to augment the original model. Care should be taken to prevent adding polygons that can overlap the existing model, and during this augmentation process, the global position and orientation do not need to be re-estimated, since the are already established by the particle filter.

In conclusion, the HyHOPE system comprises a new method for estimating the pose of a human head that overcomes the difficulties inherent with varying lighting conditions in a moving car.

ACKNOWLEDGMENTS

We thank the Volkswagen Electronics Research Laboratory and the U.C. Discovery program for their support of this research. We thank Mr. Anup Doshi for his help with experimental setup and data collection and Mr. Shinko Cheng for his contributions in the design and development of the LISA-P testbed and motion capture system. In addition, we thank the rest of our team members in the Computer Vision and Robotics Research laboratory for their participation as test subjects and for their continuing support, comments, and assistance.

REFERENCES

[1] T. Rueda-Domingo, P. Lardelli-Claret, J. L. del Castillo, J. Jiménez-Moleón, M. García-Martín, and A. Bueno-Cavanillas, "The influence of passengers on the risk of the driver causing a car collision in spain," *Accident Analysis & Prevention*, vol. 36, no. 3, pp. 481–489, 2004.

[2] M. Trivedi, T. Gandhi, and J. McCall, "Looking-in and looking-out of a vehicle: Computer-vision-based enhanced vehicle safety," *IEEE Trans. Intelligent Transportation Systems*, vol. 8, no. 1, pp. 108–120, 2007.

[3] S. Cheng and M. Trivedi, "Holistic sensing and dynamic active displays," *IEEE Computer*, vol. 40, no. 5, pp. 60–68, 2007.

[4] P. Yao, G. Evans, and A. Calway, "Using affine correspondence to estimate 3-d facial pose," in *Proc. Int'l. Conf. Image Processing*, 2001, pp. 919–922.

[5] L.-P. Morency, A. Rahimi, and T. Darrell, "Adaptive view-based appearance models," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2003, pp. 803–810.

[6] J. Tu, T. Huang, and H. Tao, "Accurate head pose tracking in low resolution video," in *Proc. IEEE Int'l. Conf. Automatic Face and Gesture Recognition*, 2006, pp. 573–578.

[7] K. Oka, Y. Sato, Y. Nakanishi, and H. Koike, "Head pose estimation system based on particle filtering with adaptive diffusion control," in *Proc. IAPR Conf. Machine Vision Applications*, 2005, pp. 586–589.

[8] E. Murphy-Chutorian and M. Trivedi, "Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation," in *Proc. IEEE Conf. Intelligent Transportation Systems*, 2007, pp. 709–714.

[9] T. Jebara and A. Pentland, "Parametrized structure from motion for 3d adaptive feedback tracking of faces," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1997, pp. 144–150.

[10] K. Huang and M. Trivedi, "Robust real-time detection, tracking, and pose estimation of faces in video streams," in *Proc. Int'l. Conf. Pattern Recognition*, 2004, pp. 965–968.

[11] R. Pappu and P. Beardsley, "A qualitative approach to classifying gaze direction," in *Proc. IEEE Int'l. Conf. Automatic Face and Gesture Recognition*, 1998, pp. 160–165.

[12] Y. Zhu and K. Fujimura, "Head pose estimation for driver monitoring," in *IEEE Intelligent Vehicle Symposium*, 2004, pp. 501–506.

[13] J. Wu and M. Trivedi, "Visual modules for head gesture analysis in intelligent vehicle systems," in *IEEE Intelligent Vehicle Symposium*, 2006, pp. 13–18.

[14] K. Huang and M. Trivedi, "Driver head pose and view estimation with single omnidirectional video stream," *Automation and Remote Control*, vol. 25, pp. 821–837, 2006.

[15] Z. Guo, H. Liu, Q. Wang, and J. Yang, "A fast algorithm face detection and head pose estimation for driver assistant system," in *Proc. Int'l Conf. Signal Processing*, vol. 3, 2006.

[16] S. Baker, I. Matthews, J. Xiao, R. Gross, T. Kanade, and T. Ishikawa, "Real-time non-rigid driver head tracking for driver mental state estimation," in *Proc. 11th World Congress Intelligent Transportation Systems*, 2004.

[17] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[18] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2001, pp. 511–518.

[19] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[20] S. Cheng and M. Trivedi, "Turn-intent analysis using body pose for intelligent driver assistance," *IEEE Pervasive Computing*, vol. 5, no. 4, pp. 28–37, 2006.