Multi-Camera System Based Driver Behavior Analysis Final Report, Winter 2013

Jianbo Shi, Yedong Niu, Qiong Wang

December 16, 2013

1 Introduction

Understanding driver behavior is an essential component in human-centric driver systems. Particularly, driver's interaction with the environment is an important factor in controlling the vehicle, though there have been very few research studies on analyzing driver behavior. Multi-camera array system has a variety of applications because of its improved resolution, frame rate, depth of field, dynamic range and disparity map from such system. In this report we present an implementation of multi-camera array system with GoPro cameras to interact with the external environment of a moving vehicle on streets. So far, our major contribution contains specific analysis of GoPro[2] hardware and protocol, integrating the system with various sensors to collect both internal and external environment information. We also introduce a calibration and rectification method with bundle adjustment for the multi-camera array and optimize the calibration algorithm of First Person Vision glasses [1]. Our goal is to implement realtime intension prediction of drivers with the multi-camera array system.

1.1 Background

Human behavior analysis based on computer vision is a challenging but valuable research field with lots of promising applications, such as image understanding, intelligent environment system, interaction between human and computer, and so on. Generally, human behavior can be analyzed focused on different levels of the human body such as full body level[8], lower body[7], hand[9], head[4] and foot[6]. For the behavior analysis of drivers on a vehicle, both the information of the driver's behavior and the external circumstances are important. Considering the complexity of external environment, we need a system with high performance to percept it. Multi-camera array system can meet the requirements quite well.

Multi-camera system is synchronized camera array in certain geometric arrangement, which can function in various applications depending on the configuration of the camera. The system could be classified into single-center-of-projection synthetic camera and multiple-center-of-projection camera based on the geometric configuration[11]. Potential applications include but not limited to improvement of resolution, signal-to-noise ratio, depth of field and reconstruction of occluded environment. On the other hand, with the advent of low cost, high resolution compact cameras with synchronization capability such as GoPro cameras[10], building a multi-camera array system with higher resolution and frame rate, lower cost and complexity becomes available. Under the circumstances, GoPro cameras become an intellectual choice to prototype such multi-camera array system with reasonable cost and performance [5]. The potential benefit from the system and the feasibility of the construction become the motivation of the project.

1.2 Purpose

In this study, we want to develop a vision-based framework for driver behavior analysis. By collecting relevant human driver data from instrumented test vehicle, we want to analyze and develop driver intention algorithms to predict and estimate the next behavior of the driver.

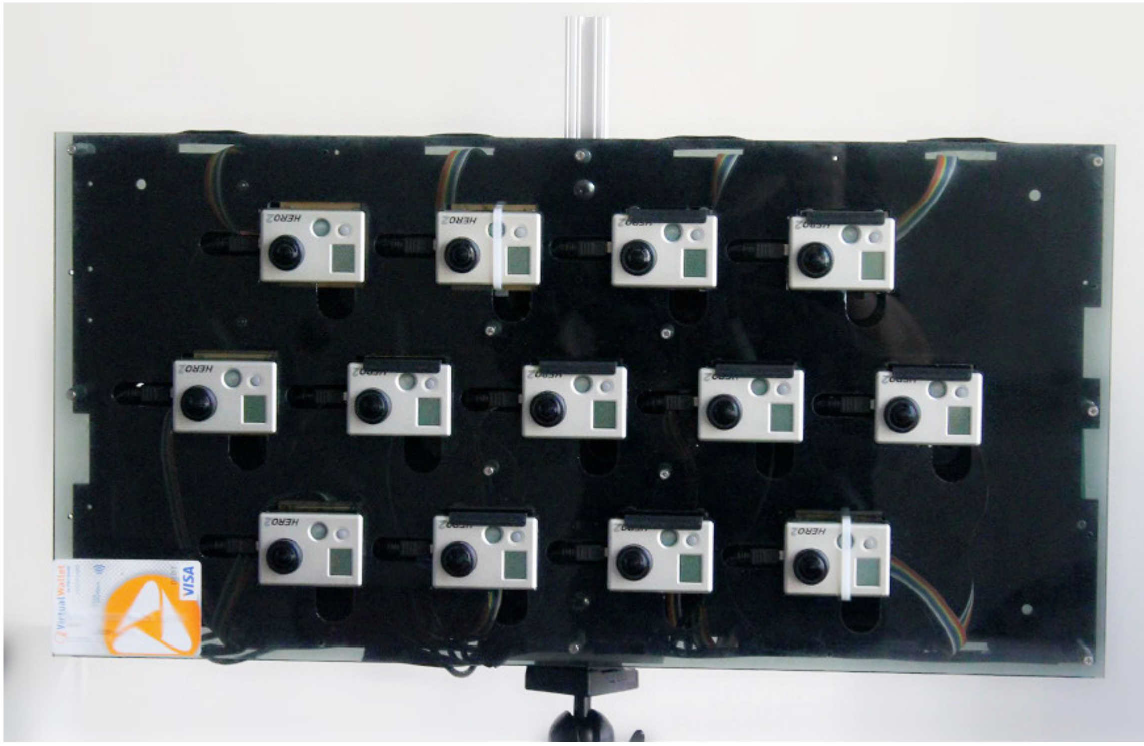

Figure 1: *Dimension of the camera array*

2 Design of the Study

The camera array system contains 13 GoPro HERO 2 cameras, glasses double-directional cameras, kinect and assistant IMU, GPS and LIDAR sensors. We address the hardware design into mechanical and electrical design in the following sections. In software section we introduce the synchronization protocol based on reverse engineering results. We will leave calibration and rectification of the camera array in the following part.

2.1 Hardware

2.1.1 Mechanical Design

We arranged 13 GoPro HERO 2 cameras in a triangular 4-5-4 configuration with a baseline of 100mm in the same plane. Here, each camera was mounted upside down to accommodate the slave mode of GoPro firmware as shown in Figure 1. We laser-cut mounting holes on the battery cap and mounted it on an acrylic sheet with stand-offs. Such design is capable for various planar and circular configurations. To maintain rigid structure another piece of acrylic sheet is covered in front of the camera by press fit. This is particularly crucial for platforms with vibrations. The system contains T-slot aluminum beam as major frames, which is able to be mounted both on tripod to collect calibration data, and on mobile vehicles to collect street data.

2.1.2 Electrical Design

Synchronization Circuit The synchronization circuit consists of master boards and slave boards. The synchronization framework is primarily based on reverse-engineering results, hacking HERO 3D sync cable, investigating schematics and protocols. In this section we will describe the schematic and leave the protocol in the next software section. For the master board we use a AVR 8-bit microcontroller to control power, mode, trigger and sync. A single master board can control up to 8 cameras. The design features a stackable

pinout, which is capable to expand up to 16 cameras for the current version. An I2C I/O expansion board is used to capture status of cameras, aka power status and slave ready status.
For the slave board, we exploit the design of 3D HERO sync cable (L side). The slave board uses HERO Bus to communicate with GoPro cameras. It is a 30-pin connector, same as the traditional iPod connector. There is an on-board I2C EEPROM programed to have value 0x05 at the first address, indicating the cameras to switch to 3D slave mode. The MOSFET connected to pin 24 controls the power/mode, functioning the same as the power/mode button on the camera. The slave board is powered at 3.3V, same as the GoPro camera.

Data Transfer and Charging To minimize overhead in data transfer and charging, we connect the GoPro cameras with mini-USB cables and multiplex them into two 7-port USB 3.0 hubs, which have better bandwidth for data transfer and power rating for charging than USB 2.0. Thus there is no need to remove batteries or SD vehicleds for charging or transferring data from time to time.

Assistant Sensors and Communication In addition to cameras, we attach IMU, GPS and LIDAR sensors to the camera array for potential sensor fusion applications such as vision-based SLAM, object recognition and 3D reconstruction. An on-board arm-based computer running Ubuntu is integrated in the system as high level management and sensor data logging. Wifi and Xbee communication are enabled to remotely access the camera array, which is necessary for remote platform such as UAV/helicopter.

External Synchronization The synchronization mechanism of GoPro uses pulses to acknowledge frames. To avoid dropping frames for long videos, an external flashing device is set up to validate and adjust frame drop besides the hardware synchronization. We use high power LEDs triggered consecutively and periodically to validate the synchronization of frames and to observer whether either of the camera drop a frame.

2.1.3 Remote Control with New Array

Apart from the camera array consisted of GoPro HERO 2, we also build a new version of camera array from GoPro HERO 3 as Figure 2 shows. Since GoPro HERO 3 Black Edition has the particular Wi-Fi communication with a remote controller. We can directly use a remote controller to control all the ten cameras in the array after pairing the cameras with the controller. This new version camera array is more modular and user friendly and solves the problem of dead cameras during the data collection.

2.2 Software

The multi-camera array system is based on ROS(Robotics Operation System) to syncronize different sensors. Synchronization token, glasses, kinect, IMU, GPS and LIDAR data is transmitted to the system with timestamps in ROS. Wireless commands through xbee can also be sent to the system with the framework. The built-in logging functionality is efficient and accurate for post-processing and analysis.

The main synchronization protocol is generated by hacking the HERO 3D sync cable. We use a logic analyzer to capture the signal between the master and slave cameras. In brief, the protocol could be described as follows.

1. Master: Get ready to take video
2. Slave: Ready
3. Master: Take video
4. Master: Start a 50% duty cycle 15Hz clock (30fps) on ID3 pin as a frame clock
5. Master: Stop
6. Master: Back to normal states
7. Slave: ID1 pin back to high

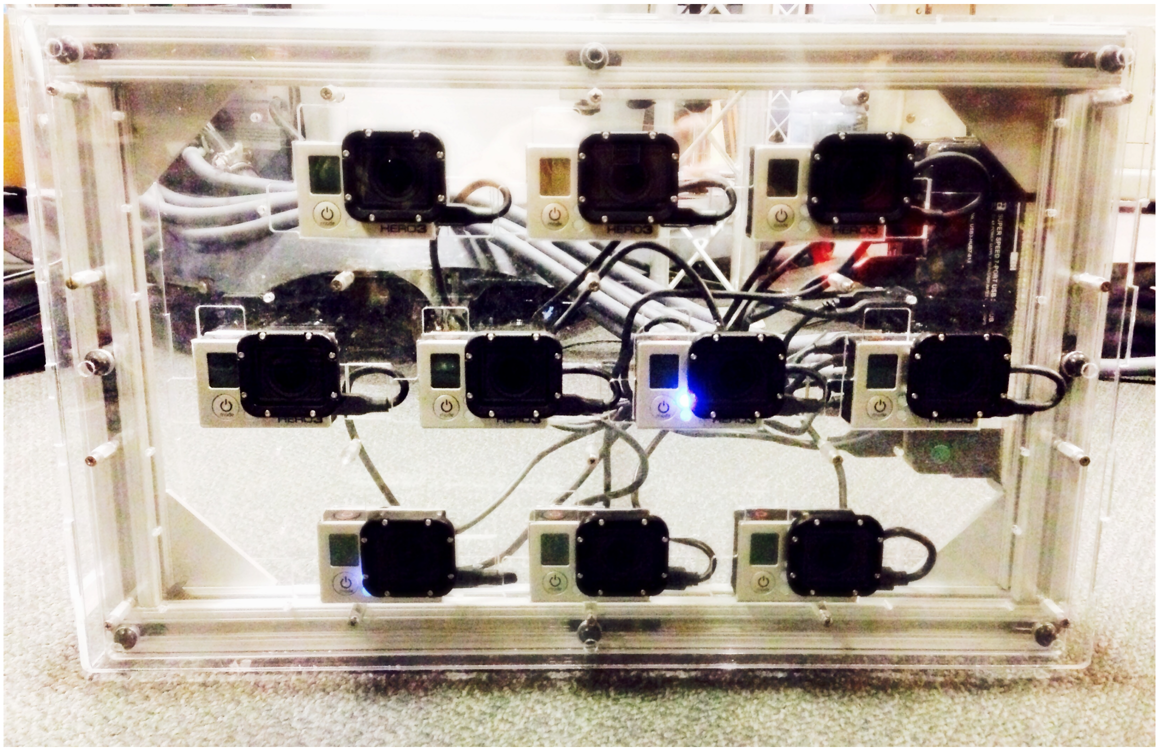

Figure 2: *New camera array with remote controller*

The frame clock is half of FPS as it uses both rising-edge and falling-edge of the square wave as frame reference.

We use ROS(Robotics Operation System)[3] running on Ubuntu as the major software framework of the system as shown Figure 3.

2.3 Data Collection Procedure

To evaluate the performance of the camera array framework, a dataset collection method was introduced. We established different platforms according to the interaction method of the driver with both the external and internal environment. The urban street dataset is good research material to analysis driver's intention and behavior. With the data of traffic, pedestrian, street, signs and buildings, as well as drivers reaction to the environment, the research could be meaningful to learn the correlation between behavior and traffic, and predict driver's intention thus contribute to improve transportation safety.

2.3.1 External Data Collection

Video sequence was captured from the camera array system mounted on a vehicle driving in the streets of Philadelphia, USA. Apart from the vision part, information of position, speed and even the street appearance were also collected from different sensors. The implementation of external data collection is shown in Figure 4.

2.3.2 Internal Data Collection

To record the driver's behavior, the driver weared one pair of FPV glasses which can take both the eye movement and the front perceptive of the driver, and Kinect was used to collect the full view of the environment inside the vehicle. The implementation of internal data collection is shown in Figure 5.

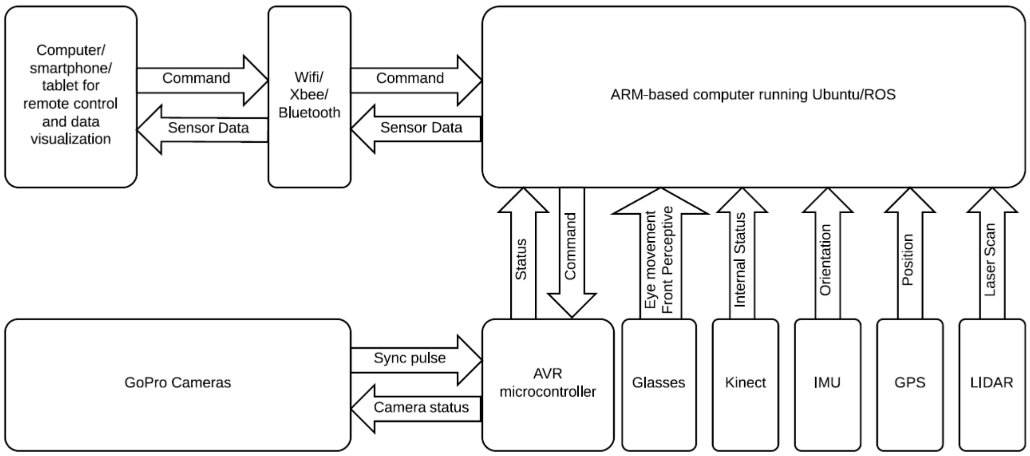

Figure 3: *ROS Framework*

Figure 4: Vehicle platform for external data collecction

Figure 5: Data collection inside the vehicle

Date	Location	Weather	Duration	FPS	Resolution	Platform
01/29/2013	Downtown, Philadelphia	Sunny	15min	30	1080 × 960	Vehicle
03/29/2013	Downtown, Philadelphia	Cloudy	30min	30	1080 × 960	Vehicle
05/24/2013	Downtown, Philadelphia	Cloudy	20min	30	1080 × 960	Vehicle
09/15/2013	University City, Philadelphia	Sunny	5min	15	3840 × 2160	Moving Tripod

Table 1: Dataset Documentation

3 Analysis of Data

3.1 Data Documentation

Table 1 shows the brief documentation of our datasets. Apart from the last one, all the others are taken by GoPro HERO 2 camera array. This table also includes the weather information since different lighting environment will affect the exposure time of cameras. Besides, the last dataset is one test dataset and we are still working on a full dataset for the new camera array.

3.2 GoPro Camera Array

3.2.1 Calibration

Accurate calibration of intrinsic parameters and extrinsic parameters is crucial for applications of the camera array. Our goal is to make a toolbox for multiple camera calibration with accurate, efficient and general approach. We hope to minimize assumptions of geometric configuration and overheads in manual operations.

We exploited the methodology of calibration in the Camera Calibration Toolbox for Matlab. Since the toolbox is a classic approach for mono and stereo calibration, it would be a reasonable idea to extend the toolbox for multi-camera calibration with bundle adjustment. There are several limitations in the original toolbox for our purpose. We address the limitations by modifying source code and introducing OpenCV to utilize automatic and efficient pipeline.

The whole pipeline of calibration and rectification is shown below. Stages processed in C++ is explicitly indicated, otherwise the stage is in Matlab.

1. Automatic corner detection (C++/OpenCV)
2. Mono camera calibration
3. Multiple camera calibration
4. Error analysis
5. Save parameters to YAML files
6. Rectification (C++/OpenCV)
7. Validation

Another limitation of the Camera Calibration Toolbox for Matlab is the lack of multi-camera support. We carefully modify the stereo calibration source code to extend the support to multi-camera system. The multi-camera calibration stage loads the result of mono calibration as initial state of bundle adjustment, which preserves stability and compatibility to the original toolbox.

The GoPro camera array is calibrated as shown in Figure 6. The reprojection error of the corners on chessboards is shown in Figure 7. The mean of the error is 2.8×10^{-10} pixel and the standard deviation is 0.1773 pixel. As GoPro is an wide angle camera, the pinhole model does not systematically fit the whole field of view, which generates rare bad points with a reprojection error around 1 pixel. These points are particularly located near the boundary of the images whose chessboard is very adjacent to the camera system. However our field of of interest is mainly the center part of the image, especially after undistortion and rectification. We accept the trade-off here to apply simple camera model and focus on the practical part of the results.

3.2.2 Rectification

With the result of calibration, we oriented the cameras into the same direction as the reference camera, and optionally shifted them at a given depth, which generates a 2D projective homography for each camera. An alternation is to shift the cameras to the primary plane derived by PCA and orient all cameras to the direction perpendicular to the primary plane. The rectification was done with OpenCV.

1. Set ideal pose based on reference camera or PCA
2. Calculate 2D projective homography based on calibration result and ideal pose
3. Undistort images based on distortion coefficients
4. Apply homography to images of each camera

A sample of rectification is shown in Figure 8. We manually select a point of interest and show the locations on different images to validate the rectification. Theoretically the points should form a similar arrangement of the camera array, as shown in Figure 9. The pixel-level error is mainly due to manual selection of the same point of interest across the dataset.

3.2.3 Disparity

A disparity algorithm was developed to exploit the redundant images of the camera system. We divided the camera system into left and right banks and calculated disparity respectively. Then we merged the results for occluded objects. The current disparity accuracy could reach 0.5 pixel. The disparity result can be used as reasonable accurate information for segmentation, recognition, super resolution and other applications. The disparity of street environment is shown as Figure 10.

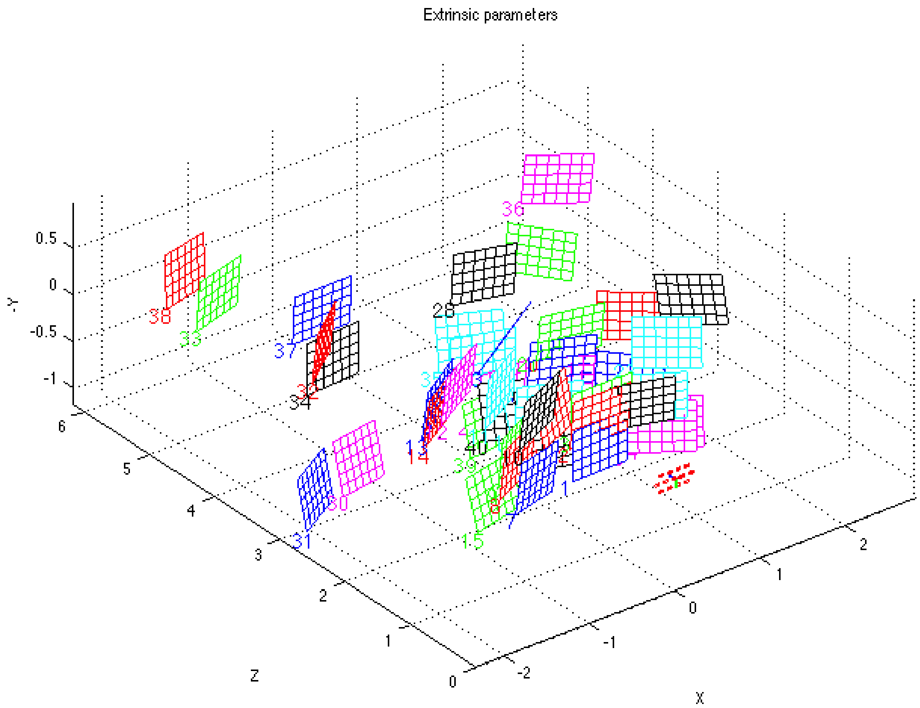

Figure 6: *Extrinsic parameters with reconstructed chessboards*

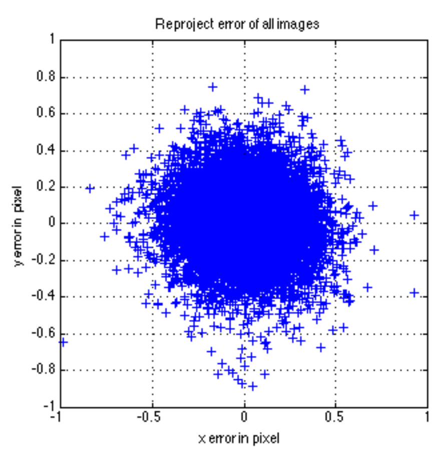

Figure 7: *Reprojection error of the corners on chessboard* $e_{mean} = 2.8 \times 10^{-10}$ *pixel*

Figure 8: *Rectified images*

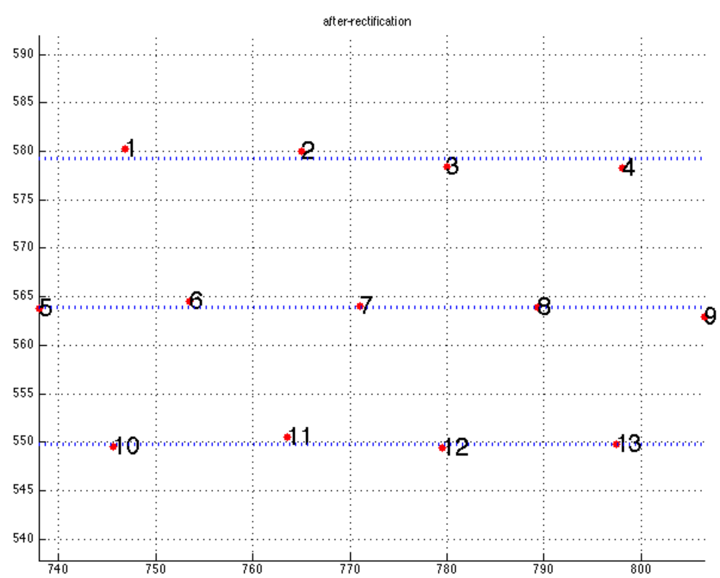

Figure 9: *The point of interest in each of the rectified images*

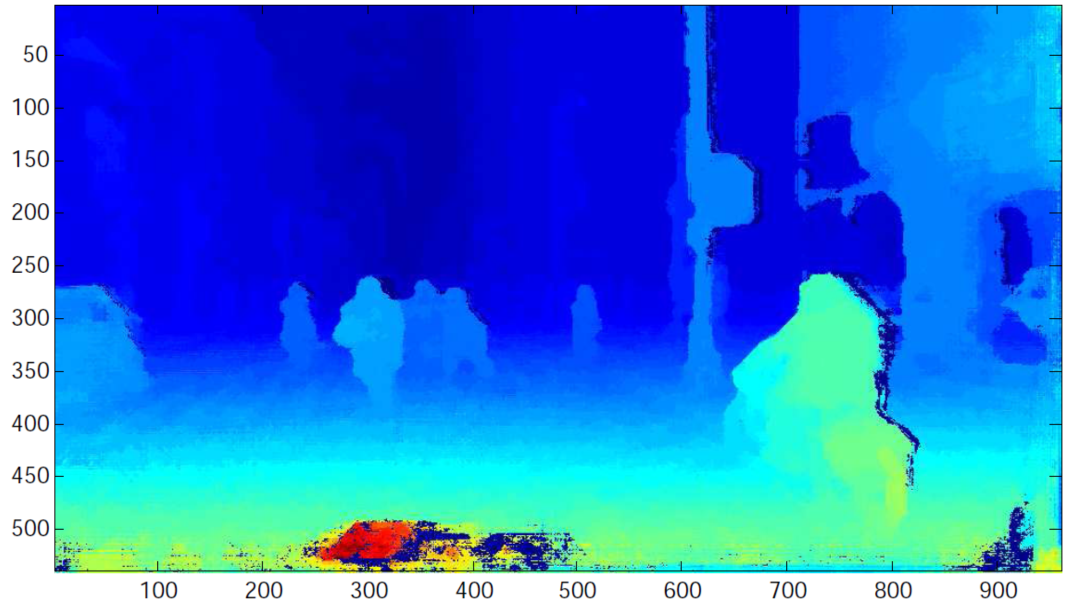

Figure 10: *Disparity of image*

3.3 FPV Glasses

There are two cameras in the FPV glasses: one is used to record the eye movement of the driver, defined as eye camera; the other is used to record the gaze point of the driver, defined as view camera. In this part, the main work of the FPV glasses detection optimization will be introduced.

3.3.1 Problem

In order to estimate the gaze point in visible spectrum, we need a reliable vision-based method for iris detection. This problem is quite challenging as human eyes are dynamic and the motion is subtle in a small range. The main challenges include occlusion introduced by the eyelids and eyelashes, presence of reflection and specularities introduced by arbitrary illumination, variability in the iris color between users, variability in the sclera texture due to changes in eye irrigation, and the fact that small changes in the camera location relative to the eye cause complex changes in the scene geometry.

3.3.2 Approach

Traditional methods for iris detection is ellipse detection on the image plane. The method proposed in this system relies on the observation that, although the iris appears as an ellipse on the eye image, it actually corresponds to a circle on the eye surface. Thus the method replaces the traditional task of fitting an ellipse on the image, by the task of fitting a circle directly on the eye surface. This approach greatly reduces the complexity of problem by reducing the number of parameters in the search space without diminishing the richness of the model. Because circle fitting requires less parameters, it is usually faster, more robust, and less prone to overfit the data, especially in degenerate conditions were only a portion of the iris is observed. Figure 11 shows the detection result of iris and gaze point.
The whole framework of the system consists of several parts, which include eye corners information obtain, 3D eye ball model establishment, iris detection, in and out camera calibration and gaze point output. Detailed information are shown as below:

1. Eye corners obtain: Ask the user to input the position of two eye corners by clicking them on the image and regard it as a diameter of the eye ball.

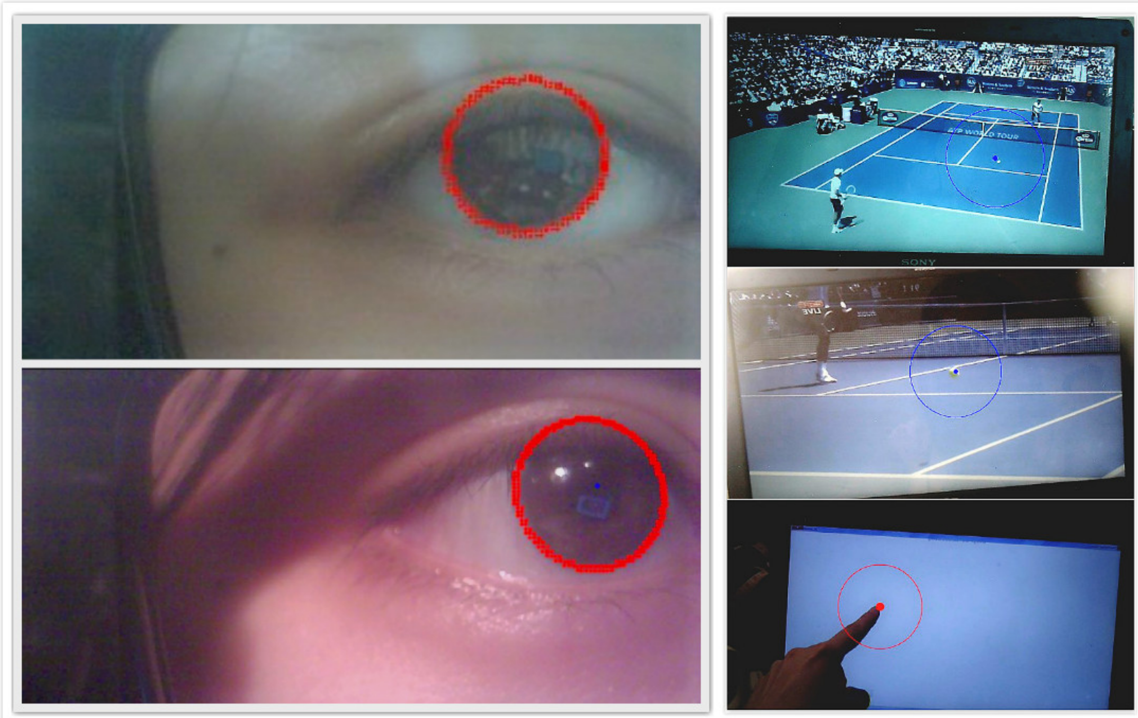

Figure 11: *Detection of Iris and Gaze Racking*

2. 3D eye model establishment: Calculate the 3D coordinate of points on the eye ball sphere and get their 2D projection coordinate on the image plane. Sample points by certain intervals and save the 2D points as the model of the eye ball sphere.

3. Iris detection: Calculate the gradients of the eye image, and use Hough transform to have a rough estimation of the position and the size of the iris center. Select features on the contour of the iris circle and refine the estimation by Taubin's method.

4. In and out camera calibration: The screen is divided into grids of , and the red ellipse target is shown randomly in one of the grid. Ask the user to click the center of the red target, and calculate the iris center position and the center of the target in the out image meanwhile. Use the information in the calibration to set up a correspondence between iris position and gaze estimation.

5. Gaze estimation output: Detect and calculate the position of iris center in real time, and give out the estimated gaze point by the correspondence between iris position and gaze estimation.

4 Future Work

We are now regularly collecting data with the GoPro camera array on vehicle. We will focus on relationship between the driver's intension and behavior. We will also analyze and develop driver intention algorithms with these data and test with different drivers in the following year.

The future work of the multi-camera calibration toolbox is to improve compatibility and stability of various types of camera array and add user-friendly interface. We are trying to figure out the optimal rectification method to move camera in small range for best disparity precision. We would like to publicly release the toolbox in near future.

5 Conclusion

Understanding driver behavior is a siginificant analysis in safe driving. In this reasearch, we want to develop a vision-based framework for driver behavior analysis. We have collected relevant human driver data from instrumented test vehicle. After deep analysis and development, we have obtained the calibration result for both the GoPro camera array and FPV glasses, the rectification information, disparity map and also the detected iris area and gaze point of the driver. Based on the data and the extended analysis, we also want to build driver intention algorithms to predict and estimate the next behavior of the driver, which are the main part of our future work.

References

[1] First person vision website from cmu.

[2] Gopro official website.

[3] Robot operating system.

[4] Brendan T. Morris Anup Doshi and Moham M. Trived. On-road prediction of driver's intent with multimodal sensory cues. *IEEE*, 2011.

[5] Vaibhav Vaish Eino-Ville (Eddy) Talvala Emilio Antunez Adam Barth Andrew Adams Marc Levoy Mark Horowitz Bennett Wilburn, Neel Joshi. High performance imaging using large camera arrays. *SIGGRAPH*, 2005.

[6] Mohan Manubhai Trivedi Cuong Tran, Anup Doshi. Modeling and prediction of driver behavior by foot gesture analysis. *Elsevier*, 2011.

[7] Marcus A. Brubaker · David J. Fleet · Aaron Hertzmann. Physics-based person tracking using the anthropomorphic walker. *International Journal of Computer Vision*, 2009.

[8] Kohsia S. Huang and Mohan M. Trivedi. 3d shape context based gesture analysis integrated with tracking using omni video array. *IEEE*, 2005.

[9] IEEE Shinko Y. Cheng, Member and Mohan M. Trivedi. Vision-based infotainment user determination by hand recognition for driver assistance. *IEEE*, 2010.

[10] Daniel Martinec Tomas Svoboda and et al. Multi-camera self-calibration.

[11] C.L. Zitnick Sing Bing Kang Marc Levoy Vaibhav Vaish, Richard Szeliski. Reconstructing occluded surfaces using synthetic apertures: Stereo, focus and robust measures. *CVPR*, 2006.