3D Perspective Effects on a Smart Phone

Jai Prakash
Carnegie Mellon University
Master of Science in Computer Vision
jprakash@andrew.cmu.edu

Jennifer Lake
Carnegie Mellon University
Master of Science in Computer Vision
jelake@andrew.cmu.edu

Abstract

In this project, we aim to create a 3D perspective transformation on a smart phone that will give the illusion of negative parallax. The user should feel that there is depth to the smart phone screen, rather than just a flat screen (citation). This is achieved by finding the three-dimensional vector between the users face and the center of the phone. This was achieved by tracking the users face using the front-facing camera and calculating the smart phones orientation using the Internal Measurement Unit (IMU) and using Kalman filtering to smooth out the measurements. Finally, these two pieces of information are combined to find the desired vector to create the perspective illusion.

1. Introduction

1.1. Motivation

As mobile phones have evolved over the last twenty years, many of the major milestones have been in display improvements. Mobile phones have gone from very small, simple displays to larger and more colorful displays. At the forefront of this evolution, there is a growing trend of three-dimensional displays. The primary motivation driving this trend in 3D perspective displays is to give the user more a more immersive user experience.

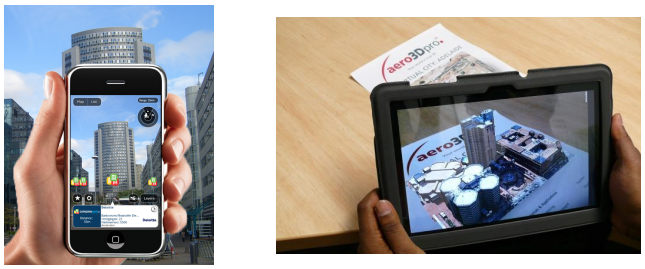

Figure 1: AR in present smartphones

In 2013, Apple introduced the parallax effect on their line of iPhones, which allows for a 3D-like feeling, just short of a full 3D perspective effect [1]. In 2015, the Amazon Fire Phone introduced a full 3D perspective effect, which was branded as ”Dynamic Perspective” [2]. It is this type of effect that we aim to create in this project however, unlike the Amazon Fire Phone, we will be implementing this effect on simpler, standard smart phone hardware

1.2. Background

1.3. Related Work

1.3.1. Amazon Fire Phone

-Jenna

1.3.2. Virtual Window

-Jenna

Figure 2: Direct AR from Meta-glasses

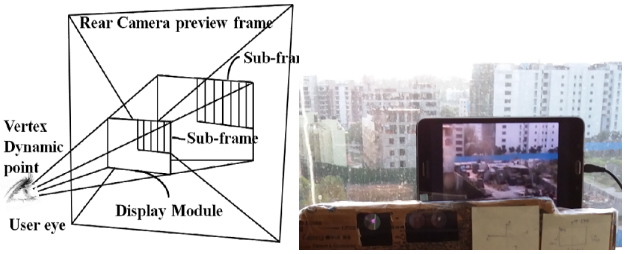

Figure 3: Digital transparency using Kinect on Android tablet

1.3.3. Digital Transparency

One more motivation for this project come from the augmented reality (AR) in present smartphones. The AR is smartphones can majorly be classified into two types

- **Indirect AR** Figure 1 shows augmented reality applications in current smartphones. This gives the user a video see-through experience, as the view is from the camera's perspective.

- **Direct AR** These are the systems that give a perspective from user's point of view and hence give more immersive experience than indirect AR system. Holo-lens and meta-glasses (Figure 2) are examples of direct AR systems.

The current direct AR systems are generally bulky and requires user to wear special glasses to use them. However, a direct AR system can be

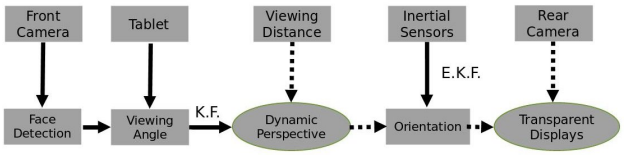

Figure 4: Block diagram of dynamic perspective system

achieved in smartphones and tablets by using digital transparency [3]. Digital transparency system is shown in Figure 3. This system tracks the eyes of the user and finds the angle which the tablet subtends at user's eye and crops the rear camera preview to the same angle to create a virtually transparent interface. This system gives a method to achieve direct AR on smartphones/tablets and hence gives a more immersive **glass-see-through** experience than conventional **video-see-through** experience.

This method uses a kinect sensor to track the accurate 3D position of the user with respect to tablet. The kinect sensor makes the system bulkier and difficult to use in smartphones. So, in this project we are addressing another method to achieve digital transparency using just the front facing camera and inertial sensors. A complete hands-free digital transparency system is out of scope of this project. The main concentration is on creating a dynamic perspective system which responds to the user's position with respect to the tablet.

2. Method

Creating a dynamic perspective system involves finding the viewing angle of the user and rendering the graphical object depending on position of the user. This might involve three scenario

- Tablet is static and user moves

- Tablet moves and user is static

- Both tablet and the user can move

To find the motion and orientation of the phone, on-board inertial sensors can be deployed to find the orientation of the phone. A study on sensor fusion has been made to find the orientation of the phone.

The block diagram of the system is shown in the figure 4. Solid arrow shows the algorithms implemented for this project. The dotted arrow parts are yet to be figured out to build a full fledge digital transparency system. However, the main focus is only on creating the dynamic perspective.

To create a dynamic perspective, we need to first track the position of user with respect to the tablet. In this project we assume that the distance of the eye from tablet is fixed (approximately 45-50 cm). This assumptions fails to address the scale factor in dynamic perspective. Hence, we don't get zoom-in and zoom-out effect as we move forward and backward. Since the distance from the tablet is assumed to be constant, we use the viewing angle to render the user interface (UI) in 3D. The following sections describe the exact methods used in each step.

2.1. 3D Geometry of Problem

-Jenna. describe the vectors involved

2.2. Face Detection

-Jenna, I'm assuming this is Viola-Jones

2.3. Kalman Filter

The face detection using the cascade classifier is subject to jitter due to noise in the scene. The face detection sometimes fails because of various reasons like change is lighting conditions, change in expression of the face etc. This detection might cause jitter in the rendering. In order to avoid the jitter, we need to smooth the face detection and handle the no-detection case. The Kalman filter [5], can handle statistical uncertainties and other accuracies and produce results that are more accurate than a single measurement.

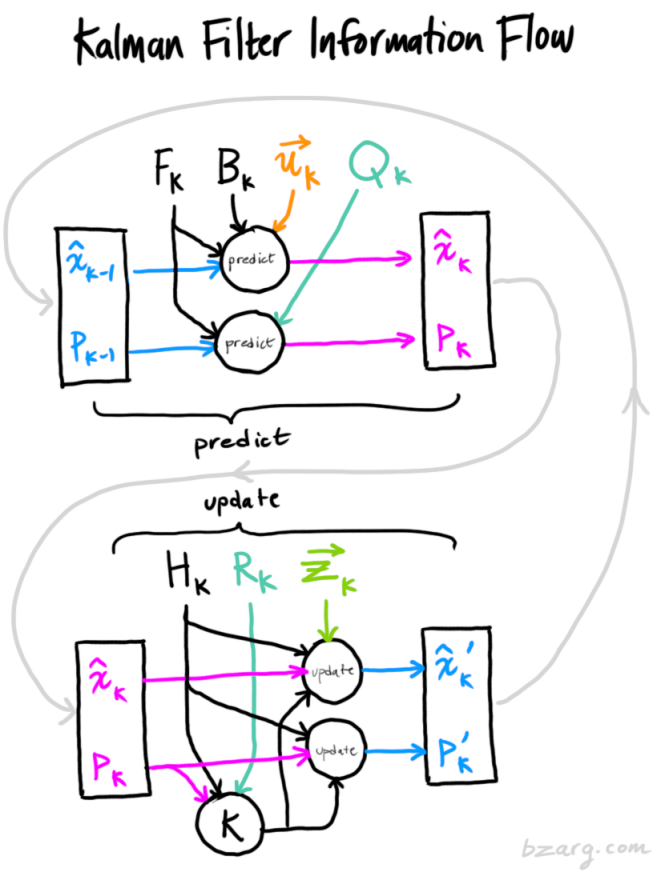

Figure 5: Diagram showing the update in Kalman filter
credits: http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

The Kalman filter models the uncertainties in the states of position and velocities. The model alone is sufficient to predict the future uncertainties without remembering the past readings. which means that we can predict the state by knowing the parameters of the current state. Once the new reading arrive, the parameters of the model is updated.

The prediction step is given by the equation

$$\hat{x_k} = \mathbf{F_k}\hat{x}_{k-1} + \mathbf{B_k}\vec{u_k} \quad (1)$$

$$\mathbf{P_k} = \mathbf{F_k}\mathbf{P_{k-1}}\mathbf{F_k}^T + \mathbf{Q_k} \quad (2)$$

Equation 1 is called the state estimate (a priori) equation. And equation 2 is called the estimate (a

priori) covariance. The significance of each variable is as follows:

- $\mathbf{F_k}$ is the state transition model which is applied to the previous state $\hat{x}_{k1}$
- $\mathbf{B_k}$ is the control-input model which is applied to the control vector $\vec{u}_k$
- $\mathbf{P_k}$ error covariance matrix (a measure of the estimated accuracy of the state estimate)
- $\mathbf{Q_k}$ the covariance of the process noise

Moving on to the update step, once we get the readings from the actual scenario, we need to update the model to update our belief about the scene. The update equations are given as

$$\mathbf{K'} = \mathbf{P_k}\mathbf{H_k}^T \left(\mathbf{H_k}\mathbf{P_k}\mathbf{H_k}^T + \mathbf{R_k}\right)^{-1} \quad (3)$$

$$\hat{x'}_k = \hat{x}_k + \mathbf{K'}\left(\vec{z}_k - \mathbf{H_k}\hat{x}_k\right) \quad (4)$$

$$\mathbf{P'_k} = \mathbf{P_k} - \mathbf{K'}\mathbf{H_k}\mathbf{P_k} \quad (5)$$

where,

$$\vec{z}_k = \mathbf{H_k}\hat{x}_k + \vec{v}_k$$
$$\vec{v}_k \approx \mathcal{N}(0, \mathbf{R_k})$$

The equation 3 is called the Kalman gain of the filter. The equation 4 is the updated (a posteriori) state estimation of the filter. And the equation 5 is known as the updated (a posteriori) state covariance equation. The variables used in these equations are described below:

- $\mathbf{H_k}$ is the observation model which maps the true state space into the observed space
- $\vec{v}_k$ is the observation noise
- $\mathbf{R_k}$ is the covariance of observation noise which is assumed to be zero mean Gaussian white noise

After applying Kalman filter, the face detection becomes continuous and smooth to be used for dynamic perspective.

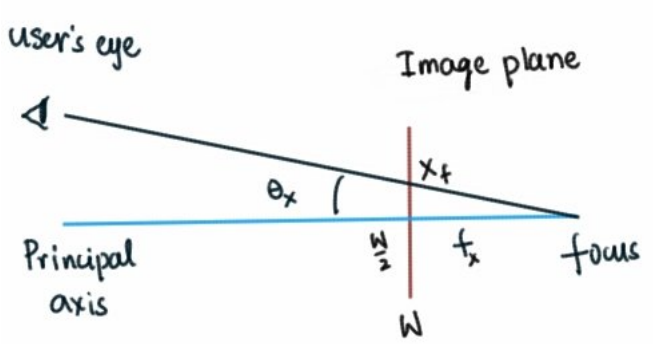

Figure 6: The viewing angle of the user with respect to front camera on tablet

2.4. Finding viewing angle of user

The viewing angle is one of the critical measurements for creating a dynamic perspective system. After applying the Kalman filter to smoothen the face detection, we need to find the viewing angle with respect to the camera. Since all the coordinates are in image frame of reference, the angle can be found by using the following equations.

$$\theta_x = tan^{-1}\left(\frac{x_f - \frac{W}{2}}{f_x}\right) \quad (6)$$

$$\theta_y = tan^{-1}\left(\frac{y_f - \frac{H}{2}}{f_y}\right) \quad (7)$$

In equation 6 and 7, (x_f, y_f) is the location of the centroid of the user's face. Image size is $W \times H$. f_x and f_y are the focal length of the camera in pixels.

Figure 6 shows the viewing angle of the user. The view angle is the angle subtended by the user's position with the principal axis of the camera. The focal length of the camera is found by using camera calibration in opencv [4].

2.5. Inertial Measurement Unit

-Jenna

2.6. Extended Kalman Filter

-Jenna

3. Experiments

3.1. Camera calibration

The camera calibration matrix is returned as

$$K = \begin{bmatrix} f_x & \alpha_x & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 145 & 0 & 930 \\ 0 & 145 & 601 \\ 0 & 0 & 1 \end{bmatrix} \quad (8)$$

The value of the focal length is in terms of the pixels. The resolution of the camea is 1920x1280, and the principal axis seems to be almost at the center of the image.

3.2. Kalman filter parameters

The Kalman filter parameters are chosen as follows $\mathbf{F_k} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, $\mathbf{H_k} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$

$$\mathbf{Q_k} = \begin{bmatrix} 0.0001 & 0 & 0 & 0 \\ 0 & 0.0001 & 0 & 0 \\ 0 & 0 & 0.0001 & 0 \\ 0 & 0 & 0 & 0.0001 \end{bmatrix}$$

$$\mathbf{R_k} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

The initial estimates are given as

$\hat{x}_0 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $\mathbf{P_{k0}} = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}$

$\mathbf{B_k}$ is not used, so its value is $\vec{0}$. For the update equation, the new detected point is used as position vector. In case of failure, $(\frac{W}{2}, \frac{H}{2})$ is used to update Kalman parameters.

3.3. IMU Experiments

-Jenna, make subsubsections as needed

4. Conclusions

-Jai

References

[1] Yarow, Jay. "Here's Why Apple Made That Motion-Effect For The Background Of The New IPhone Software." Business Insider. Business Insider, Inc, 25 Sept. 2013. Web. 11 Dec. 2015.

[2] Pelegrin, Williams. "Why Isnt the Fire Phone Truly 3D? Amazons Dynamic Perspective Tech Explained." *Digital Trends*. Digital Trends, 18 June 2014. Web. 11 Dec. 2015.

[3] J. Prakash and A. Vijayvargiya. A method for obtaining digital transparency in Electronic Devices, Indian Patent 4506/CHE/2014

[4] `http://docs.opencv.org/3.0-beta/doc/tutorials/calib3d/camera_calibration/camera_calibration.html`

[5] Kalman, R. E. "A New Approach to Linear Filtering and Prediction Problems". *Journal of Basic Engineering, 1960*