Augmented Reality in Autonomous Vehicle Testing
Project Report

Ankita Christine Victor
IMT2014005

1 Problem Statement & Motivation

Autonomous vehicles - vehicles that navigate without human input and control - are arriving. Google's self-driving car - Waymo - is undergoing live testing in parts of Phoenix, Arizona, with nobody in the driver's seat.[1] Autonomous vehicles have become a very plausible reality. The technology that controls a vehicle's navigation decisions are sensor data, video camera input that tracks obstacles and other traffic, and a central computer that analyses all of the data to manipulate steering, acceleration and breaking.[2] Object recognition on the camera's video input identifies pedestrians, other vehicles and obstacles.

A lot of test runs take place on streets with back-up drivers behind the wheel. To test the object detection ability of the autonomous vehicle, testers could use pre-recorded video of streets. Introducing a 3D model into these pre-recorded clips can help customize what obstacles are present in the scene, where they are, how fast they are moving, etcetera. The problem therefore is how to introduce a 3D model into 2D video frames, that can ultimately be used as simulation of real life scenarios to test autonomous vehicles.

2 Original Scope

The scope of this project was to develop a module that could take in a video (as 2D frames), obtain some depth information and identify what objects in the scene must occlude the new 3D model with manual selection, introduce a 3D model into the scene of every frame with occlusion, give some path/animation to the 3D model and render everything together as a new video. Figure 1 summarizes the required pipeline.

[1]https://www.wired.com/story/waymo-google-arizona-phoenix-driverless-self-driving-cars/
[2]http://www.telegraph.co.uk/cars/features/how-do-driverless-cars-work/

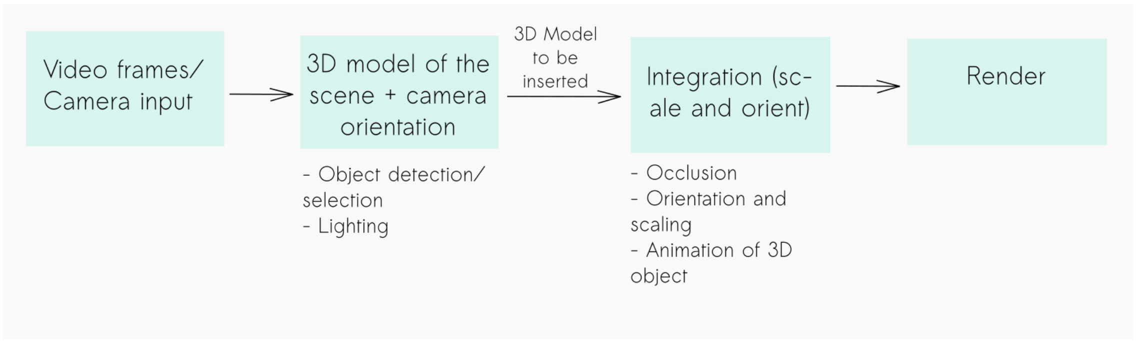

Video frames/ Camera input
3D model of the scene + camera orientation
3D Model to be inserted
Integration (scale and orient)
Render
- Object detection/ selection
- Lighting
- Occlusion
- Orientation and scaling
- Animation of 3D object

Figure 1: The Pipeline

3 Design

The code is written in an MVC structure with an underlying scenegraph.

3.1 Video Input and Depth Reconstruction

The video input is assumed to be as from a car's camera moving forward. The input is broken down into frames. Each frame is a 2D image with no implicit depth information. A 3D reconstruction of the scene has not been considered, since no object recognition techniques were used. To account for occlusion of the model to be inserted by an object in the frame, manual selection of billboards and forward projection of the billboards has been used. The following steps summarize how a billboard is picked and given depth. The user

(i) enters a billboard select mode

(ii) selects the boundary points of a particular object in the current frame using the right mouse click

(iii) enters a value for position along the Z axis - depth

(iv) toggles across frames using the arrow keys.

The picked points are used obtain both the texture extents of the polygon shaped billboard as well as its position on the frame at a fixed Z. The position of the billboard at any new depth is computed using similarity of triangles. For a picked point (x, y) (see Figure 2) at $Z = 7$ the corresponding point at $Z = d$ is found as

$$x' = \frac{x \times |-6-d|}{13}$$

$$y' = \frac{y \times |-6-d|}{13}$$

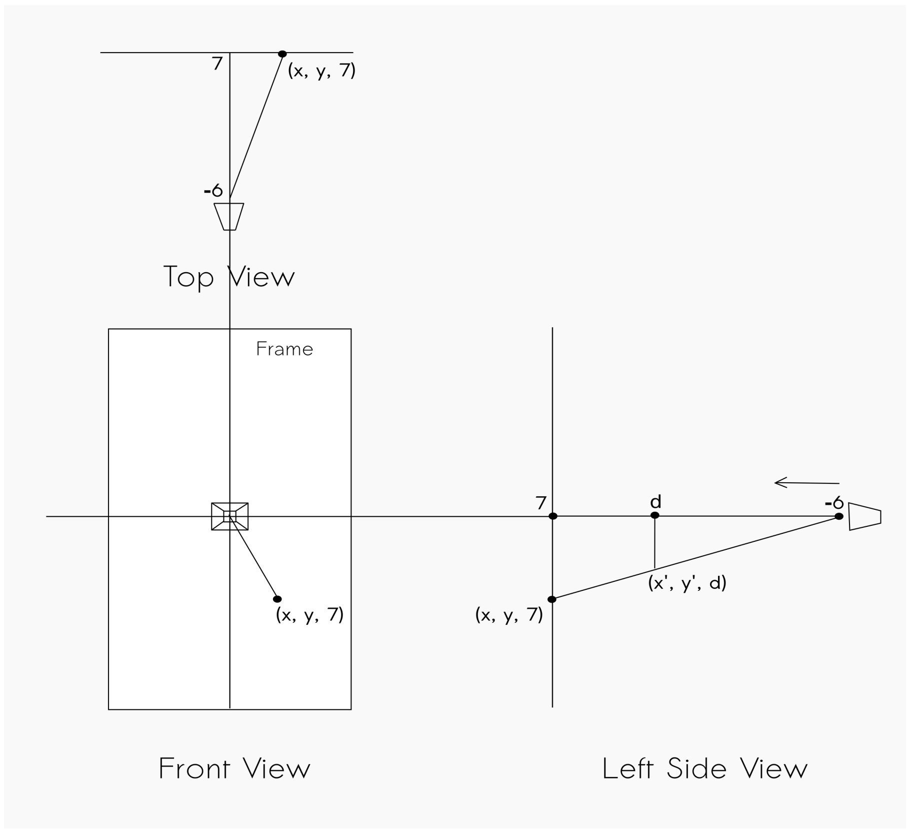
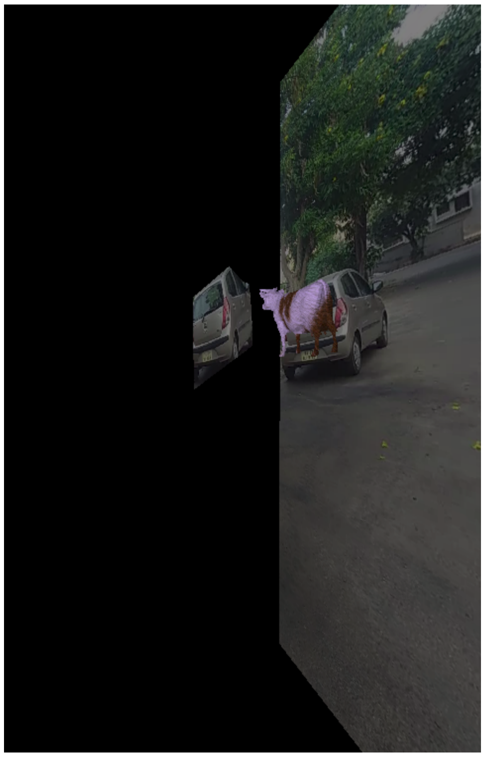


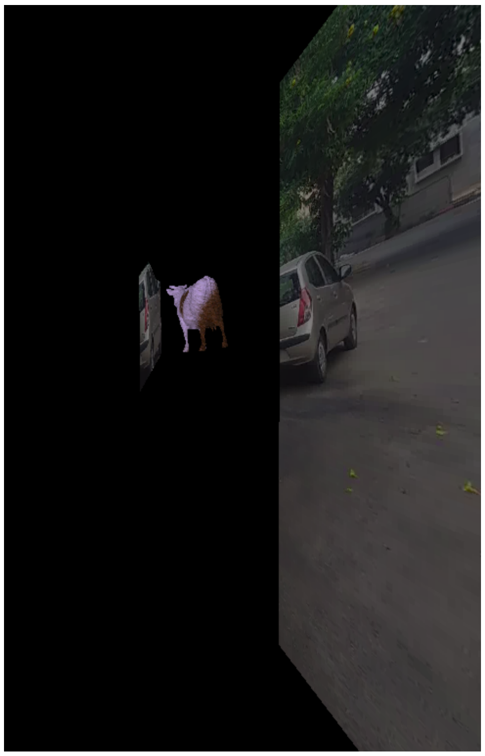
Figure 2: The Pipeline

No image recognition technique has been used to identify the current frame's picked object - a billboard - across the remaining frames. To avoid selection of the same object across all frames, the billboard can be picked across every n^{th} frame, and the position of the billboard in intermediate frames as well as depth is calculated using linear interpolation. The n value does not need to be consistent across consecutive pairs of frames. One could pick on the 1st frame, go to the 5th, then the 12th and so on. The code requires that the first and last frame have picked billboards. The interpolation assumes however that the same number of points are picked for every billboard.

Depth is taken as a user input to allow for billboards to be positioned at different depths as well as for the same billboard to be positioned at different depths across every frame. This allows billboards to come closer to the camera as they are in the actual video that's moving into the scene if the user provides depth as such. Again position along the Z axis is interpolated where required.



(a) Billboard at time T



(b) Billboard at time $T + \Delta T$

Figure 3: Changing Depth of Billboards

3.2 Lighting

Modelling light as it is in the scene has not been attempted. The inserted model alone is lit using ambient light, by enabling and disabling the light source just before and after the model is rendered. The prevents lighting up of the frame itself.

3.3 3D Model

The initial scale of the model needs to be set as there is no object comparison. The model moves in a simple horizontal line moving towards the camera eye to make it appear as if the car is moving towards it. The object is positioned on the Z axis relative to the billboard. Assuming that only billboards that must occlude the object are being selected, the object is positioned behind the billboard in every frame (see Figure 3). Only straight line paths have been experimented with.

3.4 High Level Design

The code is structured using MVC. The view used here is written to be replaceable with minimal code changes. The view simply calls the controller which in turn either calls a model function or executes one of its own.

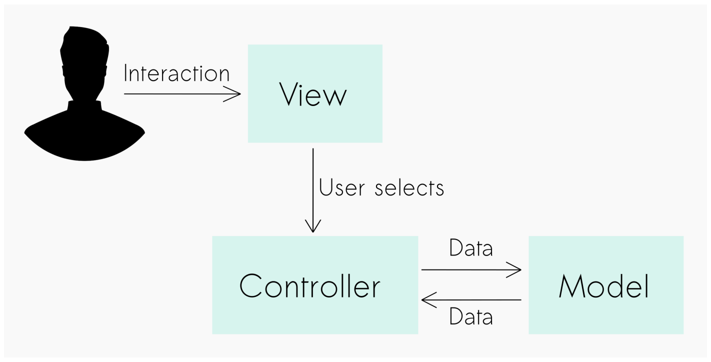


Figure 4: Control Flow

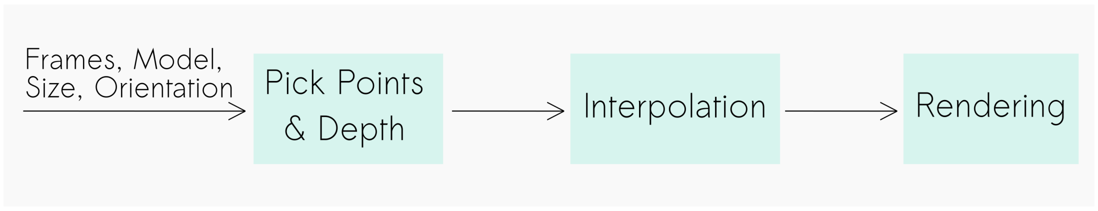


Figure 5: Program Flow

4 Tools

The code was entirely written in C++ using OpenGL, SOIL to read the input frames and Free GLUT as the windowing interface.

5 Usage

(i) The program by default enters a frame selection mode

(ii) Click 'm' to begin picking the boundary points of billboard on the current frame and 'm' to finish

(iii) Use the right mouse click to pick points and left to rotate the camera

(iv) While on the same frame click ‘i’ to enter depth (negative values are closer to the camera)

(v) Use the arrow keys to change frames

(vi) Click ‘c’ to change the camera to top view

(vii) Click ‘s’ to start/restart the simulation

6 Extent of Completion

The program accepts video frames as input. The first part of the pipeline - extracting objects from the scene has been completed using billboard picking. Occlusion has been accounted for by projecting the billboard forward. The perception of objects moving closer has been handled by giving each billboard a different depth moving closer to the camera. The initial height and size of the virtual object needs specification. Only straight line paths of motion intended to visualise the occlusion effect have been experimented with.

7 Challenges

The major challenges were occlusion and depth perception. The video input is a flat 2D image while the introduced object, to look realistic, needs to be occluded at the right places. Questions like where the virtual object must be placed, how it’s size should change with moving into the scene, and the height at which it must be placed, needed to be answered. Using no image recognition mechanisms or libraries makes the latter significantly harder. The code here depends largely on user intervention and cues.

8 Open Issues & Future Work

Lighting is an issue that hasn’t really been addressed. The animation of the virtual object - introducing more complex paths and adjusting positions based on objects in the scene is something that should be worked on. For demonstration purposes the path of the 3D model (here a cow) has been hard coded, but depth is taken based on user input. There is provision for a path to be picked using points, but again this is very rudimentary. Introducing image recognition into the mix is the next step to take. This could eliminate the need to pick the same billboard across multiple billboards, manually provide the 3D model’s size and height above the ground.