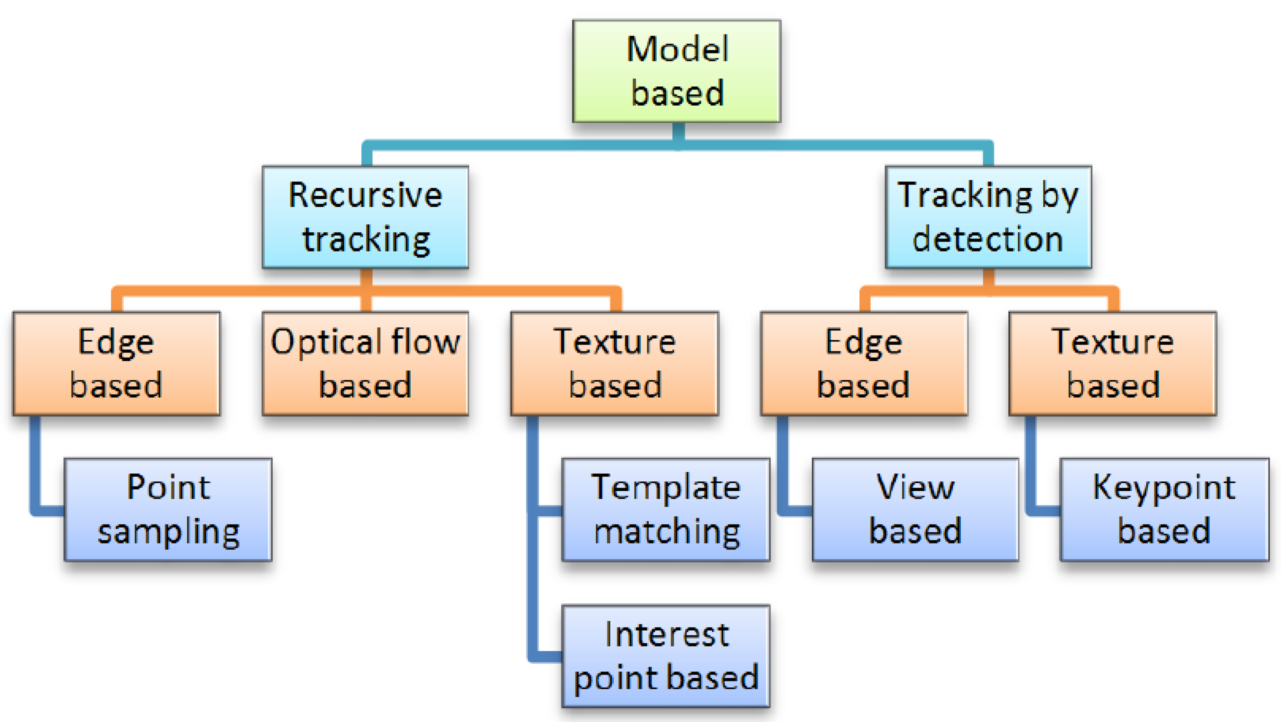
Model Based Tracking for Augmented Reality on Mobile Devices

Abhilash Sunder Raj, Michael Lowney
Department of Electrical Engineering, Stanford University

Motivation

- Real-time camera pose estimation and tracking is a key part of many Augmented Reality applications.
- Marker based tracking relies on a printed out pattern to be placed in the scene
- Model based tracking allows tracking to be performed using (natural) objects in scene whose 3D CAD models are known.

Related Work

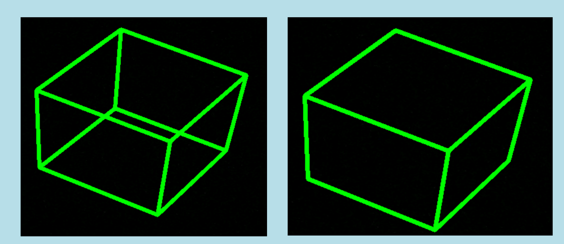


Model based tracking methods can be broadly classified into two main categories. Recursive tracking uses the previous pose as an estimate to calculate current pose, but is generally very fast. Tracking by detection can calculate pose without prior knowledge but requires a large amount of processing time[1]. In our work, we implemented recursive edge based tracking

[1] Lima, João Paulo, et al. "Model based markerless 3D tracking applied to augmented reality." *Journal on 3D Interactive Systems* 1 (2010).

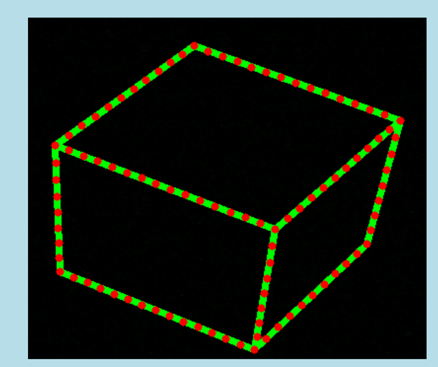
Pipeline

Visible Edge detection

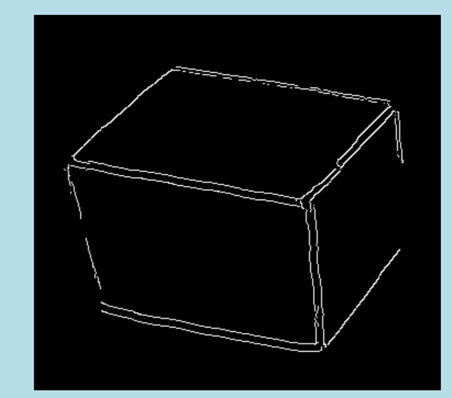


Before After

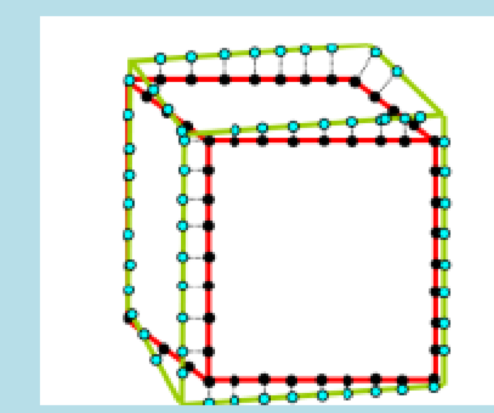
Control Points Sampling



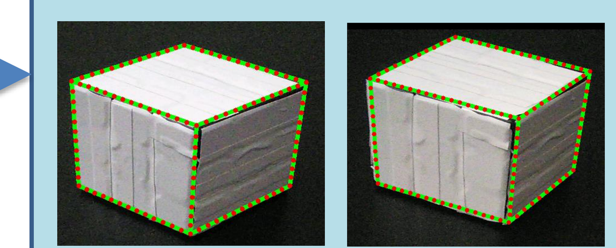
Canny Edge Detection



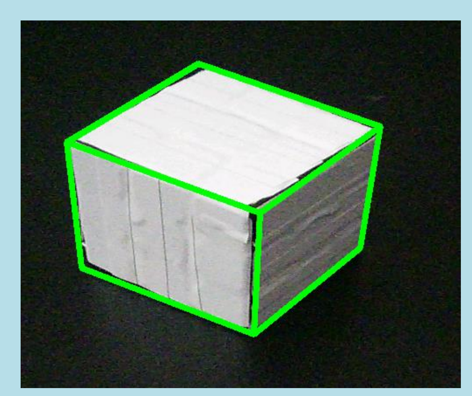
Correspondence Search



Pose Estimation



Pose initialization

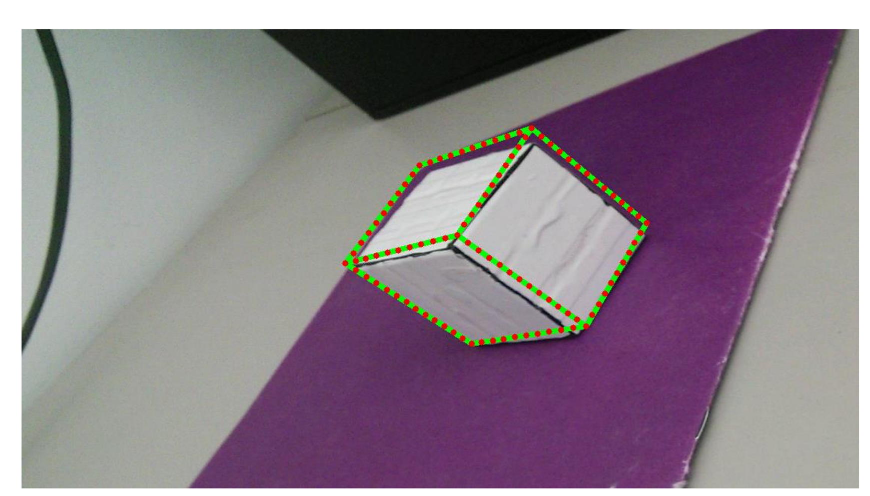


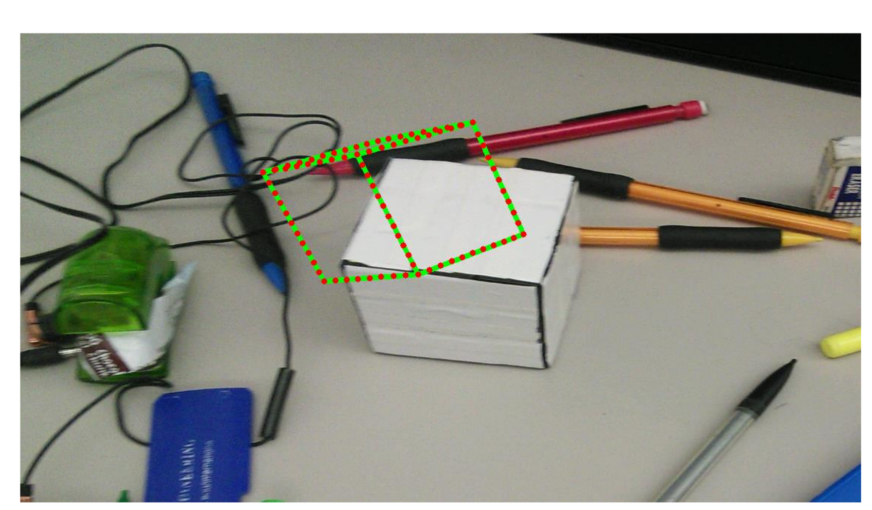
- The model is projected onto the image plane and a wire frame image is displayed on the screen so that the user can align it with the object.
- The visible edges are computed by calculating the dot product between the camera vector and the surface normal vectors.

- Control points on the 3D model are sampled at a rate that is defined by the length of each edge in the image.
- Correspondences in the next frame are estimated by finding the nearest gradient in the orthogonal directions

- The change in pose between frames is calculated by minimizing the projection error between the control points and their correspondences.

Experimental Results





- The algorithm capable of tracking the model. However, it fails in cluttered scenes and when fast camera motion is applied.
- The system was implemented on a Nexus 7 tablet, and ran at 3-5 fps. This is a 43% reduction from the initial camera frame rate of 16fps.
- The jitter error in rotation and translation was measured by keeping both the camera and object stationary, and measuring the average mean squared change over 100 frames.

Jitter Error

Coordinate	MSE
ω_1	3.97×10^{-4} (radians)
ω_2	4.34×10^{-5} (radians)
ω_3	3.13×10^{-5} (radians)
X	0.6006 (mm)
Y	0.2787 (mm)
Z	10.449 (mm)