Optimizing 3D models from 2D images

T. Kostelijk
mailtjerk@gmail.com

December 16, 2010

Abstract

Here comes the abstract

1 Introduction

2 Skyline detection

This section describes how the skyline of an image is detected. A skyline separates a building from the sky and is used as an indicator of the building contour. Some previous work on skyline detection is done.
[9] yields a good introduction of different skyline detection techniques. In [1], a column based approach is used. This method is used in this project.
The skyline detector works as follows. First the image is preprocessed for the skyline detector algorithm. The contrast of the image is maximized. Then the image undertakes a Gaussian blur. On this image a *sobel* edge detector with a manual threshold is applied. The result is feed to the skyline detector.
The Skyline Detector uses an assumption: the first sharp edge (seen from top to bottom) is always a skyline/building edge. The algorithm works as follows: The value of a pixel in a column is evaluated. Iterating from top to bottom, the system takes the first sharp edge (a pixel that is above a certain threshold) and classifies this as a skyline pixel. This is done for every column in the edge image. The result is a set of coordinates of length W, where W is the width of the image, that represent the skyline.
The Skyline detector has an accuracy of (ABOUT) 90 % (TODO CHECK THIS PERCENTAGE) In some cases outliers are detected, these are often a streetlight or a tree. The outliers are filtered in the next step.

3 Skyline in 3D as a contour of the building

The 3D building contour is used to update the sparse 3D model of the building. This section describes how the 2D images are used to get the 3D contour of the building. This is done in several distinct steps.

3.1 Project to 3D space

Every 2D pixel of an input image presents a 3D point in space. No information is known about the distance from the 3D point to the camera. Therefore the algorithm starts with an infinite line of possible points in space. This line is spanned by two known coordinates:

- The camera center (camera centers are annotated for every image)
- $K'p$, where K is the Calibration matrix of the camera and p is the homogeneous pixel coordinate.

3.2 Intersect with building

A top-view image (source:Google maps) is used to create a rough indication of the building. The height of the building is estimated.
To know which part of the building the pixel presents, the building is divided into different walls. Every wall of the building spans a plane. Intersections are calculated between the line from the previous section and these planes.

The intersections are used to determine the wall that is has the largest probability of being responsible for the pixel. If a wall represents that pixel, the intersection (projected pixel) must lie either
1) On the wall (inside the polygonal representation of the wall)
or
2) On a small distance from the wall, this is treated as an inlier because the 3D model is sparse and the height of the building is unknown.
The first is calculated using an in-polygon algorithm. The later is calculated as follows:
First the distances from the intersection to four wall sides are calculated. For every wall the minimum distance is stored. The wall with the smallest distance is the one that most likely presents the pixel. If there are two (or even more) walls that succeeded the in-polygon test then the nearest wall (closest to the camera center) is selected. How the distance measure is calculated can be read in the next section.

3.3 Calculate the intersection point - wall distance

A wall consists of four corner points. Some corner-point pairs connect a line segment which represents a side of a wall. There are four line segments.
The intersection point (ISP) is projected orthogonally on all four lines that are spanned by the line segments. If the projected ISP lies between the two corner points (on the line segment) then the distance from the ISP to the projected ISP is returned. If it lies outside the two corner points the Euclidean distance to the closest corner-point is returned.

[I can explain how I did some clever dot product tricks but I think that is to much detail right?]

4 Update 3D model

5 References

- [1] Castano, Automatic detection of dust devils and clouds on Mars.
- [9] Cozman, Outdoor visual position estimation for planetary rovers.