Monocular Visual Odometry

Open Source Implementation
by

Rainer Hessmer, PhD
August 2010

Agenda

- Odometry
- Visual Odometry
- Implementation and Demo

Odometry

- Use sensors to estimate change in pose (location and orientation) over time
- Examples
  - Quadrature encoders on motors
  - GPS and compass
  - IMU (Inertial Measurement Unit)
    http://www.seattlerobotics.org/encoder/200610/article3/IMU%20Odometry,%20by%20David%20Anderson.htm
  - Visual

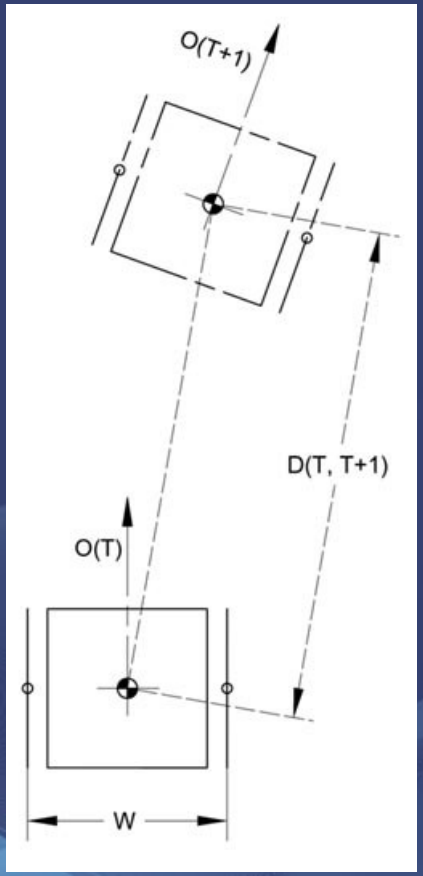


Image source: http://www.simreal.com/content/Odometry

Visual Odometry

- Stereo Vision
  - Two Years of Visual Odometry on the Mars Exploration Rovers
    http://www-robotics.jpl.nasa.gov/publications/Mark_Maimone/rob-06-0081.R4.pdf
- High Framerate Monocular Vision
  - Optical Mouse
  - Outdoor Downward-facing Optical Flow Odometry with Commodity Sensors
    http://www.ri.cmu.edu/pub_files/2009/7/DilleFSR09.pdf
- What about using one regular Web cam?

See paper:

A Robust Visual Odometry and Precipice Detection System Using Consumer-grade Monocular Vision

Jason Campbell, Rahul Sukthankar, Illah Nourbakhsh, Aroon Pahwa

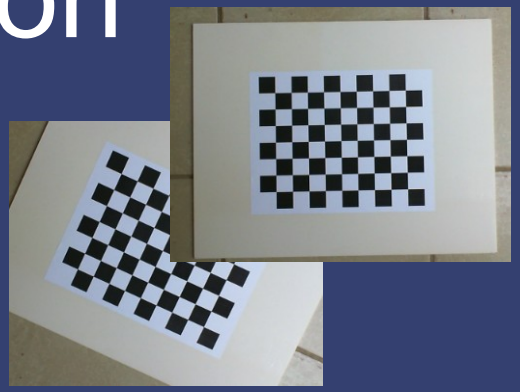
http://www.cs.cmu.edu/~personalrover/PER/ResearchersPapers/CampbellSukthankarNourbakhshPahwa_VisualOdometryCR.pdf

Ingredients

- One calibrated Web cam (Microsoft Live Cam Cinema)

- One laptop
- OpenCV
- EMGU CV (C# wrapper for OpenCV)
- Lucas and Kanade pyramidal optical flow
- The essence of the algorithm presented in the paper
- Lots of custom C# code

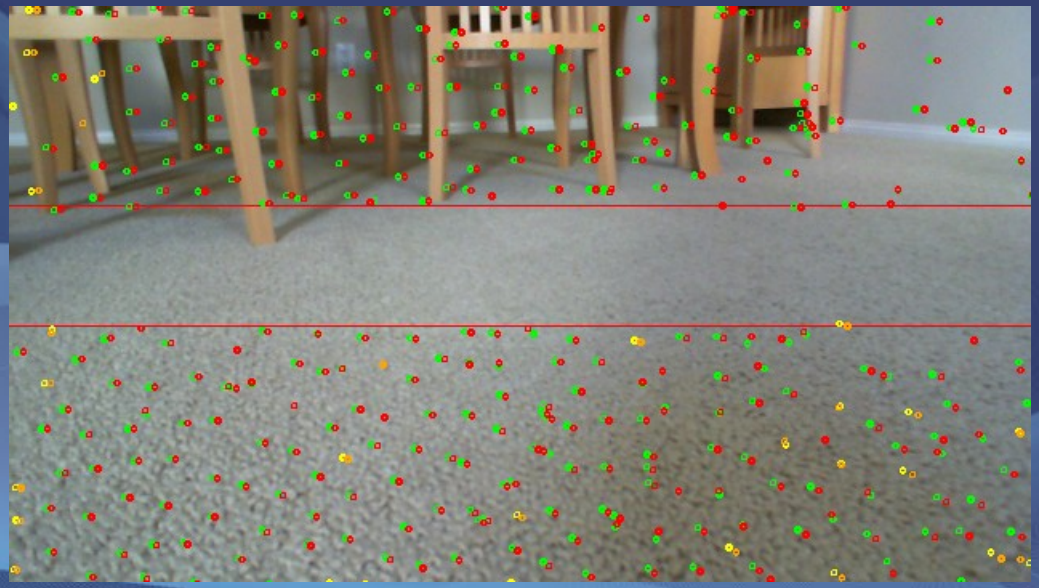
Camera Calibration



- Install OpenCV
- Print and mount a decent size chessboard like grid
- Turn off auto focus and take pictures of the chessboard in various orientations
- Run C:\OpenCV2.1\samples\c\Calibration.exe with the appropriate command line parameters against the pictures to determine the camera parameters

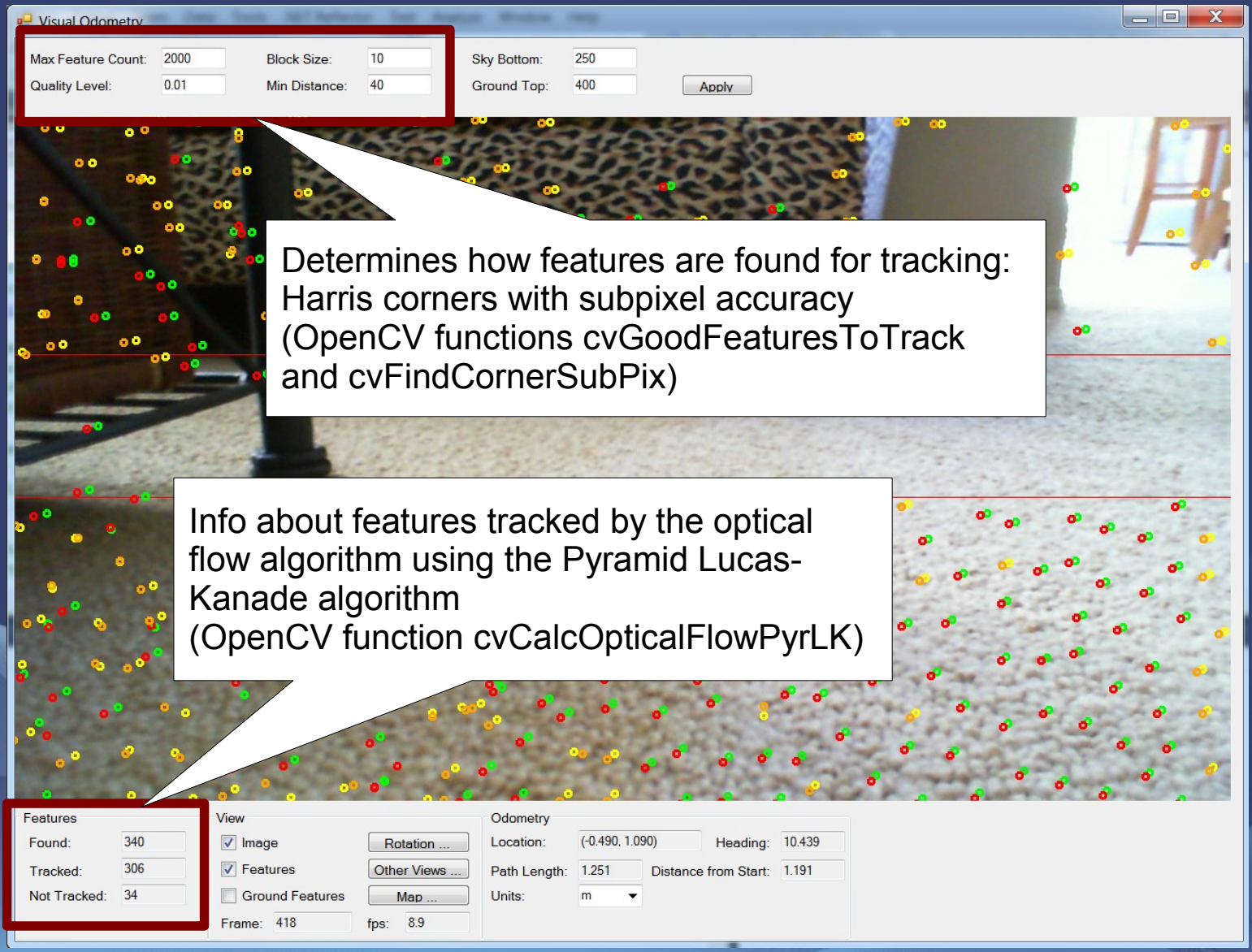
High-Level Algorithm

- Rotation: From optical flow above the 'horizon'
- Translation: From optical flow below the 'horizon'



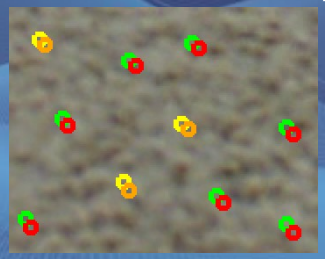
More Details

- Get camera image and correct it
- Estimate optical flow field for recent video frames
- Determine potential tracking errors and discard associated points
- Project 'sky' points into robot centered cylindrical coordinates and estimate rotation
- For ground region: Project onto floor plane, determine consensus x-y translation
- Sum incremental rotation and translation
- Periodically repopulate track points


Visual Odometry
Max Feature Count: 2000
Block Size: 10
Sky Bottom: 250
Quality Level: 0.01
Min Distance: 40
Ground Top: 400
Apply
Determines how features are found for tracking: Harris corners with subpixel accuracy (OpenCV functions cvGoodFeaturesToTrack and cvFindCornerSubPix)
Info about features tracked by the optical flow algorithm using the Pyramid Lucas-Kanade algorithm (OpenCV function cvCalcOpticalFlowPyrLK)
Features
Found: 340
Tracked: 306
Not Tracked: 34
View
Image
Features
Ground Features
Rotation ...
Other Views ...
Map ...
Frame: 418
fps: 8.9
Odometry
Location: (-0.490, 1.090)
Heading: 10.439
Path Length: 1.251
Distance from Start: 1.191
Units: m

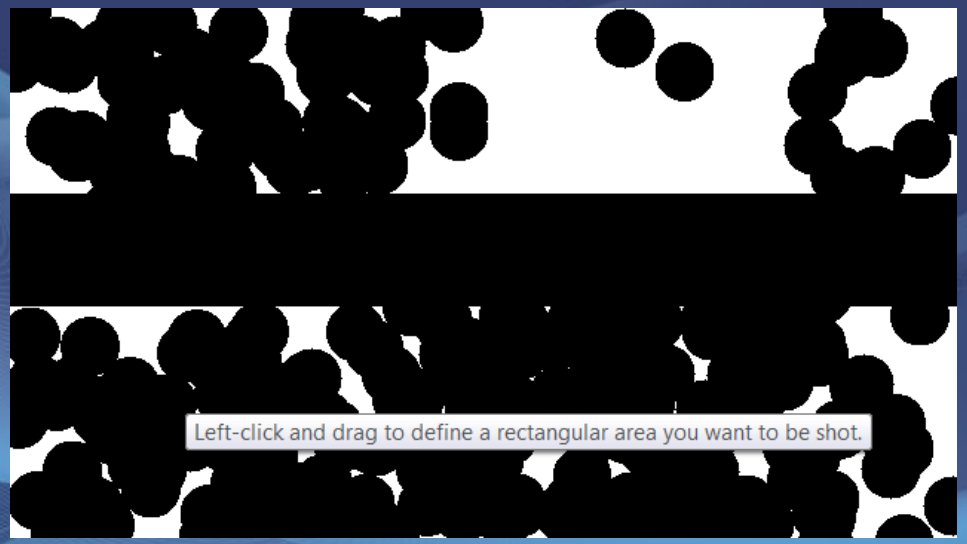
Smoothness of Tracked Features

- Keep history of feature location over the last seven frames
- Smoothness determined by comparing distance and direction changes over history
- If most features are smooth, weed out the bad features; otherwise assume jerky robot movement and keep them all
- Color code (previous vs current frame):
  - Full history: green → red
  - Incomplete: yellow → orange



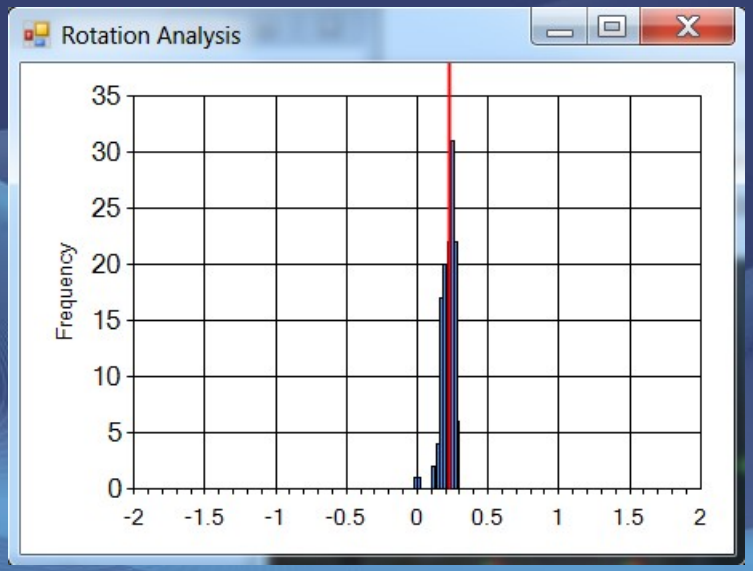
Repopulate Features

- Features are regularly removed
  - Feature moves out of the frame
  - Corresponding point cannot be found
- If feature count drops below threshold, search for new features using a mask around the existing features.



Direction Change (Robot Rotation)

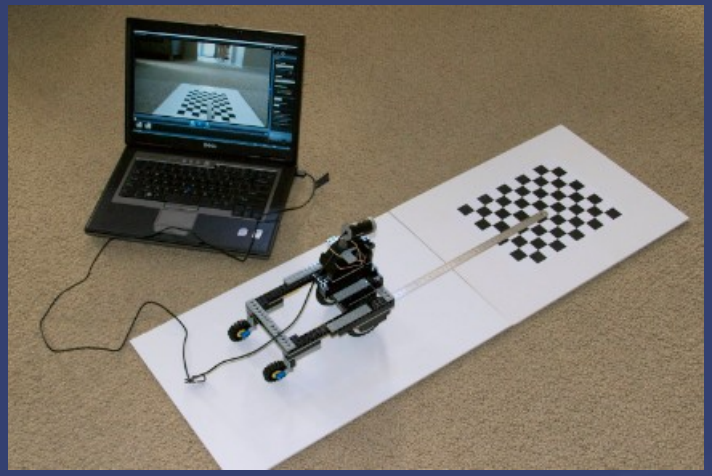
- Median of the angular rotation determined from previous / current feature pair in the sky region.
- Requires camera calibration (center x, focal length x)

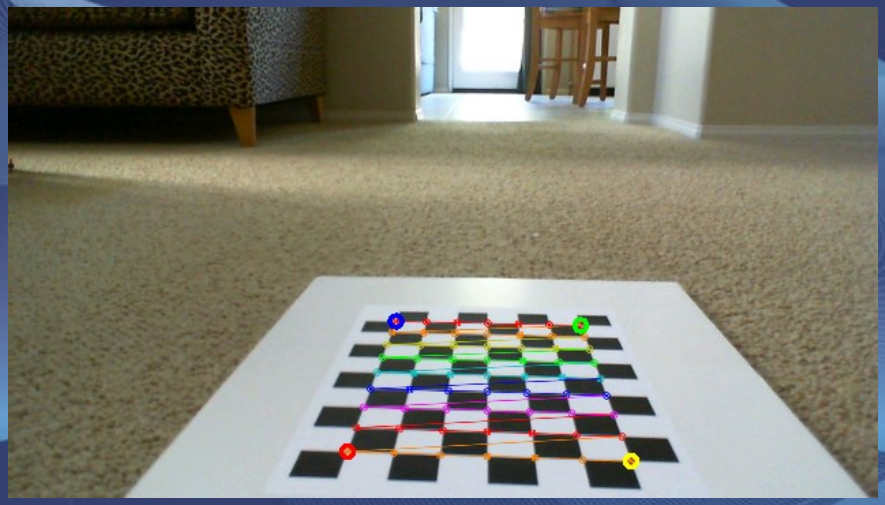


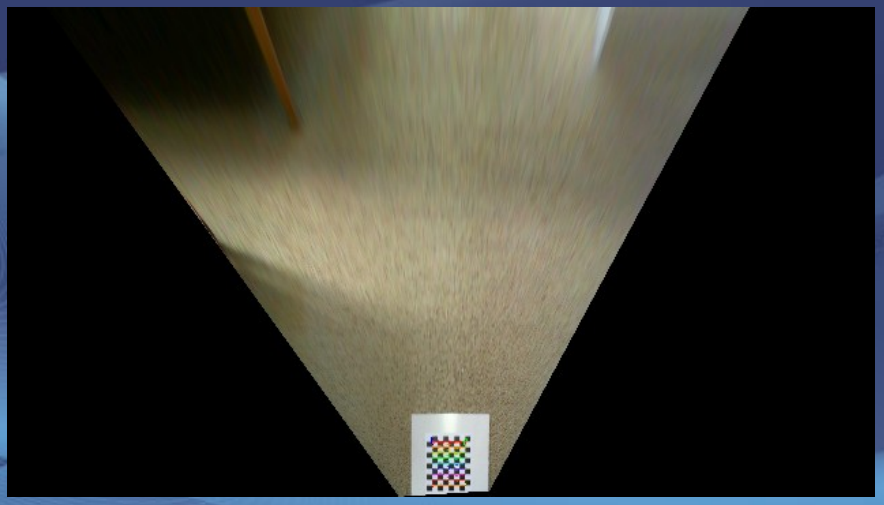
Histogram of measured angles. The red line marks the chosen calculated angle.

x-y Translation (Part 1)

- Ground floor projection
- Calibration: Chessboard on the ground in known location
- cvGetPerspectiveTransform

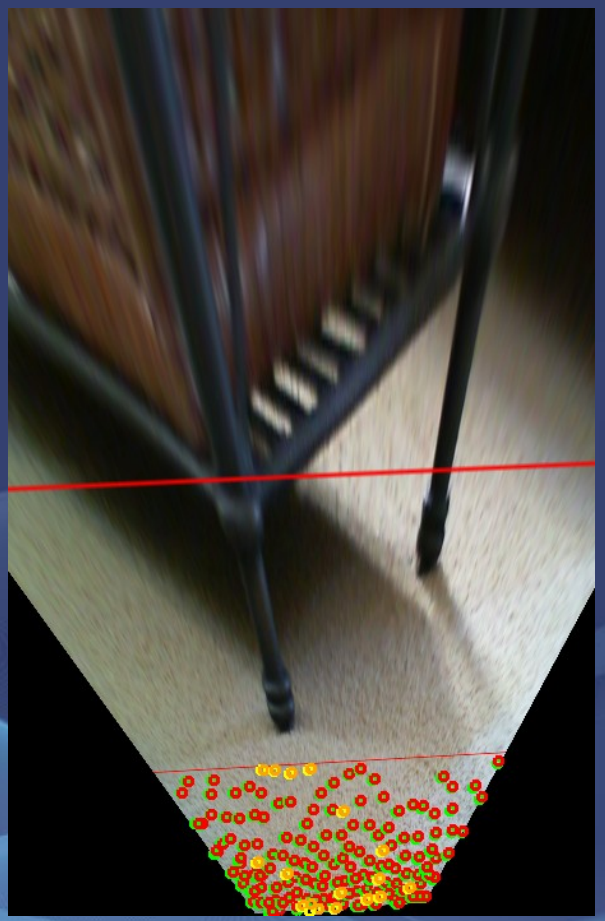






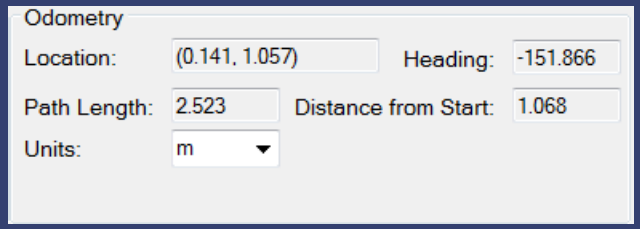
x-y Translation (Part 2)

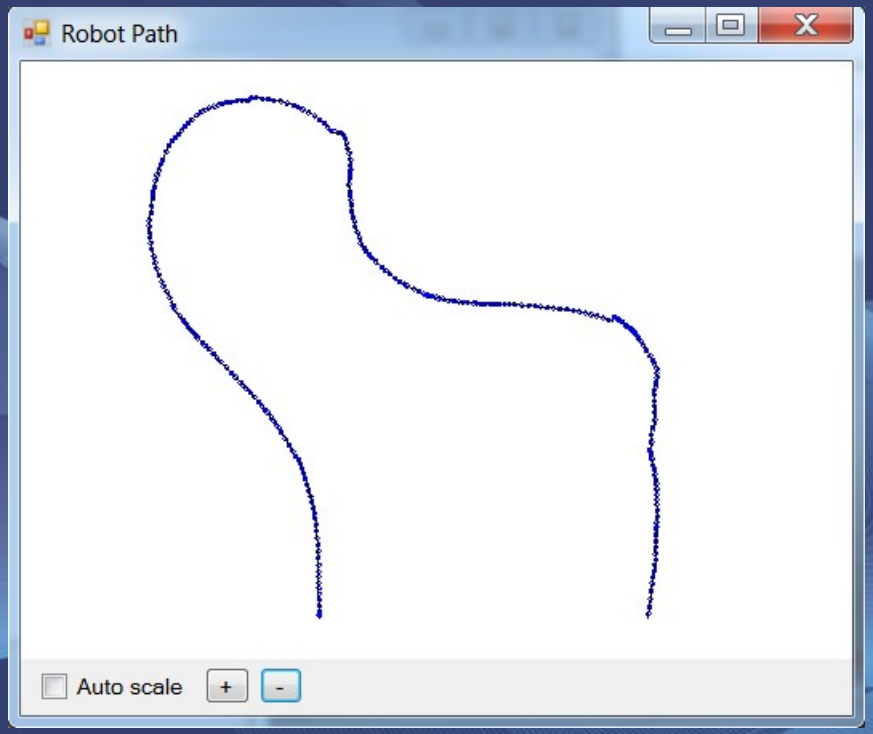
- Determine feature locations on the ground (below horizon)
- Remove rotation effect
- Consensus x-y translation:
  - Randomly pick 40 features
  - For each find number of other features with similar translation
  - Use feature with the biggest following and take the median translation of the group

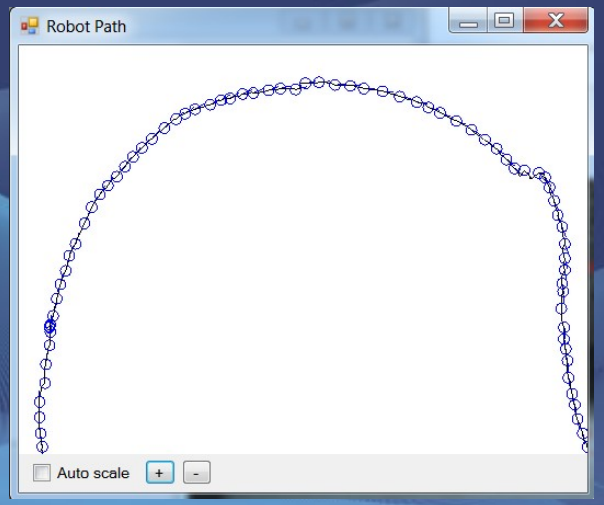


Robot Path

- Sum of incremental rotations and translations







Software

- C# 4.0 VisualStudio 2010
- Open Source GPL 3.0
- http://code.google.com/p/drh-visual-odometry/