# Computer Vision Lab Rapport 2017

Yoann Fleytoux

Aurélien Bernier

## Monocular Visual Odometry

For the implementation of this example write a small report where you should
Try to answer the following questions:

a) Qual é o objectivo do exemplo?
b) No exemplo é referida, várias vezes, a pose da câmara. O que lhe parece que é a pose da câmara?
c) A posição da segunda câmara relativamente à primeira é determinada a menos de um factor de escala. O que poderá ser esse factor de escala?
d) No resultado final duas trajectórias são mostradas. Essas trajectórias são linhas curvas em 3D. A que correspondem os pontos que constituem essas linhas? Será que o exemplo calcula mais informação que poderia ser mostrada?

The example is divided into three parts:

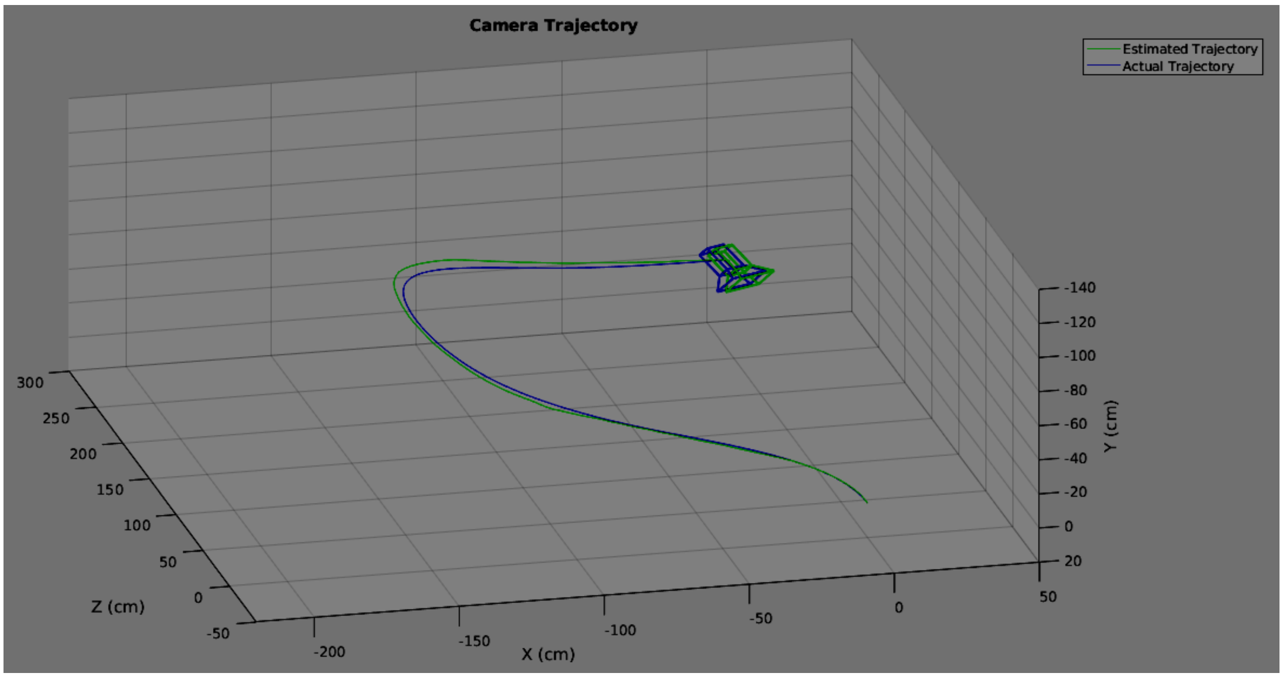
1. **Estimating the pose of the second view relative to the first view.** Estimate the pose of the second view by estimating the essential matrix and decomposing it into camera location and orientation.

2. **Bootstrapping estimating camera trajectory using global bundle adjustment.** Eliminate outliers using the epipolar constraint. Find 3D-to-2D correspondences between points triangulated from the previous two views and the current view. Compute the world camera pose for the current view by solving the perspective-n-point (PnP) problem. Estimating the camera poses inevitably results in errors, which accumulate over time. This effect is called *the drift*. To reduce the drift, the example refines all the poses estimated so far using bundle adjustment.

3. **Estimating remaining camera trajectory using windowed bundle adjustment.** With each new view the time it takes to refine all the poses increases. Windowed bundle adjustment is a way to reduce computation time by only optimizing the last *n* views, rather than the entire trajectory. Computation time is further reduced by not calling bundle adjustment for every view.

**Algorithm:**

**Read Input Image Sequence and Ground Truth**
**Create a View Set Containing the First View of the Sequence**
**Plot Initial Camera Pose**
**Estimate the Pose of the Second View**
**Bootstrap Estimating Camera Trajectory Using Global Bundle Adjustment**
**Estimate Remaining Camera Trajectory Using Windowed Bundle Adjustment**



## *A) What is the purpose of the example?*

Visual odometry is the process of determining the location and orientation of a camera by analyzing a sequence of images.
Visual odometry is used in a variety of applications, such as mobile robots, self-driving cars, and unmanned aerial vehicles.
This example shows you how to estimate the trajectory of a single calibrated camera from a sequence of images.
This example shows how to estimate the trajectory of a calibrated camera from a sequence of 2-D views.

This example expresses the estimation of the trajectory according to a calibrated monocular camera from a number of frames of a video. We can see that there’s a slight error compared to the “ground truth”. The issue here being that, as the trajectory refinement is not linear, the errors accumulate to finally get a bigger and bigger drift. Usually, the technique used to fixed this is to use multiple sensors and crossing their data to find the most likely trajectory.

## *B) In the example the chamber pose is referred to several times. What do you think is the camera pose?*

In computer vision and in robotics, a typical task is to identify specific objects in an image and to determine each object's position and orientation relative to some coordinate system. The combination of position and orientation is referred to as the pose of an object.

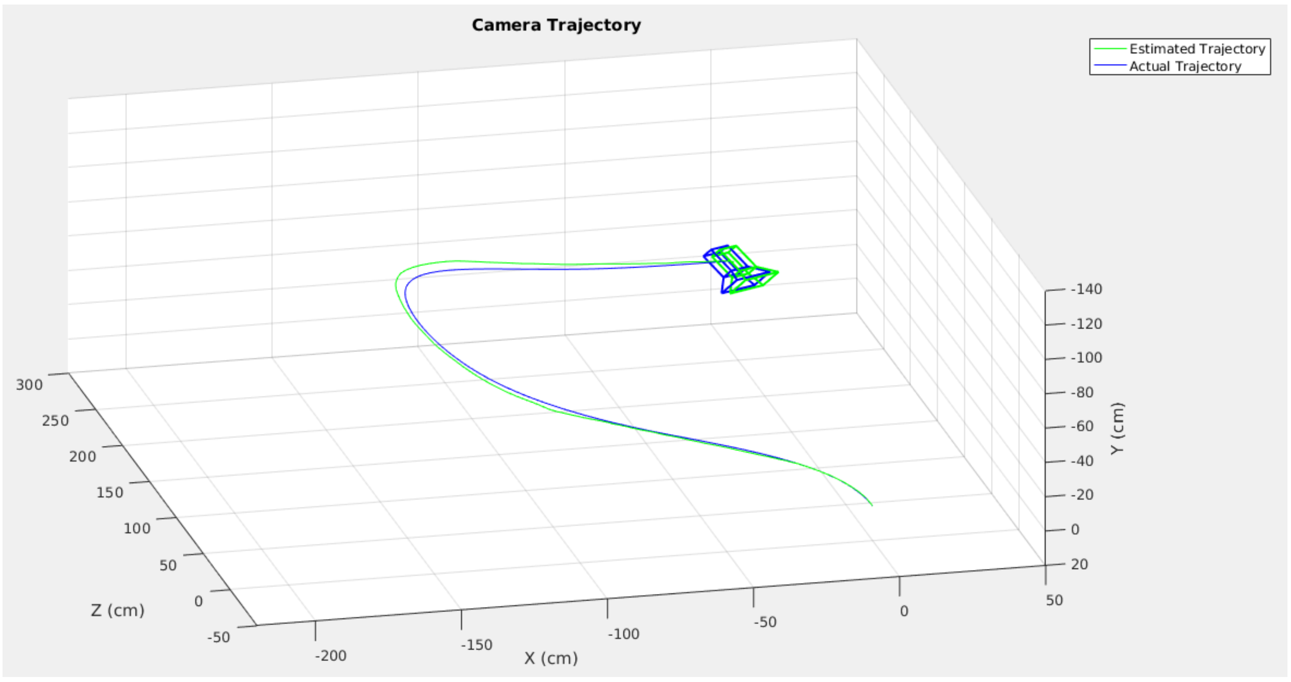
## *C) The position of the second chamber relative to the first is determined by less than one scale factor. What can this scale factor be?*

The location of the second view relative to the first view can only be recovered up to an unknown scale factor. Compute the scale factor from the ground truth using helperNormalizeViewSet, simulating an external sensor, which would be used in a typical monocular visual odometry system.

```
% Bundle adjustment can move the entire set of cameras. Normalize the
% view set to place the first camera at the origin looking along the
% Z-axes and adjust the scale to match that of the ground truth.
vSet = helperNormalizeViewSet(vSet, groundTruthPoses);
```

*ground truth:
Ground truth is a term used in various fields to refer to information provided by direct observation as opposed to information provided by inference.

*D) In the final result two trajectories are shown. These trajectories are*
*3D curved lines. What do the points constituting lines? Does the example calculate more information that could be shown?*



This example computes the scale factor from the ground truth.

We can see the green line being the estimate of the actual trajectory, which is the blue line. We can see that the difference gets bigger and bigger with the accumulation of errors.

# Annex - Useful concepts:

**Monocular visual odometry system:**
In robotics and computer vision, visual odometry is the process of determining the position and orientation of a robot by analyzing the associated camera images

**The Pose:**
In computer vision and in robotics, a typical task is to identify specific objects in an image and to determine each object's position and orientation relative to some coordinate system. The combination of position and orientation is referred to as the pose of an object

**The ground truth camera poses:**
Ground truth is a term used in various fields to refer to information provided by direct observation as opposed to information provided by inference.

**Computer stereo vision:**
is the extraction of 3D information from digital images, such as obtained by a CCD camera. By comparing information about a scene from two vantage points, 3D information can be extracted by examination of the relative positions of objects in the two panels. This is similar to the biological process Stereopsis.

**Epipolar constraint:**
Epipolar geometry is the geometry of stereo vision. When two cameras view a 3D scene from two distinct positions, there are a number of geometric relations between the 3D points and their projections onto the 2D images that lead to constraints between the image points. These relations are derived based on the assumption that the cameras can be approximated by the pinhole camera model

**Pinhole camera:**
A pinhole camera is a simple camera without a lens but with a tiny aperture, a pinhole – effectively a light-proof box with a small hole in one side. Light from a scene passes through the aperture and projects an inverted image on the opposite side of the box, which is known as the camera obscura effect..

**Pinhole camera model:**
The pinhole camera model describes the mathematical relationship between the coordinates of a 3D point and its projection onto the image plane of an ideal pinhole camera, where the camera aperture is described as a point and no lenses are used to focus light. The model does not include, for example, geometric distortions or blurring of unfocused objects caused by lenses and finite sized apertures. It also does not take into account that most practical cameras have only discrete image coordinates. This means that the pinhole camera model can only be used as a first order approximation of the mapping from a 3D scene to a 2D image. Its validity depends on the quality of the camera and, in general, decreases from the center of the image to the edges as lens distortion effects increase.

Some of the effects that the pinhole camera model does not take into account can be compensated, for example by applying suitable coordinate transformations on the image coordinates; other effects are sufficiently small to be neglected if a high quality camera is used. This means that the pinhole camera model often can be used as a reasonable description of how a camera depicts a 3D scene, for example in computer vision and computer graphics.

**Essential matrix:**
In computer vision, the essential matrix is a 3 × 3 matrix, E, with some additional properties described below, which relates corresponding points in stereo images assuming that the cameras satisfy the pinhole camera model.

**Drift:**
perspective-n-point (PnP) probledem:
errors, which accumulate over time. This effect is called the drift

**Pnp problem:**
Perspective-n-Point is the problem of estimating the pose of a calibrated camera given a set of n 3D points in the world and their corresponding 2D projections in the image. The camera pose consists of 6 degrees-of-freedom (DOF) which are made up of the rotation (roll, pitch, and yaw) and 3D translation of the camera with respect to the world. This problem originates from camera calibration and has many applications in computer vision and other areas, including 3D pose estimation, robotics and augmented reality. A commonly used solution to the problem exists for $n = 3$ called P3P, and many solutions are available for the general case of $n \geq 3$. Implementations of these solutions are also available in open source software

**Bundle adjustment:**
Given a set of images depicting a number of 3D points from different viewpoints, bundle adjustment can be defined as the problem of simultaneously refining the 3D coordinates describing the scene geometry, the parameters of the relative motion, and the optical characteristics of the camera(s) employed to acquire the images, according to an optimality criterion involving the corresponding image projections of all points.

**Loop closure:**
Loop closure is the problem of recognizing a previously-visited location and updates the beliefs accordingly. This can be a problem because model or algorithm errors can assign low priors to the location. Typical loop closure methods apply a second algorithm to measure some type of sensor similarity, and re-set the location priors when a match is detected. For example, this can be done by storing and comparing bag of words vectors of SIFT features from each previously visited location.