AVG Practical Session 3 : Homography-based visual odometry

Fall Semester

1 Content

In this lab, a video made by holding a camera while moving around a table on which a plane was placed is provided. The goal of this lab is to compute the camera's poses during this video using Visual Odometry (VO). Since the ground truth is not available, the odometry result is evaluated qualitatively by drawing the plane's local frame onto every image frame (as shown in Fig.1).

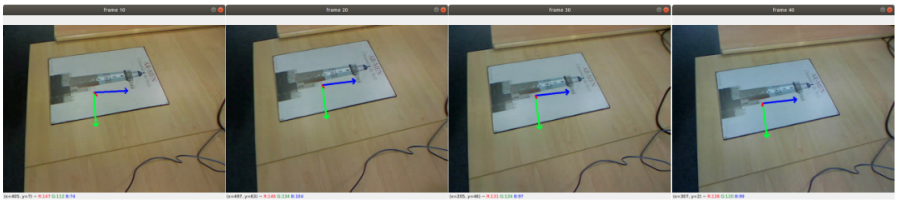

FIGURE 1 – Plane's local frame drawn on four different image frames (frame span is 10). Here, the x-, y-, z-axis are respectively denoted by blue, green, red arrow.

To be more specific, a frame $O^p x^p y^p z^p$ is created such that
— O^p is on the plane
— z^p is parallel to the plane's normal direction
This frame is rigidly attached to the plane, and is used as the plane's local frame. The pose of $O^p x^p y^p z^p$ with respect to the first camera frame $\mathcal{C}^0$ is known under the assumption that
— the x-axis of the first camera frame x^0 is in the same direction as x^p
— the z-axis of the first camera frame z^0 is deviated from z^p by a small angle
This pose is mapped into the i-th camera frame $\mathcal{C}^i$ thanks to the homogeneous transformation ${}^{\mathcal{C}^i}M_{\mathcal{C}^0}$ computed by VO, then projected onto i-th camera's image using camera intrinsic matrix. If the VO is done correctly, the appearance of the plane's frame on the i-th camera's image should remain unchanged (like the rest of the plane).

2 Plane-based Visual Odometry

2.1 Theory

The theory presented in the following section extensively uses the notion of ***camera frame*** $\mathcal{C}^i (i = \overline{0, n})$. A camera frame doesn't refer to the image captured by the camera, but rather denote the i-th 3D pose of the camera with respect to a world frame. Since

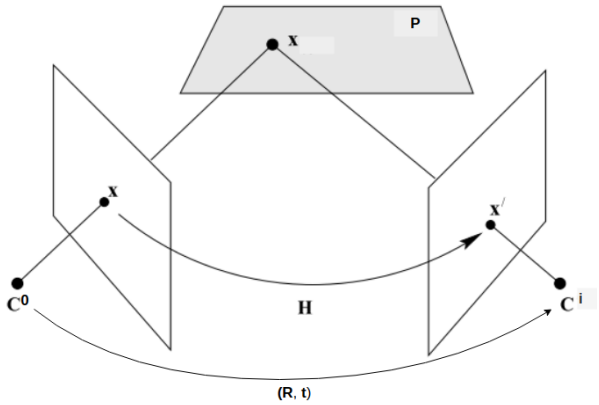

FIGURE 2 – Homography induced by a plane

the world frame can be chosen arbitrarily, it can chosen as the first camera frame $\mathcal{C}^0$. On the other hand, the i-th ***image frame*** refers to the image captured by the camera at the pose $\mathcal{C}^i$.

The basis of the visual odometry based on homography is to compute a homography $\mathbf{H}$ from two images of a plane (Fig.2). This homography is responsible for mapping a pixel $\mathbf{x}^0$ in the first image, which is captured with camera at the pose $\mathcal{C}^0$, to a pixel $\mathbf{x}^i$ in the second image, which is captured with camera at the pose $\mathcal{C}^i$, according to Eq.(1).

$$\mathbf{x}^i = \mathbf{H} \cdot \mathbf{x}^0 \tag{1}$$

The plane equation expressed in the first camera frame C^0 can be written as following

$$\mathbf{X}^T \cdot \mathbf{n} = d \tag{2}$$

Here, $\mathbf{X}, \mathbf{n}$ are 3-vector respectively representing a 3D point on the plane and the normal vector the plane, while d is a scalar denotes the distance from the origin of $\mathcal{C}^0$ to the plane.

Given the plane equation Eq.(2), the homography $\mathbf{H}$ can be computed by

$$\mathbf{H} = \mathbf{R} + \mathbf{t} \cdot \frac{\mathbf{n}^T}{d} \tag{3}$$

In this equation, $(\mathbf{R}, \mathbf{t})$ are the rotation matrix and translation vector makes up the transformation from the first camera frame $\mathcal{C}^0$ to the i-th camera frame $\mathcal{C}^i$ as shown in Fig.2.

The process of homography-based VO can be summed up in the following steps

1. Detect keypoints and compute their descriptors from the incoming frame
2. Find matches between keypoints in the incoming frame with those in the source frame (e.g. image capture when camera's pose is $\mathcal{C}^0$)
3. Using RANSAC, compute homography $\mathbf{H}$ and also get inliners (matched keypoints that agree with $\mathbf{H}$)
4. Decompose $\mathbf{H}$ to get $(\mathbf{R}, \mathbf{t}, \mathbf{n})$ candidates (usually 4).
5. Prune candidates that give a negative depth for at least 1 inlier.
6. There still may be two candidates, keep the one that has the normal $\mathbf{n}$ closer to the current estimation of the plane's normal vector.

2.2 Algorithm Details

2.2.1 Pose of the plane's local frame in the first camera frame $\mathcal{C}^0$

As mentioned in the Section.1, the VO result is qualitative evaluated by drawing the plane's local frame $O^p x^p y^p z^p$ in every image frame. To do this, the pose of $O^p x^p y^p z^p$ in the first camera frame $\mathcal{C}^0$ is required.

The position of the origin of the plane's local frame O^p is defined by ${}^{\mathcal{C}^0}\mathbf{t}_p$

$$ {}^{\mathcal{C}^0}\mathbf{t}_p = \begin{bmatrix} -0.1 & -0.1 & 0.7 \end{bmatrix}^T \tag{4} $$

Under the assumption about the direction of x^p and z^p relisted below for convenience,

— the x-axis of the first camera frame x^0 is in the same direction as x^p

— the z-axis of the first camera frame z^0 is deviated from z^p by a small angle

the orientation of the plane's local frame ${}^{\mathcal{C}^0}\mathbf{R}_p$ is

$$ {}^{\mathcal{C}^0}\mathbf{R}_p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{5} $$

Here α is a small angle which can be tuned to get a nicer visualization.

2.2.2 Compute the scale factor of the translation inferred from H

The translation vector recovered by decomposing the homography $\mathbf{H}$ is in fact $\mathbf{t}/d$ (d is distance from the origin of $\mathcal{C}^0$ to the plane), because they cannot be separated without extra information. This extra information can come from the position of the plane ${}^{\mathcal{C}^0}\mathbf{t}_p$ in the first camera frame $\mathcal{C}^0$.

Given ${}^{\mathcal{C}^0}\mathbf{t}_p$ and the estimated normal vector $\mathbf{n}$, the distance d is calculated as

$$ d = \mathbf{n}^T \cdot {}^{\mathcal{C}^0}\mathbf{t}_p \tag{6} $$

Having the value of d, the translation vector resulted from the decomposition of $\mathbf{H}$ can be easily scaled to get $\mathbf{t}$ (the translation from first camera frame $\mathcal{C}^0$ to the i-th camera frame $\mathcal{C}^i$).

It is important to know that the calculation of d should be carried out only once if the source frame is kept unchanged (e.g source frame is the image captured when camera at $\mathcal{C}^0$). On the other hand, if the source frame is updated, d must be updated as well.

2.2.3 Compute sign of iniliers' depth

The decomposition of the homography $\mathbf{H}$ usually yields 4 solutions, at least 2 of which result in negative depth of some inliers. Therefore, to prune these false solutions the depth of inliers or at least their sign need to be computed.

Expand Eq.2 by replacing $\mathbf{X}$ and $\mathbf{n}$ with $[\mathrm{X}, \mathrm{Y}, \mathrm{Z}]^T$ and $[\mathrm{n_x}, \mathrm{n_y}, \mathrm{n_z}]^T$ respectively

$$ \mathrm{X}\ \mathrm{n_x} + \mathrm{Y}\ \mathrm{n_y} + \mathrm{Z}\ \mathrm{n_z} = d \tag{7} $$

Divide both sign of the equation above for Z and rewrite it into vector form,

$$ \begin{bmatrix} \frac{\mathrm{X}}{\mathrm{Z}} & \frac{\mathrm{Y}}{\mathrm{Z}} & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathrm{n_x} \\ \mathrm{n_y} \\ \mathrm{n_z} \end{bmatrix} = \frac{d}{\mathrm{Z}} \tag{8} $$

Notice that the first term on the left-hand side of Eq.(8) is the normalized coordinate of a 3D point in the camera frame which can be calculated from camera intrinsic matrix $\mathbf{K}$ and the pixel coordinate of its projection on the camera (u, v). Eq.(8) can be transformed into

$$\left(\mathbf{K}^{-1} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}\right)^T \cdot \begin{bmatrix} \mathrm{n_x} \\ \mathrm{n_y} \\ \mathrm{n_z} \end{bmatrix} = \frac{d}{\mathrm{Z}} \tag{9}$$

Since d is positive number by definition, the sign of Z is the sign of the left-hand side of Eq.(9)

$$\mathrm{sign}\,(\mathrm{Z}) = \mathrm{sign}\left(\left(\mathbf{K}^{-1} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}\right)^T \cdot \begin{bmatrix} \mathrm{n_x} \\ \mathrm{n_y} \\ \mathrm{n_z} \end{bmatrix}\right) \tag{10}$$

3 Implementation Details

The programming work of this lab is made of two Python script `main_vo.py` and `visual_odometry.py`. The first script represents the highest level of abstraction. Upon receiving a new image frame, this script invokes methods of an instance of the class `VisualOdometry` (defined in `visual_odometry.py`) to calculate the relative transformation, then draw the plane's local frame onto the incoming (i.e. received) image frame.

3.1 `VisualOdometry` Class

This class is the implementation of homography-based VO which consists of three high-level steps

1. Find the matches of keypoints on two images of the plane
2. Infer $(\mathbf{R}, \mathbf{t}, \mathbf{n})$ candidates from the homography $\mathbf{H}$
3. Prune the wrong candidates

For convenience, among two images of the plane, the older (with respect to time) one is referred to as the **source frame**, while newer is called **incoming frame**. This naming convention is to illustrate two points

— The pose of the camera with respect to the world frame chosen as $\mathcal{C}^0$, ${}^{\mathcal{C}^0}\mathbf{M}_{\mathcal{C}^{src}}$, when it captured source frame is known

— The Homography $\mathbf{H}$ is used to map pixels from source frame to incoming frame, thus $(\mathbf{R}, \mathbf{t})$ recovered by decomposing $\mathbf{H}$ makes up the transformation, ${}^{\mathcal{C}^{inc}}\mathbf{M}_{\mathcal{C}^{src}}$, from $\mathcal{C}^{src}$ to $\mathcal{C}^{inc}$

The initial value of the source frame is naturally the first image in the video, thus making the first camera frame $\mathcal{C}^0$ become the source camera frame $\mathcal{C}^{src}$. As the video progressing, the source frame can be either kept unchanged or updated periodically by incoming frame. In the first case, $\mathcal{C}^{src}$ remains unchanged, while it is updated in the second case.

3.1.1 Attributes

The important attributes of this class are listed below

— `camera_intrinsic` : intrinsic matrix of the camera used to make the video

— `orb` : an instance of OpenCV `ORB` class for detecting ORB keypoints and computing ORB descriptor

— `matcher` : an instance of OpenCV `BFMatcher` class for matching keypoints
— `plane_d_src` : distance from the plane to the source camera frame $\mathcal{C}^{src}$
— `c0_M_p` : the mapping, ${}^{\mathcal{C}^0}\mathbf{M}_p$, from the plane's local frame to the first camera frame
— `plane_normal_src` : plane's normal vector expressed in the source camera frame. The initial value of this is the 3rd column of ${}^{\mathcal{C}^0}\mathbf{M}_p$ since the plane's normal vector is its z-axis, and the source camera frame is initialized by the first camera frame $\mathcal{C}^0$
— `plane_origin_src` : coordinate of plane's origin O^p expressed in the source camera frame
— `src_M_c0` : the mapping, ${}^{\mathcal{C}^{src}}\mathbf{M}_{\mathcal{C}^0}$, from the first camera frame $\mathcal{C}^0$ to the source camera $\mathcal{C}^{src}$. Its value is the identity matrix since the source camera frame is initialized by the first camera frame $\mathcal{C}^0$

3.1.2 `run` method

This method plays the role of the main function of the VO process which receives the incoming frame (i.e. an BGR image) and sequentially performs

1. Matching keypoints in the incoming frame to keypoints in the source frame by invoking `find_matches` method
2. Calling `compute_relative_transform` method to process the matched keypoints to find the transformation, ${}^{\mathcal{C}^{\mathrm{inc}}}\mathbf{M}_{\mathcal{C}^{src}}$, from source camera frame $\mathcal{C}^{src}$ to incoming camera frame $\mathcal{C}^{inc}$
3. Computing the transformation, ${}^{\mathcal{C}^{\mathrm{inc}}}\mathbf{M}_{\mathcal{C}^0}$, from the first camera frame $\mathcal{C}^0$ to the incoming (i.e. current) camera frame $\mathcal{C}^{\mathrm{inc}}$ using ${}^{\mathcal{C}^{inc}}\mathbf{M}_{\mathcal{C}^{src}}$ just computed above and cached transformation, ${}^{\mathcal{C}^{src}}\mathbf{M}_{\mathcal{C}^0}$, from the first camera frame $\mathcal{C}^0$ to the source camera frame $\mathcal{C}^{src}$

The output of this method is ${}^{\mathcal{C}^{inc}}\mathbf{M}_{\mathcal{C}^0}$ (a 4-by-4 matrix).

`if update_src_frame` In the case that source frame is updated by incoming frame, the attribute `src_M_c0` need to be updated accordingly using its current value and the relative transformation ${}^{\mathcal{C}^{inc}}\mathbf{M}_{\mathcal{C}^{src}}$ computed by method `compute_relative_transform`.

3.1.3 `find_matches` method

This method takes as input the incoming frame (an BGR image) and sequentially perform

1. Conversion to gray scale image using `cv2.cvtColor`
2. Identifying keypoints and computing their descriptor by invoking method `detectAndCompute` of attribute `orb`
3. Finding matches between keypoints in source frame and incoming frame by invoking method `match` of attribute `matcher`

The output of this method is
— An empty `list` if this is the first image frame
— A `list` of `cv2.DMatch` (the output of method `match` invoked in the last step above), otherwise

3.1.4 `compute_relative_transform` method

This method performs step 3 - 6 of the algorithm presented in the end of Section.2.1. Its input is the `list` of `cv2.DMatch` produced by method `find_matches`, and its output is transformation , ${}^{\mathcal{C}^{inc}}\mathbf{M}_{\mathcal{C}^{src}}$, from source camera frame $\mathcal{C}^{src}$ to incoming camera frame $\mathcal{C}^{inc}$.

The functions of OpenCV used in this method are listed below

- `cv2.findHomography` : returns the homography $\mathbf{H}$ (a 3x3 matrix) and a mask (binary NumPy array) of inliers
- `cv2.decomposeHomographyMat` : returns the number of candidates, a `list` of 3x3 matrix representing the rotation matrix $\mathbf{R}$, a `list` of 3x1 vector representing the translation vector $\mathbf{t}$, and a `list` of 3x1 vector representing the normal vector $\mathbf{n}$

`if update_src_frame` As mentioned in Section.3.1, the source frame can be updated with incoming frame, leading to the following extra steps need to be carried out

1. Overwrite attributes `src_kpts` and `src_desc` with keypoints and descriptors extracted from incoming frame
2. Map the plane's normal vector (attribute `plane_normal_src`) from source camera frame to incoming camera frame using rotation matrix $\mathbf{R}$
3. Compute the depth for every inlier using Eq.(9) and the current value of d (attribute `plane_d_src`)
4. Compute the 3D coordinate in the source camera frame for every inlier using their depth and normalized coordinate
5. Map these 3D coordinate to the incoming camera frane
6. Average the projection of these 3D coordinate on the plane's normal vector (attribute `plane_normal_src`) to get the new value for d (attribute `plane_d_src`)

3.2 `main_vo.py` script

As mentioned in the beginning of Section.3, this script executes high-level steps which are

1. Read the video file
2. Invoke `run` method of class `VisualOdometry` for the transformation, ${}^{\mathcal{C}^{inc}}\mathbf{M}_{\mathcal{C}^0}$, from the first camera frame $\mathcal{C}^0$ to the incoming (i.e. current) camera frame $\mathcal{C}^{inc}$
3. Map the plane's local frame from $\mathcal{C}^0$ to $\mathcal{C}^{inc}$ and project it onto the image of $\mathcal{C}^{inc}$

4 Expected Work

Since the purpose of this lab is to implement homography-based visual odometry, the `main_vo.py` script is finished. The lab work is focused on the class `VisualOdometry`. To reduce the complexity, a skeleton of this class is provided in the script `visual_odometry.py`. Your task is to fill the blank marked by **TODO** in this skeleton code. If your implementation is correct, the result should look like this video https://youtu.be/MYW_VMwzwJ8.