Discussion Forums / Week 2

Assignment: Visual Odometry for Localization in Autonomous Driving

Assignment: Visual Odometry for Localization in Autonomous Driving

SUBFORUMS

All

Assignment: Visual Odometry for Localization in Autonomous Driving

Some hints on assignment

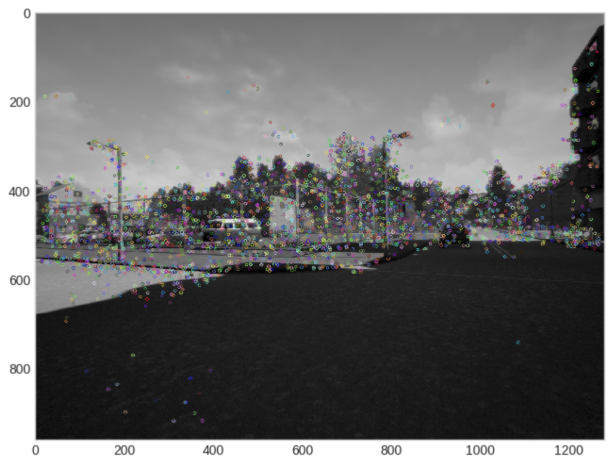
Anton Tmur Assignment: Visual Odometry for Localization in Autonomous Driving · 9 months ago

This assignment is my worst nightmare. I've spent hours and hours (actually, a couple of months) solving it. So I've decided to give some additional information and hints (no code) for those who will solve this assignment later.

Feature Extraction. I have tried ORB, SURF, SIFT and BRIEF. For me SURF appeared to work best. If you use ORB some sources recommend to set (*nfeatures=5000, WTA_K=3* or *4*) as input arguments of ORB_create.

I have tried doing the sorting of keypoints by it's response and cut the end of the list (found such a technique on the Internet). Don't do this!

At the end of section 1.1 you have to obtain something like this (colorful circles are the keypoints).

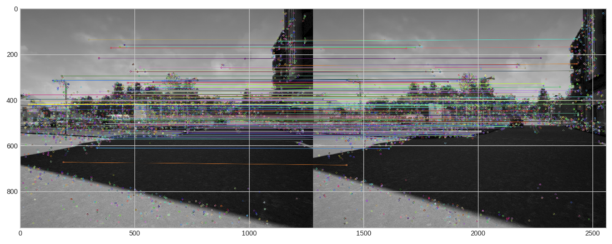


Feature Matching. I have tried BruteForce, knn and flann. All of them are well described opencv tutorial. If you are using the ORB, BRIEF, BRISK don't forget to choose cv2.NORM_HAMMING. For all others use cv2.NORM_L2 (see the same link for details)

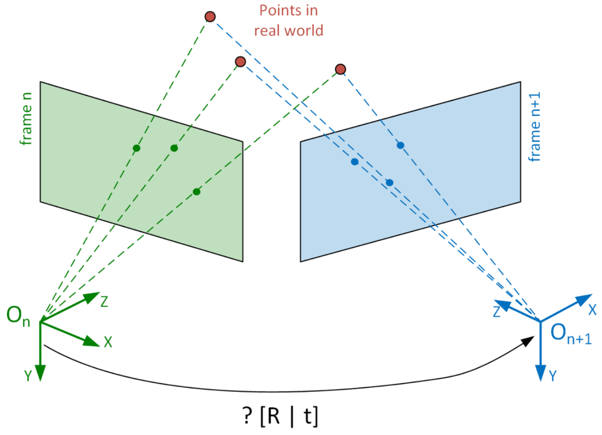
If you are doing the knn or flann, set k=2 as knnMatch input argument. It would make the further filtering easier.

Matches' filtering. Be sure to do it, despite the fact that it says "optional". If you are using Brute Force, just remove all the matches for which match.distance is less than threshold. For knn or flann methods use the ratio filtering which is also described in tutorial.

At the end of section 2.1 you will have something like this (horizontal lines show the matches between keypoints; if they are not horizontal, something went wrong).



Motion estimation. What we are trying to do here is to calculate the transition matrix *T* (which consist of rotation matrix *R* and translation vector *t*). This transition matrix will show us the camera pose when getting the frame *n+1* in terms of coordinate system corresponding to the camera at the frame *n*. In other words (see figure below) let us have the coordinates of the real world points (red ones) in terms of the coordinate system *On*.



We are trying to find the pair

$$(R_{n+1}|t_{n+1})$$

for camera at the moment n+1 so that image points at the frame n+1 (blue points) satisfy the equation:

$$s\begin{pmatrix}u\\v\\1\end{pmatrix}_{at\ frame\ n+1} = K\,(R_{n+1}|t_{n+1})\begin{pmatrix}X\\Y\\Z\\1\end{pmatrix}_{\substack{with\ respect\\ to\ camera\ at\ n}}$$

Looking at this equation we can notice:

K - the intrinsic matrix is known

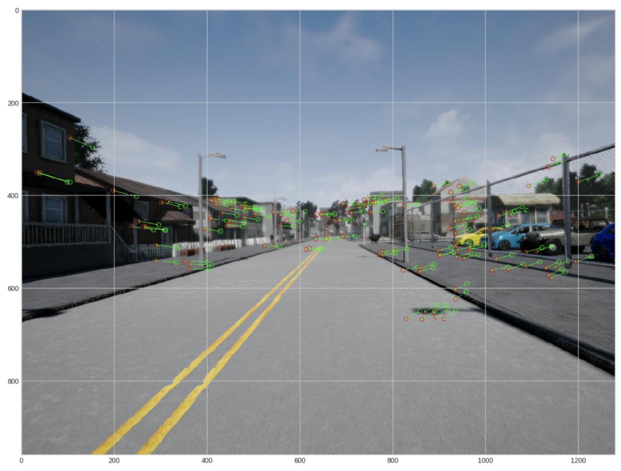
u,v at the frame *n* - can be found in the list of keypoints (you just need to know what is *match.trainidx* and *keypoint.pt*). It is also named as *image_points2*.

X,Y,Z with respect to camera at *n*. To find it we can first calculate the *image_points1* (analogously as in case of *image_points1* form the vector of *u, v, 1*, and using *match.queryidx*). Then we find the scale factor *s* for the point, taking advantage of the fact that the *Z* axis is directed deep into the image (see figure). Hence *s*1* must be equal to the depth of the corresponding pixel (which is provided). The last step to determine *X, Y, Z* is to look at the equation written completely for frame *n*.

$$K^{-1}\, s\begin{pmatrix}u\\v\\1\end{pmatrix}_{at\ frame\ n} = \overset{\curvearrowleft}{K}\ \underset{I}{\cancel{(R_n|t_n)}}\begin{pmatrix}X\\Y\\Z\\1\end{pmatrix}_{\substack{with\ respect\\ to\ camera\ at\ n}}$$

We transfer the matrix *K* to the left by inverting and replace *R,t* with the identity matrix, because we are in the reference system He (and it does not have turns and transfers relative to itself). Thus forming the 3D object points for frame n and 2D image points for frame *n+1* we can use these two lists as inputs to solvePnPRansac function. Please don't forget to filter points with depth > 900 (it means we do not actually know the depth and the point is somewhere far away).

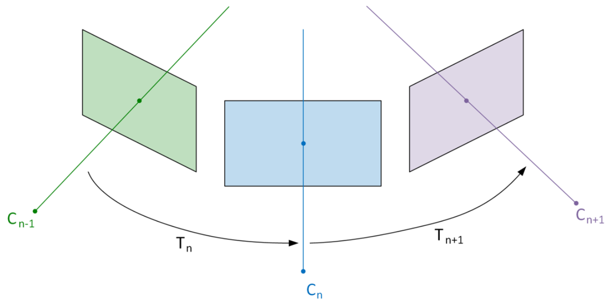
At the end of section 3.1 you will have something like this (for image 48):



Trajectory estimation. This is the most problematic part of assignment (because it was not covered by lectures or supplementary materials quite well). Now we know how to calculate the transformation matrix T:

$$T_n = \begin{vmatrix} R_n & t_n \\ 0^T & 1 \end{vmatrix}$$

Let us consider the sequence of frames.



Every transition matrix T gives us knowledge of how points expressed in terms of the *n + 1* frame of reference are related to the *n* frame of reference. Hence for any point P its coordinates in two reference systems are connected by the equation:

$$(P)_n = T_n \; (P)_{n-1}$$

The subscript n after the brackets means the coordinates in frame n. The reverse transition is:

$$(P)_{n-1} = T_n^{-1} \; (P)_n$$

The camera position at time n and in terms of frame n is always (as we assume camera is in the origin):

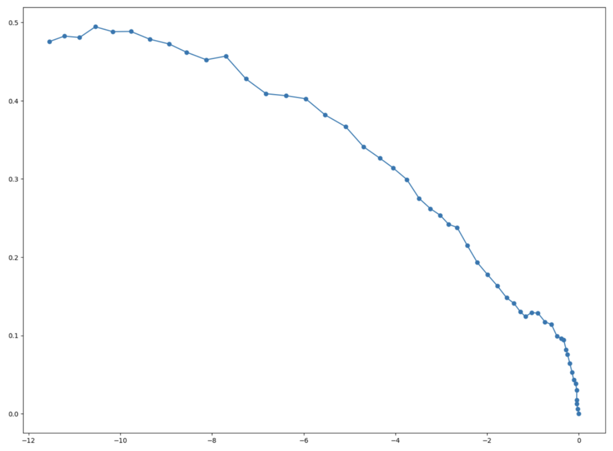
$$(C_n)_n = \begin{Vmatrix} 0 \\ 0 \\ 0 \\ 1 \end{Vmatrix}$$

It means that if we want to calculate the camera position at time t in terms of frame 0 we have to calculate all the reversion:

$$(C_n)_0 = T_1^{-1} T_2^{-1} \dots T_n^{-1} \; (C_n)_n = T_1^{-1} T_2^{-1} \dots T_n^{-1} \; \begin{Vmatrix} 0 \\ 0 \\ 0 \\ 1 \end{Vmatrix}$$

Now you know what to do. Every time moment n calculate the inverse of T, do the multiplication and take the first three elements of the last column of the product.

The final trajectory should look something like this:



Some additional helpful materials (thank to previous forum threads) could be found here and here (page 5).

PS Hope you won't lose so much time as I have. If this information helped you, please write "thanx"-comment to keep this thread above.

38 Upvotes Reply Follow this discussion

Earliest **Top** **Most Recent**

LH **Long Hui** · a month ago

thanx

0 Upvotes Reply

C **Carlos** · 5 months ago

This is not properly expressed and is quite confusing.

The first rule in robotics is that, given a point P referred to a frame n, the transformation FROM n-1 TO n (so, (n-1)T(n)) helps us to, given a point at frame n, to find its information at frame n-1.

However, you have written Tn (so, (n-1)T(n)) help us to get P(n) given P(n-1). This is not correct.

0 Upvotes Hide 1 Reply

SS **Sajid Sarkar** · 5 months ago

Hi Carlos,

Were you able to complete this assignment? I am stuck at the last part where my graphs are not making any sense.

0 Upvotes

DW Reply

Reply