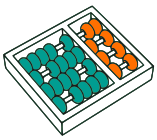

MC949/MO446 — Introduction to Computer Vision
INSTITUTO DE COMPUTAÇÃO — UNICAMP
2nd Semester, 2017

UNICAMP

Project 3

Prof. Adín Ramírez Rivera
adin@ic.unicamp.br

1 Description

In this project you will be reconstructing the 3D shape of the world from a series of images (video). There are three parts for this task: keypoint selection (not need for description), feature tracking, and structure from motion.

2 Data

Before starting the project you need to get some data. You can use the data available at http://vision.middlebury.edu/mview/data/ for a couple of classic datasets. Note that these datasets are a set of cameras in a hemisphere. Hence, no temporal data exist. However, you can use the movement from camera to camera as a way of creating the sequence you want. You can use other standard datasets for SfM or scene reconstruction if you want.

You are also encouraged to get your cellphone or camera, and go take a few seconds of video to something interesting with your hand and slowly moving around the object. Read the restrictions on the algorithms to get a better idea of what to record before doing it.

3 Keypoint Selection

You can use any keypoint selection that you want. However, the good features to track [1] are ones whose motion can be estimated reliably. Harris corner detector is a good choice to start. If you want to explore other selectors you can do it, just remember to explain and justify your selection in your report.

Before moving forward, evaluate the result of your keypoint selector on a couple of frames of your target sequences. Show your results (markers on the images) on your report.

4 Feature Tracking

In this part you will be implementing the Kanade-Lucas-Tomasi (KLT) tracking algorithm [2, 3] for the keypoints selected in § 3. Essentially, you will be implementing the optical flow between successive video frames. My suggestion is to capture videos with small movements to get a grasp of the algorithm. However, if you want to have a better and more robust algorithm you will need to implement a coarse-to-fine KLT (see § 7).

The assumption of the KLT tracker is brightness consistency between frames. One point should have the same value after the translation in the next frame. Let I be our image function, then our assumption is

$$I(x, y, t) = I(x + u, y + v, t + 1), \tag{1}$$

where u and v are the translation values of the pixel (x, y) at time $t+1$. To use this assumption, we take the Taylor expansion of $I(x + u, y + v, t + 1)$ and truncate it, such that

$$I(x + u, y + v, t + 1) \approx I(x, y, t) + I_x \cdot u + I_y \cdot v + I_t, \tag{2}$$

where I_α represents the gradient in the dimension α (that is, x, y, and t). Therefore,

$$I(x + u, y + v, t + 1) - I(x, y, t) \approx I_x \cdot u + I_y \cdot v + I_t. \tag{3}$$

And, by (1)

$$0 \approx \nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + I_t \tag{4}$$

The problem is that we have one equation (4) with two unknowns (u and v). We can get more equations by assuming that nearby pixels in the neighborhood, $\mathcal{N}_p$, of the pixel p move with the same deformation vector, (u, v). Let $p_i \in \mathcal{N}_p$, with the numeric index $i \in [1, |\mathcal{N}_p|]$, be a way of naming the pixels in $\mathcal{N}_p$. (✎ You can experiment with different sizes of the neighborhood, but a good starting point can be 15×15, so you will have $i \in [1, 225]$.) Thus, we have a set of equations, for every p_i,

$$0 = I_t(p_i) + \nabla I(p_i) \cdot \begin{bmatrix} u \\ v \end{bmatrix}, \tag{5}$$

which translates into matrix form as

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_N) & I_y(p_N) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ \vdots \\ I_t(p_N) \end{bmatrix}, \tag{6}$$

where $N = |\mathcal{N}_p|$ is the number of pixels in the neighborhood of p. As we did in the past project, we can solve this system with least-squares. By abbreviating (6) as $Ad = b$, we can solve it

$$(A^T A)d = A^T b, \tag{7}$$

$$\begin{bmatrix} I_x(p_1)I_x(p_1) + \cdots I_x(p_N)I_x(p_N) & I_x(p_1)I_y(p_1) + \cdots I_x(p_N)I_y(p_N) \\ I_x(p_1)I_y(p_1) + \cdots I_x(p_N)I_y(p_N) & I_y(p_1)I_y(p_1) + \cdots I_y(p_N)I_y(p_N) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_x(p_1)I_t(p_1) + \cdots I_x(p_N)I_t(p_N) \\ I_y(p_1)I_t(p_1) + \cdots I_y(p_N)I_t(p_N) \end{bmatrix}, \tag{8}$$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}. \tag{9}$$

You can solve for u and v in (9) by inverting the 2×2 matrix and multiplying by the left in both sides. Then, you will have the optical flow of the keypoint p. You will need to repeat this process for every keypoint you found in § 3. You can use the OpenCV interpolation functions to lookup the points in the next frame.

You are advised to discard any keypoints if their predicted locations go out of frame, or if they are too close to the borders. Since you will be using a simple implementation of SfM (see § 5) your tracks (the keypoints positions over time) need to exist in all frames. That means that you won't be able to recover full 3D views of objects.

5 Structure from Motion

You will use the discovered tracks of your keypoints as input to the SfM algorithm. You will use an affine (you can use a projective if you want better results, see § 7) SfM as introduced by Tomasi and Kanade [4] (see Section 3.4 of the paper for the overview of the algorithm). You will need a sufficient amount of tracks (that remain after the pruning of the KLT § 4) to perform the factorization on the dense matrix (you can use the SVD algorithm from OpenCV or other library). ✎ Explain how do you know you have enough points to perform your reconstruction based on your type of camera model.

In order to eliminate the affine ambiguity (i.e., the metric constraint) by discovering the transformation QQ^T (eq. 16 in Tomasi and Kanade [4] paper), we will assume $L = QQ^T$ for easiness, and solve for L using least squares (as discussed in class). Finally, you need to compute the Cholesky decomposition of $L = QQ^T$ to recover Q. You can find the details in Section 2.3 of Morita and Kanade's [5] work.

You need to plot the predicted 3D locations of your reconstruction. ✎ For your report you need to select at least three viewpoints of your reconstruction that shows the shape of your object. Additionally, for the selected viewpoints, plot the path of the cameras for each frame. ✎ Explain how you recover the position of the camera from each frame transformation matrix.

I have updated the docker to include `meshlab` and `viz`, both are tools that you can use to visualize 3D point clouds. There are several examples on the use of `meshlab` within OpenCV samples.[1] Another tool that is available inside OpenCV is `viz` from the VTK library. However, to the best of my knowledge, `viz` bindings only work for C++. If you want to use other visualization tool you need to install it through your `Makefile`.

[1] E.g., https://github.com/opencv/opencv/blob/master/samples/python/stereo_match.py.

6 Experiments

Evaluate your method on the data you gathered at the beginning (see § 2). Perform several experiments tweaking your parameters and evaluating different configurations.

You need to show your work (that is, show frames of the keypoints, then from the tracks, and from the point cloud and camera positions). Additionally, output a `.ply` file that is a dump of the 3D points and color of each point to be rendered in `meshlab`. Check OpenCV samples[2] to get an quick idea of the `.ply` file structure.

7 Extras

If you are eager to do more and bring your method to a next level try the following:

- Use a projective transformation instead of an affine one (+10%).
- Use pyramids into the KLT tracker to improve the tracking on sequences with large frame-to-frame variations (+10%).

Both extras should be demonstrated in experiments and in the report to get the extra credit.

8 Evaluation

Your grade will be defined by the following aspects:

1. KLT tracking 40%
2. SfM 40%
3. Overall report 20%

Each item corresponds to the questions and requirements defined in the previous sections. The overall report, point 3, refers to the evaluation of the details presented in your report, your way of presenting results, explanation and use of concepts and theory, references, etc. **Your English usage won't be graded**, but your ability to present your results, ideas, and how the are supported will be. Each other point will be evaluated according to the completeness and correctness of the requested items.

You will be eligible for extra points, defined in § 7, if you completed the required assignment, and you have an explanation of your extra development in your report.

9 Submission

You need to create a folder named `p3-XX-YY` where `XX` and `YY` are the RAs of the members of your team. Note that the RAs **must** be sorted.

Your submission must have the following subfolders:

- `input`: a directory containing the input assets (images, videos or other data) supplied with the project. **Store the videos used to produce the experiments here, if they are small enough.** For any other asset or big video file, setup your `Makefile` to automatically download them from a public server or repository.
- `output`: a directory where your application should produce all the generated files (otherwise stated in the problem). This directory should be empty.
- `my-output`: use this directory to show your previous results (in case you want to show intermediary work).
- `src`: a directory containing all your source code. You only need to submit files that are not derived from other files or through compilation. In case some processing is needed, prefer to submit a script that does that instead of submitting the files.

[2] E.g., https://github.com/opencv/opencv/blob/master/samples/python/stereo_match.py.

- `Makefile`: a makefile that executes your code through the docker image. An image is already built and available for use (`adnrv/opencv` at the docker hub registry). The code will be executed through a standard call to `make`, so other dependencies must be provided by you under that constraints.
- `report.pdf`: a PDF file that shows all your work for the given project, including images and other outputs needed to explain and convey your work. When needed include explanations to the questions given in the project.

The principal folder must be zipped into `p3-XX-YY.zip` and submitted through the Moodle website. No other files will be accepted. **Note that upon unzipping your file, the original folder `p3-XX-YY.zip` must be created.** That is, do not zip the contents, but rather the folder.

10 Notes

- **Your videos shouldn't be too big or your processing time will be too high.**
- Note that there are several implementations that you can find in the internet. This project is for **you to implement** the algorithms. Thus, do not submit code from others. And if you re-use code from someone for a non-restricted part, disclose it in your report and code.
- You are not allowed to use any prebuilt function for the KLT tracker and for the SfM reconstruction. You can use other OpenCV functions to compute the matrix factorization and decompositions, keypoint extractors, pyramids, etc.
- All the submissions must be self contained and must be executable in a Linux environment. Specifically, your code must execute in the docker image `adnrv/opencv`, available at docker hub (`https://hub.docker.com/r/adnrv/opencv/`).
- It is your responsibility to make sure your code compiles and executes correctly. No effort will be made to run your code besides executing `make` inside a docker.
- You must program in Python 3.6 (or higher) or in C/C++ with `gcc` version 6.2.0, using OpenCV 3.2.0. All available within the image. If you need to install other packages you must do so within your `Makefile` as automatic prerequisites.
- Check the `Makefile` provided with the project zero. It will be used to automatically execute your code. You must pass the full path of your project folder: `make SOURCE_DIR=$(realpath ./pA-XX-YY)`.

References

[1] J. Shi *et al.*, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, IEEE, 1994, pp. 593–600.

[2] C. Tomasi and T. Kanade, "Detection and tracking of feature points," *Tech. Rep., Carnegie Mellon University Technical Report CMU-CS-91-132*, 1991.

[3] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision," 1981.

[4] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: A factorization method," *International Journal of Computer Vision*, vol. 9, no. 2, pp. 137–154, 1992.

[5] T. Morita and T. Kanade, "A sequential factorization method for recovering shape and motion from image streams," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 8, pp. 858–867, 1997.