Structure from motion

Ilia Nechaev

08 November 2024

SFM pipeline

1. Retrieving key points from the first two images
2. Matching key points between first two images
3. Using epipolar geometry to find second camera position and orientation (on this step we lose scale)
4. Triangulation to find 3D points
5. Retrieving key points from the next image
6. Matching key points between the next image and the previous images
7. Using resection to find the next camera position and orientation
8. Triangulation to find 3D points from the next image
9. Repeat steps 5–8 until the end of the images
10. Bundle adjustment to refine the camera positions and 3D points (after each iteration, after some iterations, or at the end)

What will be today?

1. Camera projection (with math)
2. Triangulation (with math)
3. Resection (with math)
4. Key points (just a scratch)
5. Key points matching (just a scratch)

Preparations. Homogenious coordinates

In simple words: homogeneous coordinates are a way of representing points and vectors in space that extends the standard coordinates by adding an extra dimension.
Using homogeneous are helpful in following cases:

1. Affine transformations can be represented by a single matrix.
2. Multiplying point's coordinates by a scalar doesn't change the point.

$$\begin{bmatrix} x \\ y \end{bmatrix} \leftrightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \leftrightarrow \begin{bmatrix} kx \\ ky \\ k \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \leftrightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \leftrightarrow \begin{bmatrix} kx \\ ky \\ kz \\ k \end{bmatrix}$$

Camera projection

From previous lecture we know that camera projection is described by the following formula:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x \cdot (x' \cdot (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x' y' + p_2 \cdot (r^2 + 2x'^2)) + c_x \\ f_y \cdot (y' \cdot (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1 \cdot (r^2 + 2y'^2) + 2p_2 x' y') + c_y \end{bmatrix}$$

Where $r^2 = x'^2 + y'^2$ and $x' = \frac{x}{z}, y' = \frac{y}{z}$

But here we made one assumption: origin of the world coordinate system is in the center of the camera and orientation of the camera is the same as the world coordinate system.

Camera projection with different camera orientation and position

First of all, in this lecture when we talk about images we talk about undistorted images. So in the simplest case the projection formula will be:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x \cdot \frac{X}{Z} + c_x \\ f_y \cdot \frac{Y}{Z} + c_y \end{bmatrix}$$

No we need to add camera orientation and position to the formula. If we have camera orientation and position:

1. $\hat{R} \in \mathbb{R}^{3\times3}$ - rotation matrix of camera in world coordinate system.
2. $\hat{t} \in \mathbb{R}^{3\times1}$ - camera position in world coordinate system.

Then we can introduce camera matrix P:

$$P = K \cdot [\hat{R}^T | -\hat{R}^T \cdot \hat{t}]$$

Where $K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$ is the camera matrix.

How to use matrix P?

If we have 3D point $\begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$ in world coordinate system, then we can project it to the image plane using the following formula:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = P \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = K \cdot [\hat{R}^T | -\hat{R}^T \cdot \hat{t}] \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = K \cdot \left(\hat{R}^T \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - \hat{R}^T \cdot \hat{t} \right)$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{u}{w} \\ \frac{v}{w} \end{bmatrix}$$

Triangulation

Triangulation is the process of finding the 3D point from two 2D points and two camera matrices.

Assume we have two camera matrices: P_1 and P_2; and two 2D points in homogeneous coordinates: $x_1 = \begin{bmatrix} u_1 & v_1 & 1 \end{bmatrix}^T$ and $x_2 = \begin{bmatrix} u_2 & v_2 & 1 \end{bmatrix}^T$ on each image.

Then we can find the 3D point $\hat{X} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$ using the following idea:

$$x_1 \times (P_1 \cdot \hat{X}) = 0$$
$$x_2 \times (P_2 \cdot \hat{X}) = 0$$

where $\times$ is the cross product in homogeneous coordinates.

$$a \times b = \begin{bmatrix} a_2 \cdot b_3 - a_3 \cdot b_2 \\ a_3 \cdot b_1 - a_1 \cdot b_3 \\ a_1 \cdot b_2 - a_2 \cdot b_1 \end{bmatrix}$$

Important that collinear vectors give zero cross product, that's why we need homogeneous coordinates.

Triangulation. Preparations for SVD.

Using the results of previous slide we can write the following systems of equations:

$$\begin{cases} -u_1 \cdot (p_1^{3T}\hat{X}) + (p_1^{1T}\hat{X}) = 0 \\ v_1 \cdot (p_1^{3T}\hat{X}) - (p_1^{2T}\hat{X}) = 0 \\ u_1 \cdot (p_1^{2T}\hat{X}) - v_1(p_1^{1T}\hat{X}) = 0 \end{cases}$$

$$\begin{cases} -u_2 \cdot (p_2^{3T}\hat{X}) + (p_2^{1T}\hat{X}) = 0 \\ v_2 \cdot (p_2^{3T}\hat{X}) - (p_2^{2T}\hat{X}) = 0 \\ u_2 \cdot (p_2^{2T}\hat{X}) - v_2(p_2^{1T}\hat{X}) = 0 \end{cases}$$

Here p_j^{iT} is the i-th row of the matrix P_j. We can notice that in each system only 2 equations are independent, so we can take 2 equations from each system and solve the system of 4 equations with 4 unknowns.

$$\begin{cases} -u_1 \cdot (p_1^{3T}\hat{X}) + (p_1^{1T}\hat{X}) = 0 \\ v_1 \cdot (p_1^{3T}\hat{X}) - (p_1^{2T}\hat{X}) = 0 \\ -u_2 \cdot (p_2^{3T}\hat{X}) + (p_2^{1T}\hat{X}) = 0 \\ v_2 \cdot (p_2^{3T}\hat{X}) - (p_2^{2T}\hat{X}) = 0 \end{cases}$$

In matrix form we can represent this system as:

$$A = \begin{bmatrix} -u_1 \cdot p_1^{3T} + p_1^{1T} \\ v_1 \cdot p_1^{3T} - p_1^{2T} \\ -u_2 \cdot p_2^{3T} + p_2^{1T} \\ v_2 \cdot p_2^{3T} - p_2^{2T} \end{bmatrix} = \begin{bmatrix} -u_1 \cdot P_{1(3,1)} + P_{1(1,1)} & -u_1 \cdot P_{1(3,2)} + P_{1(1,2)} & -u_1 \cdot P_{1(3,3)} + P_{1(1,3)} & -u_1 \cdot P_{1(3,4)} - P_{1(1,4)} \\ v_1 \cdot P_{1(3,1)} - P_{1(2,1)} & v_1 \cdot P_{1(3,2)} - P_{1(2,2)} & v_1 \cdot P_{1(3,3)} - P_{1(2,3)} & v_1 \cdot P_{1(3,4)} - P_{1(2,4)} \\ -u_2 \cdot P_{2(3,1)} + P_{2(1,1)} & -u_2 \cdot P_{2(3,2)} + P_{2(1,2)} & -u_2 \cdot P_{2(3,3)} + P_{2(1,3)} & -u_2 \cdot P_{2(3,4)} - P_{2(1,4)} \\ v_2 \cdot P_{2(3,1)} - P_{2(2,1)} & v_2 \cdot P_{2(3,2)} - P_{2(2,2)} & v_2 \cdot P_{2(3,3)} - P_{2(2,3)} & v_2 \cdot P_{2(3,4)} - P_{2(2,4)} \end{bmatrix}$$

$$A \cdot \hat{X} = 0$$

We got the linear system of homogeneous equations, so we can solve it using SVD.

Trianfulation. SVD

After we got the matrix the system of equations $A \cdot \hat{X} = 0$, we can solve it using SVD.

$$A = U \cdot \Sigma \cdot V^T$$

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 \\ 0 & 0 & 0 & \sigma_4 \end{bmatrix}$$

If $\sigma_4 = 0$, then the exact solution is the last column of V (or last row of V^T). But usually $\sigma_4 \neq 0$, due to noise in the system. However, this doesn't change the fact that the solution is the last column of V. But in this case, the solution is the least squares solution: $\min ||A \cdot \hat{X}||$.

Resection

Resection is the process of finding the camera's position $[R|t]$ using the 3D points and 2D points.
Assume we have 3D points $\hat{X}_i = \begin{bmatrix} X_i & Y_i & Z_i & 1 \end{bmatrix}^T$ and 2D points $x_i = \begin{bmatrix} u_i & v_i & 1 \end{bmatrix}^T$ on the image. Then we can write the following system of equations:

$$\begin{cases} -u_i \cdot (p^{3T} \cdot \hat{X}_i) + (p^{1T} \hat{X}_i) = 0 \\ v_i \cdot (p^{3T} \cdot \hat{X}_i) - (p^{2T} \hat{X}_i) = 0 \end{cases}$$

Looks familiar, right? We can solve this system using SVD.
But in this case, we need to find the camera matrix P not the $\hat{X}$. So we need to change a matrix A a bit.

$$A = \begin{bmatrix} \hat{X}_i^T & 0 & u_i \cdot -\hat{X}_i^T \\ 0 & -\hat{X}_i^T & v_i \cdot \hat{X}_i^T \end{bmatrix} = \begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_i \cdot X_i & -u_i \cdot Y_i & -u_i \cdot Z_i & -u_i \\ 0 & 0 & 0 & 0 & -X_i & -Y_i & -Z_i & -1 & v_i \cdot X_i & v_i \cdot Y_i & v_i \cdot Z_i & v_i \end{bmatrix}$$

Then the system can be represented as:

$$A \cdot \begin{pmatrix} p^1 \\ p^2 \\ p^3 \end{pmatrix} = 0$$

Where p^{iT} is the i-th row of the matrix P. In other words, second multiplier is the vectorized matrix P:

$$(p_{1,1}, p_{1,2}, p_{1,3}, p_{1,4}, p_{2,1}, p_{2,2}, p_{2,3}, p_{2,4}, p_{3,1}, p_{3,2}, p_{3,3}, p_{3,4})^T$$

Matrix P is 3×4 matrix, so it has 12 unknowns. So we need at least 6 3D points to solve this system. But adding more points will increase the accuracy of the solution.

Resection. SVD

So as we mentioned above more points will increase the accuracy of the solution. Let's take n points, so we have $2n$ equations, $A \in \mathbb{R}^{2n \times 12}$
After we got the matrix representation for the system of equations $A \cdot \begin{pmatrix} p^1 & p^2 & p^3 \end{pmatrix}^T = 0$, we can solve it using SVD.

$$A = U \cdot \Sigma \cdot V^T$$

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_{12} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

Once again if $\sigma_{12} = 0$, then the exact solution is the last column of V (or last row of V^T). Otherwise, the same column represents the least squares solution.
Be aware that this is only main part of algorithm, in practice you need a lot of optimizations to make it work (like using RANSAC to get rid of outliers).

Resection. Retreiving $\hat{R}$ and $\hat{t}$

So after SVD we got the matrix P. What now? We need to retrieve the rotation matrix $\hat{R}$ and the translation vector $\hat{t}$ from the matrix P.

$$P = K \cdot [\hat{R}^T | -\hat{R}^T \cdot \hat{t}]$$

Matrix K is the camera matrix, we can get it using RQ decomposition (not covered in this lecture, you can use cv2.decomposeProjectionMatrix). Important that matrix K is invertible, that's mean that:

$$K^{-1} \cdot P = [\hat{R}^T | -\hat{R}^T \cdot \hat{t}]$$

Here's we met standard notation:

$$R = \hat{R}^T \text{ - rotation matrix}$$

$$t = -\hat{R}^T \cdot \hat{t} \text{ - translation vector}$$

This notation is widely used, you will probably see it in the future especially in OpenCV docs. Be aware that OpenCV functions like solvePnP or triangulatePoints return R and t, not $\hat{R}$ and $\hat{t}$.

Key points

Key points in terms of computer vision are points that are easy to detect and match between images. For each key point, we can compute its position on image and its descriptor that will help us to match key points between images.
Methods for key points detection:

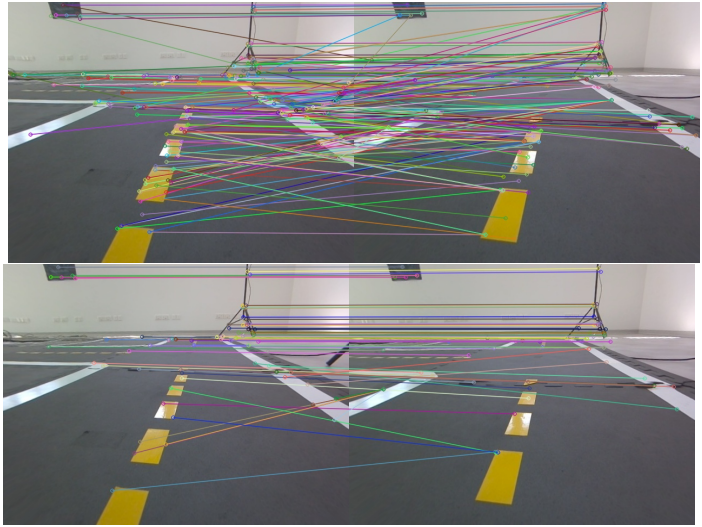
1. FAST
2. SIFT
3. ORB
4. Deep learning-based methods
5. A lot of other

Key points matching

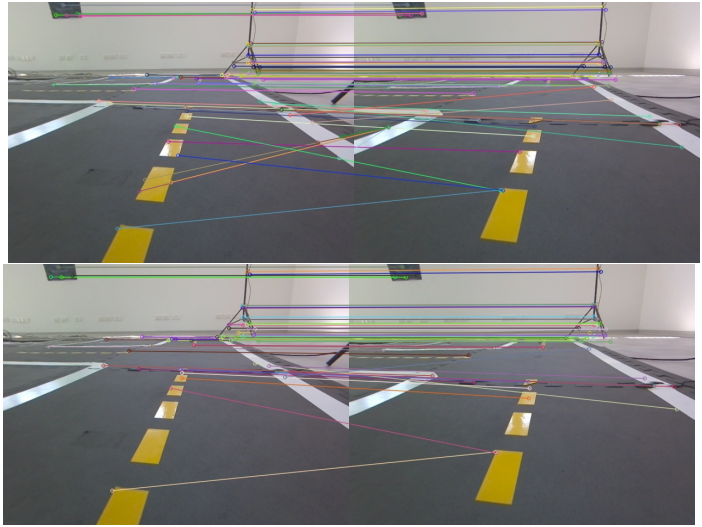
After we detected key points on the images, we need to match them. The simplest way to match key points is to use the nearest neighbor search.
But in this case, we can have a lot of false matches. To reduce the number of false matches we can use a few methods:

1. k-Ratio test
2. Left-right consistency check
3. Cluster matching
4. Estimation of homography matrix
5. Estimation of fundamental matrix
6. Deep learning-based methods

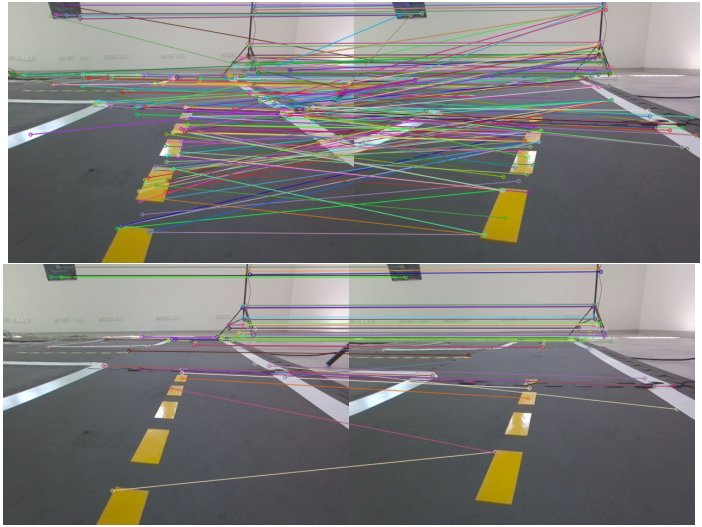
Checks for key points matching



Checks for key points matching



Checks for key points matching



References

1. Multiple View Geometry in Computer Vision by Richard Hartley and Andrew Zisserman

   https://www.r-5.org/files/books/computers/algo-list/image-processing/vision/Richard_Hartley_Andrew_Zisserman-Multiple_View_Geometry_in_Computer_Vision-EN.pdf

2. OpenCV documentation:
   1. Camera Calibration and 3D Reconstruction:
      https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html
   2. Perspective-n-Point (PnP) pose computation:
      https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html
   3. Triangulation:
      https://docs.opencv.org/4.x/d0/dbd/group__triangulation.html
   4. Structure From Motion:
      https://docs.opencv.org/4.x/d8/d8c/group__sfm.html