M4 - Lab 5: Structure from Motion

Sergio Montoya, Laia Albors, Ibrar Malik, Adam Szummer

Pompeu Fabra University
{ sergio.montoyadepaco, laia.albors, ibrar.malik, adam.szummer
}@e-campus.uab.cat

1 Introduction

The goal of this week is the 3D reconstruction from N uncalibrated cameras with a stratified method, by designing and applying a Structure from Motion (SfM) pipeline. Therefore, we will understand how the pipeline of the structure-from-motion works and we will apply it to a set of images of a facade.

2 3D reconstruction

3D reconstruction consists of reconstructing the 3D geometry of the world based on 2D images obtained from the same scene. We follow the stratified reconstruction defined in section 10.8 of the book Multiple View Geometry in Computer Vision (MVG) (1). The stratified reconstruction consists of obtaining a projective reconstruction of the scene and cameras, which will be first rectified into an affine reconstruction by locating the plane at infinity and rectified with a metric reconstruction based on orthogonality conditions. The structure from motion pipeline that we implement is as follows:

1. Projective reconstruction: Compute a pair of projective cameras based on the fundamental matrix estimated between a pair of initial images. Once the cameras have been estimated, point correspondences are triangulated to obtain a projective reconstruction of the scene. This process is described in section 3.
2. Affine rectification: Define the plane at infinity with three different 3D vanishing points. The plane at infinity under a projective reconstruction does not need to be at its canonical position $(0, 0, 0, 1)^T$. By defining a 3D homography that moves the plane to its canonical position, we can upgrade the reconstruction to an affine reconstruction of the scene where the 3D parallel lines that define vanishing points become parallel. This process is described in section 4.
3. Metric rectification: Define constraints on the image of the absolute conic by defining orthogonality constraints and assumptions on the intrinsic camera matrix. This will result in an affine transformation that recovers metric properties of the scene such as orthogonality. This process is described in section 5.
4. Bundle adjustment: We can perform a non-linear optimization by minimizing a reprojectiong error with respect to camera parameters and 3D points. Given a set of images with corresponding matching points that have been triangulated, we can define a minimization problem that simultaneously refines triangulated 3D point coordinates and camera parameters. This process is defined in section 6.

3 Computation of projective cameras

First, we compute the fundamental matrix between pairs of views with the algorithms that we have seen in previous assignments. For that, we first extract SIFT (2) features, establish matches between pairs of images by filtering correspondences with the ratio test, and based on those correspondences compute the fundamental matrix. By knowing the fundamental matrix F between pair of images and not knowing the intrinsic camera matrix K (cannot compute the essential matrix), the scene of the geometry can be reconstructed up to a projective transformation. This is due to the fact that a pair of cameras $(\mathrm{P}, \mathrm{P}')$ and $(\mathrm{PH}, \mathrm{P}'\mathrm{H})$, where H is a projective transformation, are related by the same fundamental matrix F. As we have that $\mathrm{PX} = (\mathrm{PH})(\mathrm{H}^1\mathrm{X})$, we can see that camera PH with world points transformed as $\mathrm{X}_p = \mathrm{H}^{-1}\mathrm{X}$ give rise to the same projected points with the same fundamental matrix.

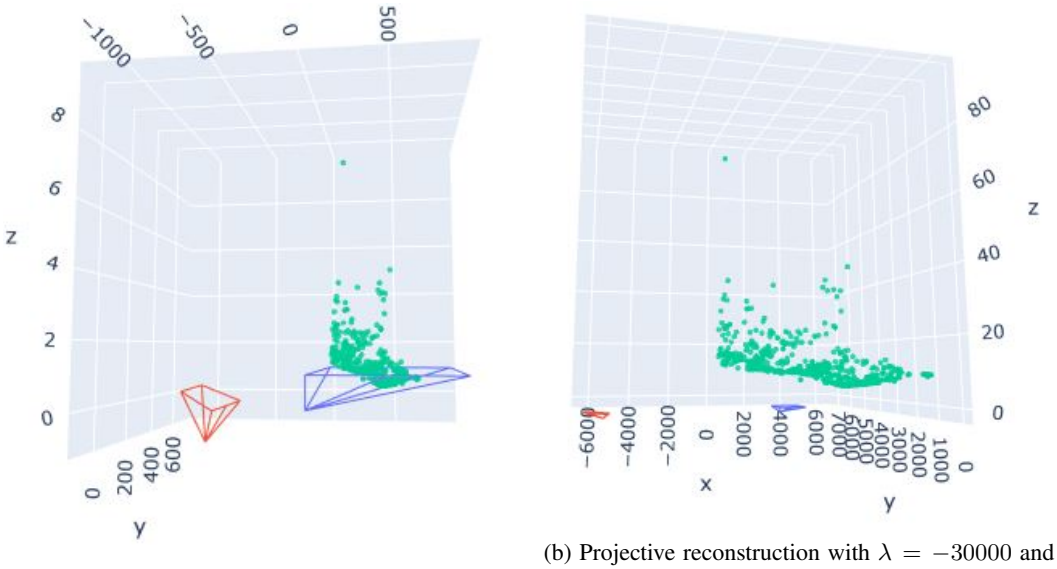
As described in section 9.5.3 of MVG, given F, we can define a pair of cameras related by this fundamental matrix as:

$$\mathrm{P}_1 = [\mathrm{I}|\mathbf{0}] \quad \mathrm{P}_2 = [\mathrm{SF}|\mathbf{e}'] \tag{1}$$

where S is any skew-symmetric matrix and $\mathbf{e}'$ is the epipole in the second image. A good choice for S is $\mathrm{S} = [\mathbf{e}']_\times$. The general formula for a pair of camera matrices in canonical form with fundamental matrix F is given by:

$$\mathrm{P}_1 = [\mathrm{I}|\mathbf{0}] \quad \mathrm{P}_2 = [[\mathbf{e}']_\times \mathrm{F} + \mathbf{e}'\mathbf{v}^T|\lambda\mathbf{e}'] \tag{2}$$

where $\mathbf{v}$ is any 3-vector, and λ is a non-zero scalar. In Figure 1, we can observe two projective reconstructions with different λ.



(a) Projective reconstruction with $\lambda = -3000$ and $\mathbf{v} = (5, 5, 5)^T$.

(b) Projective reconstruction with $\lambda = -30000$ and $\mathbf{v} = (5, 5, 5)^T$. We can observe larger translation between cameras in this place.

Fig. 1: Projective reconstruction with cameras defined as in Equation 2.

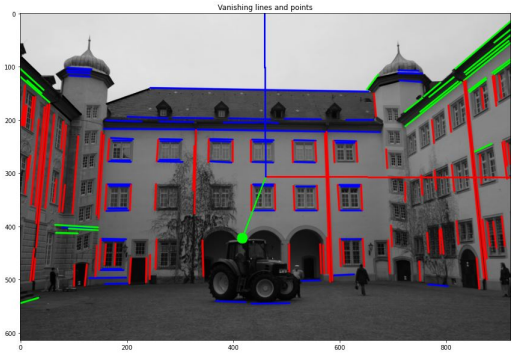
4 Affine rectification

In order to perform an affine reconstruction, we need to locate the plane at infinity, which under a projective reconstruction of the scene does not need to be at its canonical position $(0, 0, 0, 1)^T$. We have tried different approaches to determine the plane at infinity. Once we determine the plane at infinity π_∞, we have to define a 3D homography that moves the plane to its canonical position. Once the plane at infinity is in its canonical position, the geometry of the scene is recovered up to an affine rectification. Considering the way a projective transformation acts on planes, we want to find H such that $\mathrm{H}^{-T}\pi_\infty = (0, 0, 0, 1)^T$. The 3D homography H that performs this rectification is:

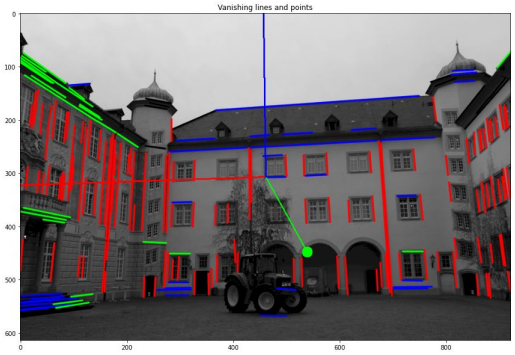
$$\mathrm{H} = \begin{bmatrix} \mathrm{I}_{3x3} | \mathbf{0} \\ \pi_\infty^T \end{bmatrix} \tag{3}$$

As a projective transformation that leaves the plane at infinity is an affine transformation, this reconstruction differs by an affine transformation from the true reconstruction, resulting in an affine reconstruction. Under this type of reconstruction, parallel lines are parallel in the scene. We have explored three main

methods for computing 3D vanishing points. All of them rely on the method that is provided for clustering lines corresponding to different sets of parallel lines in the scene. The provided method (3) relies on the focal length and principal points of the camera, which is a bit unfair as we are following an autocalibration approach, we are reconstructing the scene geometry with no camera information. An example of vanishing point estimation based on parallel lines in the scene is provided in Figure 2.



(a) Vanishing lines and vanishing points of the first image.



(b) Vanishing lines and vanishing points of the second image.

Fig. 2: Estimated vanishing lines and points using the provided method (3). We have drawn lines from the center of the image to the vanishing point to provide an intuition of where the vanishing point is. We can see how only one of the vanishing points are in the image frame.

The methods that we have used to locate 3D vanishing points to define the plane at infinity are:

1. Triangulate corresponding vanishing points in the images to estimate 3D vanishing points defining the plane at infinity. The problem that we had with this approach is that the noise on the estimation of corresponding vanishing points affects the estimation of the plane at infinity greatly. Specifically, we can see in Figure 3 that the plane at infinity is poorly estimated, the geometry of the scene is not what we expect, and results in a poor reconstruction.
2. The second method is very similar to the first one, but it first corrects the position of vanishing points so that 3D points are correctly triangulated. For every vanishing point, we compute the corresponding epipolar line using the previously estimated fundamental matrix. Then, we select one of the parallel lines that intersect at that vanishing point. By intersecting this parallel line with the epipolar line of the corresponding vanishing point, we obtained another point that can be perfectly triangulated. We follow this approach to refine the location of the three vanishing points and obtain a good affine reconstruction of the scene.
3. By intersecting 3D lines, we can estimate vanishing points in 3D. For each vanishing point to be estimated, we select a pair of parallel lines for which we know 2D point correspondences in both images. Then, we can triangulate two points for each line, where each pair of points define a line in 3D. As line estimates are noisy, lines will not typically intersect in 3D. By adapting the notation of lines with Plücker line coordinates (MCV section 3.2.2), we have that these lines will intersect if and only if the determinant of the matrix produced by stacking these 4 points is equal to 0. If we denote such points as $A, \hat{A}, B, \hat{B}$, then we need that $det[A, \hat{A}, B, \hat{B}] = 0$. We decide to find the closest matrix in Frobenius norm to this matrix, by setting the last eigenvalue to 0 in the resulting SVD, so that the determinant of the matrix is 0. From this matrix, we extract the refined points that intersect. By solving a simple system of equations we can determine the intersection point of these 3D lines. By applying this approach to all 3D points, we can get the estimated 3D points. We observe that we get similar results to the first approach, producing noisy 3D vanishing points which leads to a poor estimation of the plane at infinity and a poor reconstruction of the scene.

To use the last two approaches for estimating 3D vanishing points, we had to locate salient points located by SIFT that are close to line segments that are part of the parallel lines. To do so, we have gone through

each set of parallel lines and located those points by iterating over all the points. We need this so that we can establish correspondences of lines and points between a pair of images.

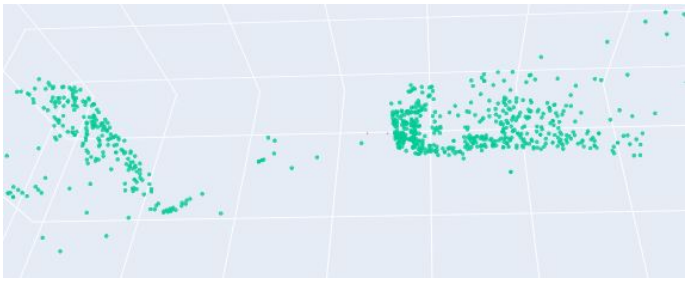
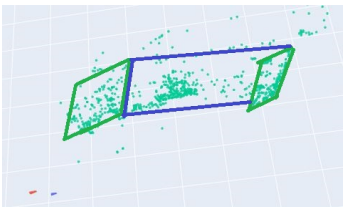


Fig. 3: Wrong affine reconstruction resulting from poor estimation of the plane at infinity.

In Figure 4, we can see an affine reconstruction of the scene following the second approach for estimating 3D vanishing points. In the figure, we can clearly see the three facades of the scene. In Figure 4c, we have approximately drawn the three facades of the scene. Due to the affine reconstruction, we have recovered the affine properties of the scene such as line parallelism. We can see how the 3D parallel lines that defined vanishing points are now parallel. For example, the left and right facades are approximately parallel. However, as we have not recovered metric properties, the orthogonality of the scene has not been recovered and the three facades are not perpendicular as it should be. In the provided figures, we have filtered points that are behind both cameras, as they will correspond to match outliers.

(a) Affine rectification from a frontal view, near from both camera perspective.

(b) Affine rectification from a top view.



(c) Affine rectification from a top view. Approximate planes of facades are drawn in green and blue.

Fig. 4: Affine reconstruction. The first camera in canonical position is the red camera and the second camera is the blue one.

In order to make sure that the error of the reconstruction does not increase in any step of the reconstruction due to any error, we implement the reprojection error, which is defined as:

$$\text{Reprojection error} = ||\mathrm{P}\mathrm{X} - x||_2 + ||\mathrm{P}'\mathrm{X} - x'||_2 \tag{4}$$

where P and P' are the camera matrices, X is the matrix of 3D observations, and x and x' are the projected points in the image frame. The reprojection error of the projective reconstruction is of approximately $6.38e-8$, which remains the same after the affine rectification of the scene points and cameras.

By denoting $\{\mathrm{P}^i, \mathrm{X}^i\}$ the reconstructed projective camera matrices and projectively reconstructed points, we apply a rectifying homography H as $\{\mathrm{P}^i\mathrm{H}, \mathrm{H}^{-1}\mathrm{X}^i\}$. As we are applying H^{-1} to 3D points, we want the affinely rectifying homography H_{aff} to be the inverse of Equation 5, which is:

$$\mathrm{H}_{\text{aff}} = \begin{bmatrix} \mathrm{I}_{3x3} | \mathbf{0} \\ -\pi_\infty^T \end{bmatrix} \tag{5}$$

5 Metric rectification

Once affinely rectified points are obtained after identifying the plane at infinity we proceed to perform a metric reconstruction by identifying absolute conic. The next step is to find such affine transformation that projects the image of the absolute conic to an euclidean frame. Then, we can say that the geometry of the scene is recovered up to a similarity transformation. We denote the image of the absolute conic in the affinely reconstructed scene as w. Then the affine reconstruction can be upgraded into a metric reconstruction by applying a 3D homography as:

$$\mathrm{H}_{met} = \begin{bmatrix} \mathrm{A}^{-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{6}$$

If we denote the camera intrinsic matrix as K, the image of the absolute conic is written as:

$$w = \mathrm{K}^{-T}\mathrm{K}^{-1}$$

By making the assumption that both cameras have the same intrinsic camera parameters K, the skew for K is equal to 0 and pixels are square we have 2 constraints:

$$w^{11} = w^{22} \quad \text{and} \quad w^{12} = 0$$

and by orthogonality of lines u,w and z we have three extra constraints defined as:

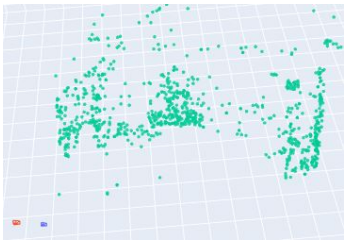
$$u^T w v = 0, u^T w z = 0, v^T w z = 0$$

Finally, by given constraints we determine w and the homography estimation is calculated by defining matrix A by rearranging and solving equation 7.

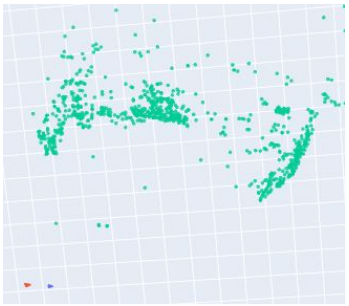
$$\mathrm{A}\mathrm{A}^T = (\mathrm{M}^T w \mathrm{M})^{-1} \tag{7}$$

where M is the first 3 by 3 matrix of camera parameter P.

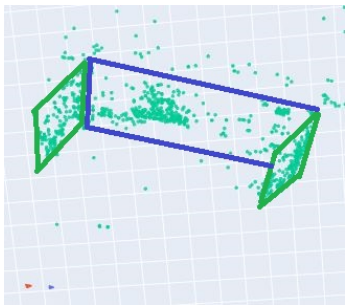
The particular value A that satisfies the equation is found by *Cholesky* factorization that allows us to construct the needed H matrix that transforms the conic to euclidean frame, hence giving the metric reconstructed scene as shown in Figure 5. We then verify that reprojection error is kept at $6.38e-8$. By obtaining the transformed conic in euclidean frame we ensure that metric properties such as orthogonality are recovered.



(a) Metric rectification from a frontal view, near from both camera perspective.



(b) Metric rectification from a top view.



(c) Metric rectification from a top view. Approximate planes of facades are drawn in green and blue.

Fig. 5: Metric reconstruction. The first camera in canonical position is the red camera and the second camera is the blue one.

6 Bundle adjustment

Once we have an initial estimate of the camera poses and the geometry, we can perform Bundle adjustment to refine these using non-linear optimization. We minimize the reprojection error for all image points. First, on subsection 6.1 we will get the intrinsic and extrinsic parameters from an initial estimation of the camera matrices, which are needed as our optimization problem is non-linear. On subsection 6.2 we further explain the optimization problem, and how we form the Jacobian sparse matrix.

6.1 Camera parameters from projection matrix

From a camera/projection matrix, we want to get the intrinsic and extrinsic parameters. A camera matrix maps points in the so called *world* coordinates to the camera coordinates, by moving the origin to the optical center, applying a rotation to the axes, and the intrinsics matrix enables us to correctly project these points to the image plane. On Equation 8 we can see this decomposition of the P camera matrix, where K is the intrinsics matrix, R the rotation, $\mathbf{C}$ the optical center (in world coordinates) and $\mathbf{t}$ the translation.

$$\mathrm{P} = \mathrm{KR}[\mathrm{I}|-\mathbf{C}] = \mathrm{K}[\mathrm{R}|-\mathrm{R}\mathbf{C}] = \mathrm{K}[\mathrm{R}|\mathbf{t}] \tag{8}$$

Note that the 3x3 left submatrix of P can be decomposed into an upper-triangular matrix K and an orthogonal matrix R. We can get this decomposition from any square matrix using the *RQ* decomposition.

The *RQ* decomposition factorizes our chosen matrix into a product of an upper-triangular and an orthogonal matrix. This decomposition is not unique, as we can change the sign of any column of R and corresponding row of Q and get the same matrix. This won't be much of a problem for us, as we know *a priori* that the intrinsics matrix has a positive diagonal, meaning that the focal length is positive and the image plane is in front of the camera. We will just change the signs of the K columns with a negative diagonal entry, and also the sign of the corresponding R rows. Moreover, we can be reassured that this decomposition will be unique, as the 3x3 KR submatrix is non-singular and thus all values on the diagonal of R will be non-zero.

What about the sign of $\det(\mathrm{R})$? As the intrinsics matrix is correct and consistent with our camera model, we shouldn't change the signs of R. We can take a negative determinant as a sign that the *handedness* of the world and camera coordinate systems are different, as R is performing a reflection. This does not happen in the points we are using.

Finally, getting $\mathbf{t}$ is straight-forward once we have R. Let's say that $P = [\mathrm{M}|\mathbf{p_4}]$, where M is the submatrix we decomposed K and R from, and $\mathbf{p_4}$ is the last column of P. Then $P = \mathrm{M}[\mathrm{I}|\mathrm{M}^{-1}\mathbf{p_4}] = \mathrm{KR}[\mathrm{I}|-\mathbf{C}]$ as we saw in Equation 8. Having computed the optical center $\mathbf{C} = \mathrm{M}^{-1}\mathbf{p_4}$, we can find the missing column $\mathbf{t} = -\mathrm{R}\mathbf{C}$.

6.2 Jacobian Sparse matrix

To apply this Bundle Adjustment algorithm, the strategy consists in minimizing the re-projection error (see Equation 9). To do this minimization, the Trust Region Reflective method can be used. This method needs to compute the derivative of the function that we want to minimize (the re-projection error in our case) with respect to the unknowns (P^i and $\mathbf{X_p}$). All this derivatives are gathered in the Jacobian matrix.

$$\min_{\mathrm{P}^i, \mathbf{X_p}} \sum_i \sum_p d(\mathrm{P}^i \mathbf{X_p}, \hat{\mathbf{x}}^i_{\mathbf{p}})^2 \qquad \text{where } \hat{\mathbf{x}}^i_{\mathbf{p}} \text{ is the observed point } p \text{ in image } i. \tag{9}$$

The Jacobian matrix is a matrix that has $m = \#points \cdot \#cameras \cdot 2$ rows (2 equations for each point in each image) and $n = \#cameras \cdot 9 + \#points \cdot 3$ columns (9 unknows for each camera matrix and 3 for each 3D point). As we can see in the visual representation of this Jacobian matrix for this problem, in Figure 6, the matrix is very sparse. This is because the equations

$$d(\mathrm{P}^i \mathbf{X_p}, \hat{\mathbf{x}}^i_{\mathbf{p}})^2 = ([\mathrm{P}^i \mathbf{X_p}]_1 - [\hat{\mathbf{x}}^i_{\mathbf{p}}]_1)^2 + ([\mathrm{P}^i \mathbf{X_p}]_2 - [\hat{\mathbf{x}}^i_{\mathbf{p}}]_2)^2 \qquad \forall i, p \tag{10}$$

where $[\cdot]_1$ and $[\cdot]_2$ are the first and second terms of the 2D vectors respectively, only depend on an specific image i and point p. So, for example, the first two equations

$$d(\mathrm{P}^1\mathbf{X_1}, \hat{\mathbf{x}}_\mathbf{1}^1)^2 = ([\mathrm{P}^1\mathbf{X_1}]_1 - [\hat{\mathbf{x}}_\mathbf{1}^1]_1)^2 + ([\mathrm{P}^1\mathbf{X_1}]_2 - [\hat{\mathbf{x}}_\mathbf{1}^1]_2)^2 \tag{11}$$

only depend on the 3D point $\mathbf{X_1}$ and the camera matrix P^1. This means that their derivative with respect to any other 3D point or camera matrix will be zero. This can be seen in the right image in Figure 6, where the first two rows (corresponding to the first two equations) only have values different from zero in the first 9 columns (corresponding to the 9 entries of the camera matrix P^1) and in the columns 18, 19 and 20 (corresponding to the 3 values of the 3D point $\mathbf{X_1}$).

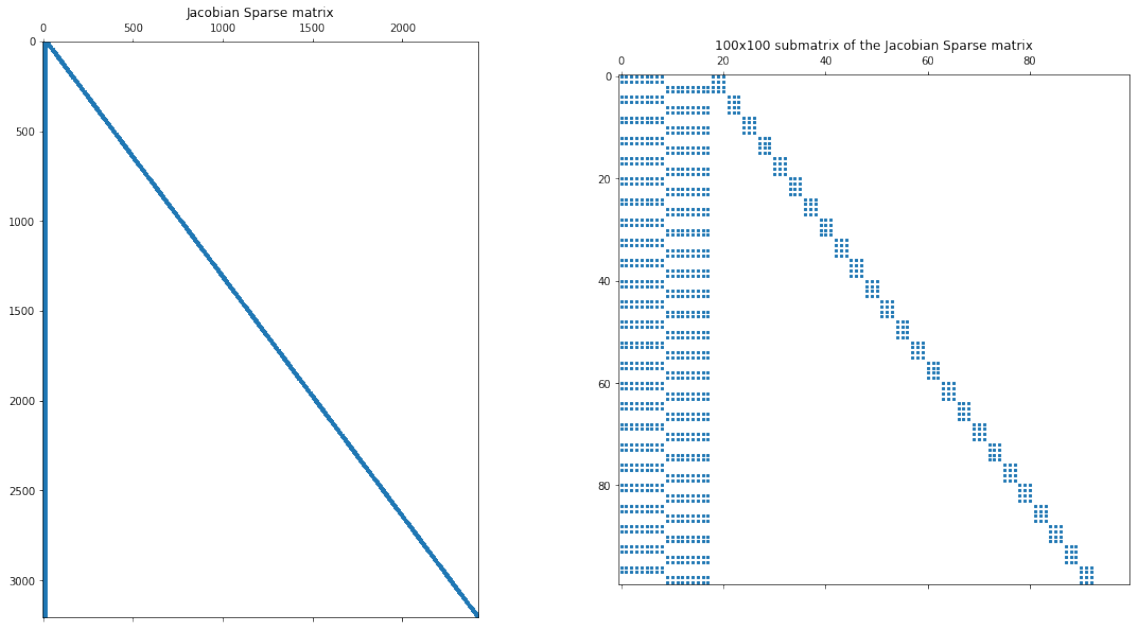


Fig. 6: Visual representation of the Jacobian Sparse matrix of this problem (with 2 cameras and 802 point correspondences between the two images). On the left, we have the whole Jacobian matrix. On the right, we have the first 100 rows and columns, to see the details of the matrix.

In our case we are just defining the sparsity structure of the Jacobian matrix, not all its values. This structure is used in `scipy.optimize.least_squares` function to speed up the computations by using an iterative procedure for finding a solution of the least-squares problem which only requires matrix-vector product evaluations.

6.3 Results

Due to erroneous correspondences detected between the images (i.e., "matching" points between the images that don't actually correspond to the same real-world point), we have 3D points whose projection in an image does not correspond to the 2D point that we have detected. This difference between the detected 2D points and the 3D points projections can be reduced using the so called bundle adjustment algorithm. This algorithm refines the visual reconstruction to produce optimal 3D points and camera matrix parameters. Optimal means that this algorithm looks for the 3D points and camera matrix parameters that minimize a cost function. In this case, we are interested in reducing the distance between the detected 2D points and the 3D points projections, so our cost function is the re-projection error (see Equation 9).

In order to assert the effect of bundle adjustment, we can look at the value of the re-projection error before and after applying it. As we can see in the notebook, we go from an error of $1.72 \cdot 10^9$ to $8.03 \cdot 10^3$. That is, we have reduced the re-projection error by 6 orders of magnitude. We can also assert its

effect qualitatively by looking at the projected 3D points over the image, before and after applying bundle adjustment, and comparing them with the 2D detected points. This is visualized in Figure 7, where we can see that some of the detected correspondences are wrong, so the distance between the 2D detected point (in *white*) and the 3D projection (in *red*) is big. In these cases, the 3D projection after bundle adjustment (in *blue*) is much closer to the 2D detected point (in *white*). So, indeed we have reduced the distance between the detected 2D points and the 3D points projections.

Fig. 7: Visualization over the images of the 2D detected points (in *white*), and the 3D projections before (in *red*) and after (in *blue*) applying bundle adjustment.

7 Conclusions

In this week's assignment, we have seen that in the structure from motion pipeline it is very important to accurately triangulate vanishing points to define correctly the plane at infinity. Otherwise, we will obtain a poor affine reconstruction. It is challenging to find 3D vanishing points and in this lab we have made the assumption of knowing the focal length and principal points to determine them. In a fully autocalibration scenario we would need to determine them by applying other constraints. When metrically rectifying the scene, we have made assumptions on the camera intrinsic matrix for getting enough constraints for determining the image of the absolute conic, but if those assumptions are not true, we will get a poor estimation. When getting camera parameters, it is important to understand our camera model: the orientation of the axes of the camera as well as the world coordinates. Lastly, when performing bundle adjustment, we have seen that a good initialization is very important.

Bibliography

[1] Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, ISBN: 0521540518, second edn. (2004)

[2] Lowe, D.: Object recognition from local scale-invariant features. In: Proceedings of the Seventh IEEE International Conference on Computer Vision. vol. 2, pp. 1150–1157 vol.2 (1999). https://doi.org/10.1109/ICCV.1999.790410

[3] Lu, X., Yaoy, J., Li, H., Liu, Y., Zhang, X.: 2-line exhaustive searching for real-time vanishing point estimation in manhattan world. In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 345–353. IEEE (2017)