3D recovery of urban scenes: Structure from Motion pipeline

Josep Brugués i Pujolràs[1], Sergi García Sarroca[1], Òscar Lorente Corominas[1], and Ian Pau Riera Smolinska[1]

Universitat Autònoma de Barcelona, Bellaterra, 08193, Catalonia, Spain {josep.brugues, sergi.garciasa, oscar.lorentec, ianpau.riera}@e-campus.uab.cat

1 Introduction

In previous weeks, we have been using calibrated cameras or rectified images to estimate projective, affine or metric transformations and performing other tasks related to the field of 3D vision. However, this week we have a different paradigm, as the goal of the project is to apply scene 3D reconstruction from non-calibrated images with a stratified method (recovery of camera matrices and a sparse set of 3D points), by applying the Structure from Motion (SfM) approach on real data. Specifically, we begin with a projective reconstruction and then refine it progressively to an affine and finally a metric reconstruction. We also implemented the resection method (algorithm 7.1 from [1]), which estimates a camera matrix P from a set of 2D-3D point correspondences by using the DLT algorithm.

2 Structure from Motion

Structure from Motion (SfM) is a technique that performs a 3D reconstruction from a set of 2D images of an object or scene taken from different viewpoints [4].

The first stage of the pipeline, and in which this project is focusing, is creating a base two-view reconstruction. To do so, we first use a keypoint detector and matching techniques to obtain 2D correspondences between the two images. Then, we estimate the fundamental matrix, F, using the robust normalized 8-point algorithm. F is then used to compute the projective cameras that will be used to perform projective reconstruction. Moreover, we estimate the 3D reconstruction using triangulation and finally compute the projective reprojection error.

To geometrically verify the system, we also apply an affine and metric rectification and calculate the corresponding affine and euclidean reprojection errors.

Once we have this base reconstruction, we can incrementally register new images, triangulate the new scene points, filter outliers, and refine the reconstruction using bundle adjustment (BA).

In this report we explain in detail the aforementioned steps, indicating the techniques used in each part of the code, as well as the results obtained and the problems that have arisen.

3 Estimating the fundamental matrix

3.1 Finding correspondences

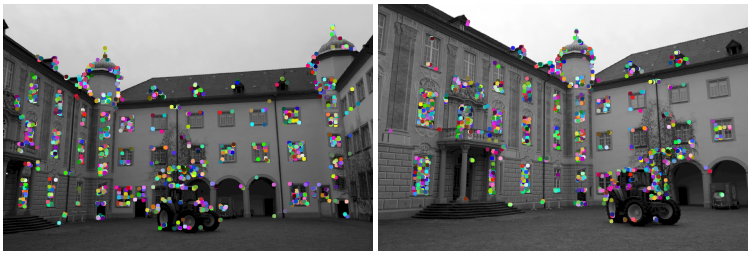
The first step to perform a 3D reconstruction is to estimate the fundamental matrix F from 2D correspondences between two images. To do so, we use two different keypoint detectors: Oriented FAST and Rotated BRIEF (ORB) [3], which is a binary detector, and the Scale-Invariant Feature Transform (SIFT) [2].

To extract the matches, two different methods are tested: kd-tree and Hamming. The first one uses Fast Library for Approximate Nearest Neighbors (FLANN), which contains a collection of algorithms optimized for fast nearest neighbor search in large datasets and for high dimensional features. In other words, it works faster than using brute force (BFMatcher), which is the one being used by Hamming. For this reason, kd-tree is useful when performing SfM with more than two images, but the BFMatcher might be enough for this base 2-image reconstruction.

In Figs. 1a and 1b we can see the keypoints computed with ORB in images 1 and 2, respectively. The results obtained with SIFT and kd-tree are presented in Section 10.1.

3.2 Estimating the fundamental matrix

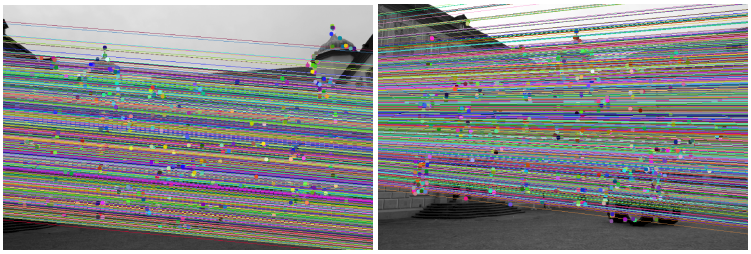
Once we have obtained the keypoints and found the matches between them, we can proceed to estimate the fundamental matrix. To properly estimate F, we use the RANSAC-based robust 8-point normalized algorithm, explained in detail on a previous report,



(a) (b)

(c)

Fig. 1: Keypoints in (a) image 1 and (b) image 2, and (c) matches between images 1 and 2 that are considered inliers



(a) (b)

Fig. 2: Epipolar lines and matches computed on (a) image 1 and (b) image 2

which provides the inlier matches. These inliers are presented in Fig. 1c, where we can see that the implemented method effectively computes 2D correspondences between the two images. Then, we use the algorithm 12.1 from [1] that implements the Optimal Triangulation Method to refine those inliers that minimize the geometric error.

To visually ascertain the estimated F, we then compute and display the correspondent epipolar lines, which are presented in Fig. 2.

4 Projective reconstruction

4.1 Computing projective cameras

Given the estimated F, we can extract two projective cameras. The first one is the canonical form, $P_0 = [I \mid 0]$. We can compute the second projective camera from F using the general formula for a pair of canonical camera matrices ([1], result 9.15):

$$P' = \left[[\mathbf{e}']_\times F + \mathbf{e}'\mathbf{v}^\top \mid \lambda \mathbf{e}' \right] \tag{1}$$

where in our case v is an array of zeros and $\lambda = 1$.

4.2 Projective triangulation for 3D structure

Now that we have the projective cameras, we can triangulate the 3D points ([1], chapter 12.2) from the two images using the corresponding 2D matches. The resulting 3D reconstruction is presented in Fig. 3.

As observed, the resulting 3D reconstruction is quite noisy, so it is hard to visually evaluate the performance of the implemented method. For this reason, we want to refine the results with the affine and metric rectifications.

5 Affine rectification

As explained in chapter 10.4.1 of [1], in order to obtain the affine rectification we need to compute the plane at infinity $\boldsymbol{\pi}$ described by three vanishing points. To do so, we

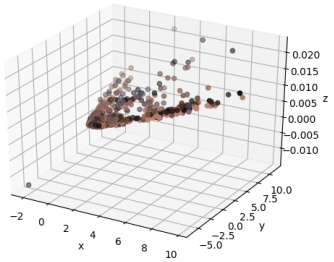


Fig. 3: Estimated 3D projective reconstruction

first compute the 3D projection of the vanishing points. As 3 points describe a plane, we can use them to compute the plane at infinity. Then, we can compute the affine transformation defined by Eq. 2.

$$H_{a \leftarrow p} = \begin{pmatrix} I & | & 0 \\ \pi^T \end{pmatrix} \tag{2}$$

By applying $H_{a \leftarrow p}$ we map p to Π_∞ and recover the parallelism. Specifically, we transform the projective 3D points and the projective camera matrices obtained in Section 4.1 to the affine space. We can observe the resulting 3D reconstruction in Fig. 4.

Notice that the 3D reconstruction of real scenes is not easy to understand. In this case, we can interpret that the 3D reconstruction corresponds to the wall of the building, as it seems that we have recovered parallelism with the affine rectification.

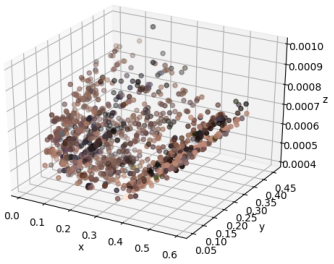


Fig. 4: Estimated 3D affine reconstruction

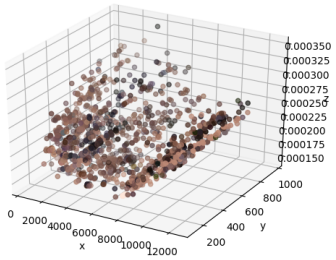


Fig. 5: Estimated 3D metric reconstruction

6 Metric rectification

Once the affine reconstruction is performed, the last step is to upgrade it into a metric reconstruction by applying a 3D transformation described by Eq. 3:

$$H_{e \leftarrow a} = \begin{pmatrix} A^{-1} & 0 \\ 0^T & 1 \end{pmatrix} \tag{3}$$

where A is obtained by *Cholesky* factorization of $AA^T = (M^T \omega M)^{-1}$.

To do so, the objective of the metric reconstruction is to find the image of the absolute conic described by Eq. 4:

$$\omega = K^{-T} K^{-1} = \begin{pmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{12} & \omega_{22} & \omega_{23} \\ \omega_{13} & \omega_{23} & \omega_{33} \end{pmatrix} \tag{4}$$

The solution of ω_v is the null vector of the equation system obtained by five constraints that come from scene orthogonality and known internal parameters. With this vector we can compute Eq. 4. We can observe the resulting 3D reconstruction in Fig. 5.

It is hard to detect any difference between the affine and metric reconstructions, so we will quantitatively measure the performance of our method with the reprojection error.

7 Reprojection error

So far we have analyzed qualitatively the results. To analyze the different methods numerically, we compute the re-projection error, described by Eq. 5.

$$\sum_i d\left(\mathbf{x_i}, \hat{\mathbf{x}}_\mathbf{i}\right)^2 + d\left(\mathbf{x}'_\mathbf{i}, \hat{\mathbf{x}}'_\mathbf{i}\right)^2 \tag{5}$$

where $\hat{x} = PX$ and $\hat{x}' = P'X$.

We can compare the results of computing the error after the projective transformation, affine and metric rectifications in Table 1. For the latter, the euclidean error, we compare the results with and without taking into account the intrinsic parameters of the camera.

Table 1: Reprojection error

Reprojection error	Intrinsics	
	Yes	No
Reprojective	3.50695067e-07	
Affine	3.50695140e-07	
Euclidean	8.614e08	3.50695165e-07

As observed, the error is minimum when using the stratified method, which is based on first performing a projective reconstruction and then refine it progressively first to an affine and finally a metric reconstruction. Notice that the affine and metric rectifications do not have a great impact in the reduction of the reprojection error, as these are useful to geometrically verify the system, which has been proven in Figs. 4 and 5.

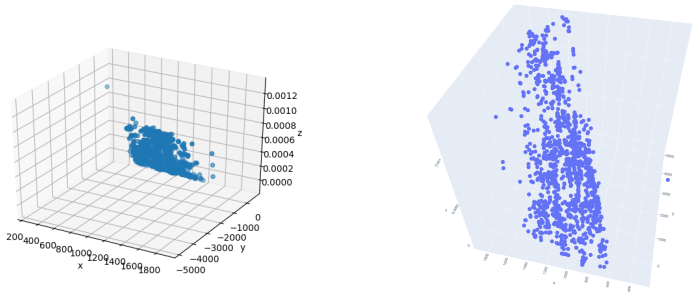
On the other hand, the error becomes huge when using the intrinsic parameters of the camera. Instead of performing a 3D reconstruction in an stratified way, this method directly estimates the 3D points using triangulation together with the fundamental matrix and the 2D correspondences. The problem, as stated in chapter 12.2 of [1], is that this is not suitable for projective reconstruction, since the linear triangulation methods are not projective-invariant, and thus the reconstruction is not correctly performed.

8 Bundle adjustment

Given a set of images taken from different viewpoints, bundle adjustment (BA) is the problem of simultaneously refine the 3D coordinates describing the scene geometry, the parameters of the motion and the optical characteristics of the camera. To perform bundle adjustment in this project, in terms of code we first need to convert the list of tracks to a suitable format, used later by the *PySBA* class that performs BA. In more detail, the transformation consists on extracting the data from the tracks list and create the following new data:

- A list of 3D points from the world.
- A list of 2D points from all the images.
- A list of 2D point indices, that controls to which track corresponds each 2D point, as for each track there are as many points as cameras.
- A list of camera indices, that are used to control from which camera is each 2D point.

The results obtained after bundle adjustment are presented in Fig. 6. As the image measurements are noisy, the projection equation for some points are not satisfied exactly,



(a) (b)

Fig. 6: Estimated 3D bundle adjustment reconstruction from two different view points

so we use the bundle adjustment to adjust the bundle of rays between each camera and the set of 3D points. This makes it robust to the noise and outliers, and the final 3D reconstruction is more precise, as observed in the result.

9 Optional. Resection method

Resection is a numerical method for estimating camera projection matrices (P) from 2D-3D point correspondences. In this project, we use this approach to estimate P when working with more than two images. This approach uses the DLT algorithm in a very similar way to how a homography H is estimated, but taking into account that the matrix P is $3x4$ instead of $3x3$, so instead of 4 pairs of correspondences we need 6 (as we have 12 - 1 degrees of freedom). The least squares method is also used to minimize the correspondent geometric error. The complete approach is described by Alg. 1.

Algorithm 1 Resection

Input: $tracks$, $camera\_index$
Output: P

1. Normalization of points: the set of 2D points have to be centered at $(0, 0)$ and their average distance to the centroid is $\sqrt{2}$, and the 3D points need to be centered also at $(0, 0)$ but with an average distance to the centroid of $\sqrt{3}$.
2. DLT algorithm: From the $2nx12$ matrix A constructed from the point correspondences, find the estimated P using SVD: the last column of V is the solution.
3. Use the least squares method together with the Levenberg-Marquadt iterative algorithm to find the camera matrix P that minimizes the geometric error $\sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2$.
4. Denormalize the matrix P.

Concretely, we have used an alternative method to compute the camera matrices when using more than two images. An example projection matrix computed with this

method is presented in Eq. 6. Notice that we cannot show any visual result (such as 3D projective reconstructions) because we were not able to implement the optional *Tracks structure improvement*, that would have allowed us to use the code with more than two images. Nevertheless, the resection method is completely implemented.

$$P = \begin{bmatrix} -1.208e-07 & -1.208e-07 & -1.208e-07 & -1.208e-07 \\ -5.456e-08 & -1.872e-08 & 6.707e-03 & -7.534e-07 \\ -4.918e-03 & -1.308e-03 & -2.986e+03 & 1.0 \end{bmatrix} \quad (6)$$

10 Optional: Strategies for pipeline improvement

We tried some strategies to check if we could improve the results, and we can find an exhaustive list of methods in [4].

10.1 SIFT and kd-tree

In section 3.1 we stated that the methods used to find 2D correspondences were using ORB and Hamming matching function. As an alternative, we tried to use SIFT to detect keypoints and kd-tree to match them, to see if we could get less outliers and thus a more robust estimation of the fundamental matrix F.

The inliers obtained in this case are presented in Fig. 7. As observed, we have a large number of outliers compared to ORB, so we are not expecting better results in this case for the 3D reconstruction. These are also shown in Fig. 8. However, RANSAC introduces a random factor in this process, so an exhaustive study should be performed in order to decide which keypoint detector and matching algorithm works better.

10.2 Search for more inliers

As observed, we obtained much more outliers using SIFT and kd-tree, so we thought that it might be useful to reuse these outliers and try to get more matches from them. This is inspired by the part (v) algorithm 11.4 of [1], where it is suggested to use the guided matching method. It basically consists on determine further interest point correspondences using the initially estimated fundamental matrix to define a search strip about the epipolar line. In other words, we search for more matches by finding the outliers that were close to the obtained epipolar lines.

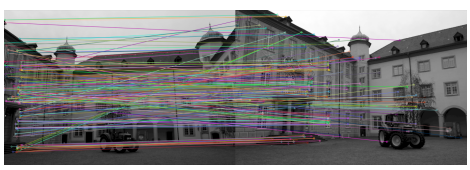
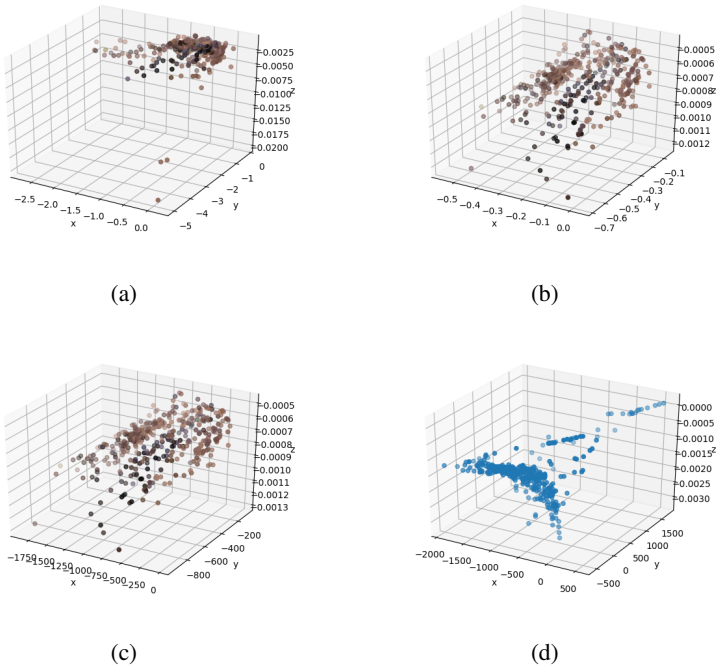


Fig. 7: Matches considered inliers using SIFT



(a) (b)

(c) (d)

Fig. 8: 3D reconstructions using SIFT: (a) projective, (b) affine, (c) metric and (d) after bundle adjustment

11 Conclusions

In this project we have worked on the Structure from Motion pipeline, that covers most of the methods implemented on the previous weeks. Specifically, we have used 2D correspondences (computed from keypoints detected with ORB) to estimate the fundamental matrix F using the robust normalized 8-point algorithm. Then, we have estimated the projective cameras using the DLT algorithm to then use linear triangulation and obtain the corresponding 3D projective reconstruction. Afterwards, we applied affine and metric rectifications and finally implemented bundle adjustment to refine the reconstructions. Finally, we have computed the reprojection error in each case to quantitatively measure the performance of our system.

As in previous weeks, we compared the obtained results using ORB and SIFT keypoint detectors. We finally used ORB as with SIFT we obtained a large number of outliers making the estimation of F inaccurate. However, this large number of outliers might be related to the random factor introduced by RANSAC, so we should deeply study this comparison to extract finer conclusions.

Using triangulation, we were able to create a cloud of points estimating their position in the 3D world. Although we could imagine the relation between the obtained visualization and the original image, the results were a bit poor. Even so, the affine

rectification seems to improve the visualization results, as it recovers the parallelism in the scene. Nevertheless, the metric rectification does not have such greater impact in the result. On the other hand, the bundle adjustment refines the bundle of rays between each camera and the set of 3D points, making the method robust to the noise.

In terms of reprojection error, the results show that the implemented method is able to correctly perform a 3D reconstruction of the scene using the stratified method. However, this error is not further decreased when refining the 3D reconstruction using affine and metric rectification or bundle adjustment.

On the contrary, when using the intrinsic parameters of the camera, the 3D points are directly estimated using triangulation together with the estimated fundamental matrix and the 2D correspondences. In this case, we fail to correctly reconstruct the scene, as this method is not suitable for projective reconstruction, since these linear triangulation algorithms are not projective-invariant.

For the scenario with more than 2 images, we implemented the resection method to estimate the camera matrix from 2D-3D point correspondences using the DLT algorithm. The algorithm was correctly implemented, but due to the fact that the optional task to adapt the tracks for more than 2 images was not implemented, we were not able to get qualitative results.

Finally, we tested different ways to improve the SfM pipeline, which include using other keypoint detectors and matching functions, or even reusing the outliers close to the epipolar lines to obtain more matches (guided matching method). Unfortunately, for the 2-image case, we did not observe any improvement using those techniques, but we will study other methods to see if the results can be improved.

References

1. R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. USA: Cambridge University Press, 2003.
2. D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–, 11 2004.
3. E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, 2011, pp. 2564–2571.
4. J. L. Schönberger and J. Frahm, "Structure-from-motion revisited," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4104–4113.