# M4 - 3D Vision

## 3D recovery of Urban Scenes

Ana Caballero - ana.caballeroc@e-campus.uab.cat
Arnau Vallvé - arnau.vallve@e-campus.uab.cat
Claudia Baca - claudiabaca.perez@e-campus.uab.cat
Joaquim Comas - joaquim.comas@e-campus.uab.cat

### INTRODUCTION

The goal of this project is to learn the basic concepts and techniques to reconstruct a real world scene given several images (points of view) of it, not necessarily previously calibrated. In this project we focus on 3D recovery of Urban Scenes using images of different datasets, namely images of facades and aerial images of cities.

### WEEK 5

**Goal:** Reconstruction from uncalibrated images with a stratified method
(recovery of camera matrices and a sparse set of 3D points)

**Mandatory:**

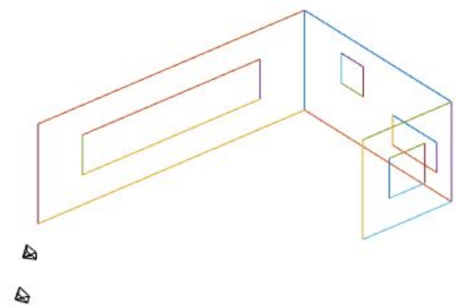
- Projective reconstruction (synthetic data)
- Affine reconstruction (synthetic data)
- Metric reconstruction (synthetic data)
- Projective reconstruction (real data)
- Affine reconstruction (real data)
- Metric reconstruction (real data)

**Optional:**

- Optional P, P' from F
- Optional add a 3rd view
- Free optionals

## 0. Create synthetic data

In this first part, we will focus on the reconstruction of a synthetic image. In the next images we can see the 3D points distribution of the scenario (on left) and the visualization of the reconstruction expected (on right):



**Images 1,2: 3D points data and visualization of lines respectively**

## 1. Projective reconstruction (synthetic data)

### Factorization Method function

Factorization method computes a projective reconstruction explained in Sturm and Triggs' 1996 paper. This function returns two values. The first one Proj corresponds to 3*Ncam x 4 matrix containing the camera matrices while the second one Xproj corresponds to 4 x Npoints matrix of homogeneous coordinates of 3D points.

The Stum and Triggs' paper describes a 8 step algorithm. The aim of the first step is to normalize 2D homogeneous points for both cameras. Then, we have to estimate the fundamental matrices and epipoles for each camera. To do this we have the fundamental_matrix function which is provided. The epipoles can be estimated directly from SVD of the fundamental matrix. The epipoles *e* corresponds to the third column of V while *e*' is given by the third column of U. In this step we only need *e* (column V from SVD, as we just explained).

Once we have the fundamental matrices and the epipoles we can proceed to determine the scale factors using the following equation:

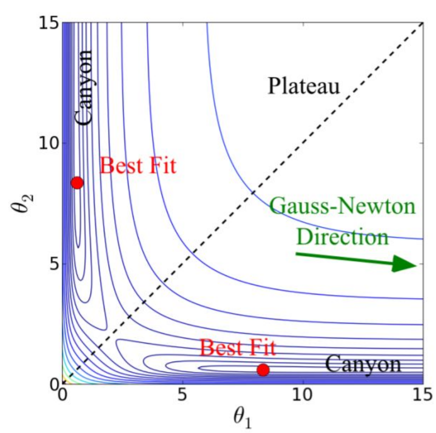
$$\lambda_{ip} = \frac{(\boldsymbol{e}_{ij} \wedge \boldsymbol{q}_{ip}) \cdot (\boldsymbol{F}_{ij}\boldsymbol{q}_{jp})}{\|\boldsymbol{e}_{ij} \wedge \boldsymbol{q}_{ip}\|^2} \lambda_{jp}$$

The following two steps are done together. They consist on building the rescaled measurement matrix W and balance it by column-wise and triplet of rows-wise scalar multiplication.

To balance W matrix we need to follow three steps described in the paper.

- Rescale each column l
- Rescale each triplet of rows
- And if entries of W changed significantly return to the first step

To know if the entries (scaled factors, lambda) have significantly changed, we consider that the euclidean difference must be greater than 10%. This percentage is obtained by Levenberg-Marquardt method for dealing with the nonlinear least-squares minimization. Applying this method, we can check that as Sturm and Triggs' paper describe in the paper, 2% is not enough.




To build the rescaled measurement matrix W in projective method and shape we have to multiply lambda and normalized coordinates (q) as follows:

$$\boldsymbol{W} = \begin{pmatrix} \lambda_{11}\boldsymbol{q}_{11} & \lambda_{12}\boldsymbol{q}_{12} & \cdots & \lambda_{1n}\boldsymbol{q}_{1n} \\ \lambda_{21}\boldsymbol{q}_{21} & \lambda_{22}\boldsymbol{q}_{22} & \cdots & \lambda_{2n}\boldsymbol{q}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{m1}\boldsymbol{q}_{m1} & \lambda_{m2}\boldsymbol{q}_{m2} & \cdots & \lambda_{mn}\boldsymbol{q}_{mn} \end{pmatrix}$$

Once we have W matrix built, before we continue, we have to check that the correct projective depths lambda, the matrix W has rank at most rank 4. Our W is size 6x24.

The next step is to compute SVD of the balanced matrix W and recover projective motion and shape.

Then we apply a convergence criteria, computing the euclidean distance (d) between data points and projecte points in both images. If they don't converge we have to readjust lambda with the projective motion and shape we just calculated. If they converge we can proceed with the last step. Finally we adapt projective motion, to account for the norm transf T of step 1. Below we can see the visualization of the projective reconstruction. Because of the inherent projective ambiguity it might look odd, but we'll be able to reconstruct it adding further steps.

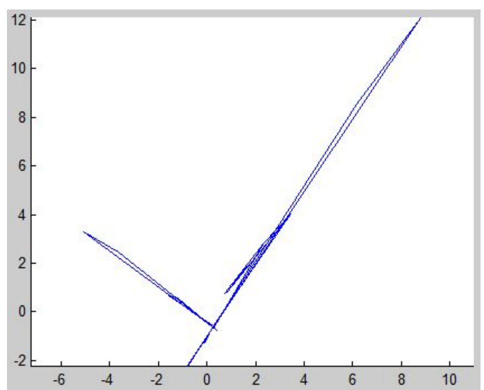
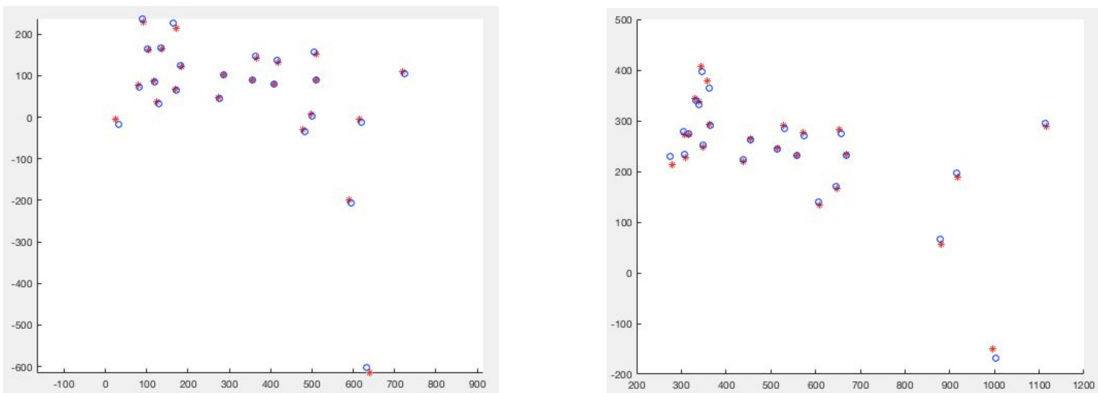



**Image 3: Projective reconstruction (synthetic data)**

**Reprojection error**

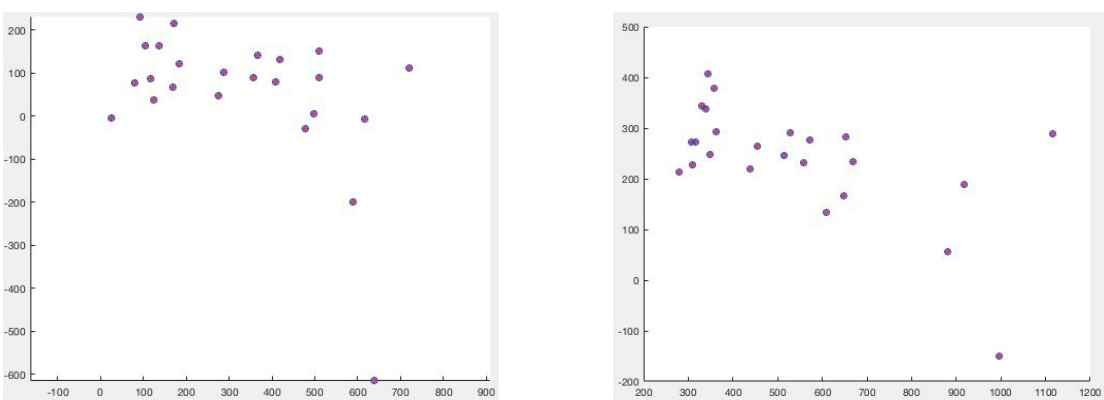
In this next section we can see the difference between the projected points and the data points which is determined by the effect of the projective depths initialization. In our case, we have used to different initializations:

First initialization, where $\lambda^i_j = 1$ for all i and j:



**Images 4, 5: Comparison between projected and data points using first initialization( $\lambda^i_j = 1$).**

Second initialization, where $\lambda^i_j = \frac{(x_j^1)^T F_{i1}(e \times x_j^i)}{\left\| e \times x_j^i \right\|^2} \lambda_j^1$ with $\lambda^1_j = 1$:



**Images 6, 7: Comparison between projected and data points using second initialization (Sturm and Triggs initialization).**

Visually analyzing the previous figures we can qualitatively notice that the second initialization seems to provide better results in comparison with the first one. To numerically see better the difference between them we have performed the following histograms according to each initialization where we observe a very small error (10^-12) in second initialization compared with the first one:

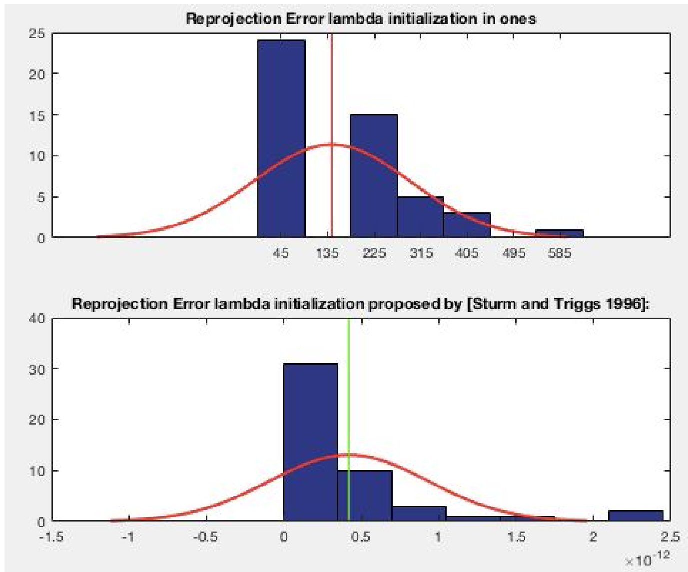



**Image 8: Comparison between projected error of first (top) and the second (bottom) initialization.**

Being the mean reprojection error for initialization with ones much higher:

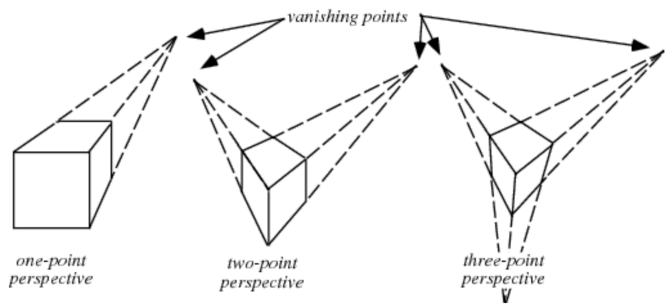
```
meanError1 =  142.8430
```

And the mean reprojection error for initialization proposed by Sturm and Triggs practically zero:

```
meanError2 = 3.2019e-12
```

### Vanishing_point

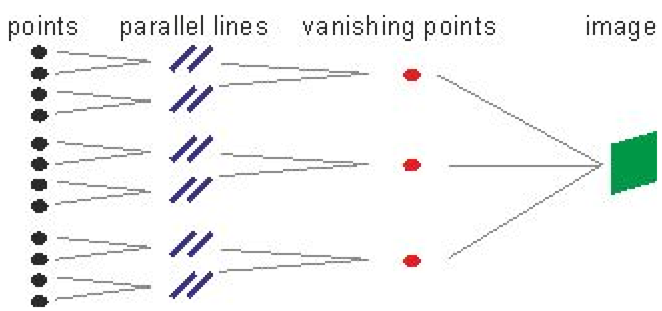
A vanishing point is a point on the image plane of a perspective drawing where the two-dimensional perspective projections of mutually parallel lines in three-dimensional space appear to converge. Vanishing points are obtained from clusters of parallel lines.




So, to compute the vanishing points we used two parallel lines formed by the line that joins two pair of points by applying the cross product. Finally we related both parallel lines by selecting the orthogonal triplet. That means applying the cross product of the normalized previous lines.

### 2. Affine reconstruction (synthetic data)

To compute the affine reconstruction we used three sets of four points. Each set form two parallel lines in order to get three vanishing points per image.
That means, for each image:




As we have two images, we have to find 6 vanishing points.

Then, this vanishing points are used to compute the matrix Hp that upgrades the projective reconstruction to an affine reconstruction. To do this we need to find 3 intersections of sets of lines in the scene that are supposed to be parallel, that's why we use vanishing points.

Finally we use SVD to find a transformation H that maps the plane. This plane contains all points at infinity. The plane, that corresponds to the last column of the right singular vectors in SVD, is used to build the Hp matrix.

```
Hp =

    1.0000         0         0         0
         0    1.0000         0         0
         0         0    1.0000         0
    1.4134   -3.5280   -0.3729    1.0000
```

Once we have the Hp matrix we can check the results. We can see how it starts to seem more like what we would want, but there's still more to add (we'll need metric reconstruction).

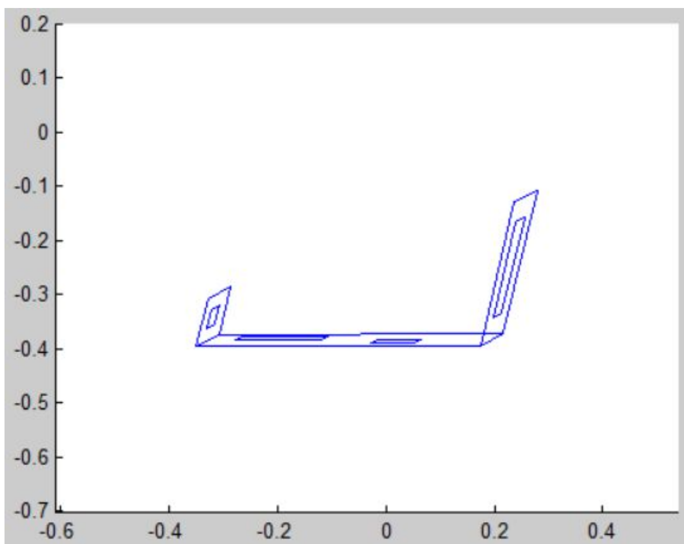



**Image 9: Affine reconstruction (synthetic data)**

### 3. Metric reconstruction (synthetic data)

Now, we want to improve the affine reconstruction to a metric reconstruction by obtaining the adequate transformation matrix. For the task of metric reconstruction, we'll first be required obtain the image of the absolute conic *w*, given a combination of constraints (orthogonality, known internal parameters *K*, same cameras). It is defined as:

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

By computing the vanishing points *u, v, z* and using the mentioned constraints below:

$$\mathbf{u}^T \omega \mathbf{v} = 0$$

$$\mathbf{u}^T \omega \mathbf{z} = 0$$

$$\mathbf{v}^T \omega \mathbf{z} = 0$$

$$\omega_{11} = \omega_{22}$$

$$\omega_{12} = 0$$

We can formulate the matrix *A* below from which we can obtain the *w* as the null vector of it, using the SVD procedure.

$$A = \begin{pmatrix} u_1 v_1 & u_1 v_2 + u_2 v_1 & u_1 v_3 + u_3 v_1 & u_2 v_2 & u_2 v_3 + u_3 v_2 & u_3 v_3 \\ u_1 z_1 & u_1 z_2 + u_2 z_1 & u_1 z_3 + u_3 z_1 & u_2 z_2 & u_2 z_3 + u_3 z_2 & u_3 z_3 \\ v_1 z_1 & v_1 z_2 + v_2 z_1 & v_1 z_3 + v_3 z_1 & v_2 z_2 & v_2 z_3 + v_3 z_2 & v_3 z_3 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 \end{pmatrix}$$

With the corresponding camera matrix as :

$$P = [M|m]$$

We can use M and the previously computed image of the absolute conic *w* to solve by Cholesky factorization:

$$AA^T = (M^T \omega M)^{-1}$$

and obtain the transformation matrix:

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

Applied to the synthetic images we obtain the following metric rectified results by consecutively applying projective to affine and affine to metric transformations to the data.

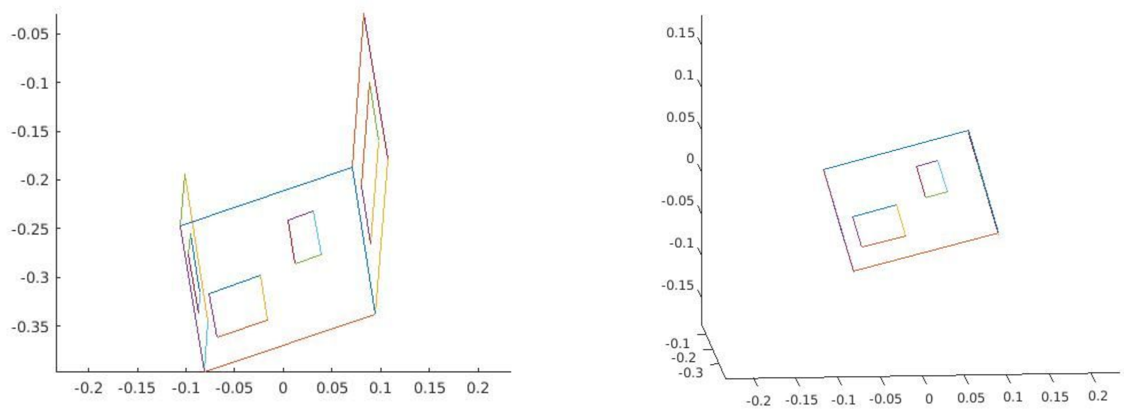



**Image 10,11: Two views of the same metric reconstruction image (synthetic data)**

## 4. Projective reconstruction (real data)

Now, we'll work with two provided images (real data) instead of synthetic data as we have been previously using. In this second part, we will focus on the reconstruction of a real scene. In the next images we can see the 2 views of the scenario that we will use for the projective reconstruction:

**Images 12,13: Pair of views for the projective reconstruction of a real scene**

To apply the projective reconstruction, first we need to compute the projective reconstruction using the factorization method. First, we need to compute the keypoints in each images and its correspondences (matches). Then, we need to apply the fundamental matrix from the previous coordinates and matches in homogeneous coordinates.

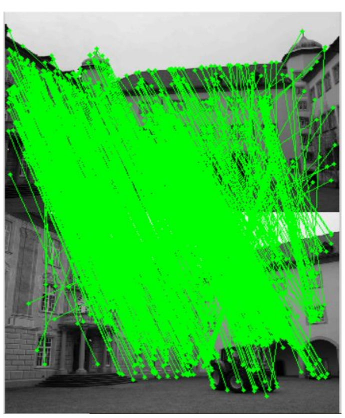


**Image 14: Keypoints and matches**

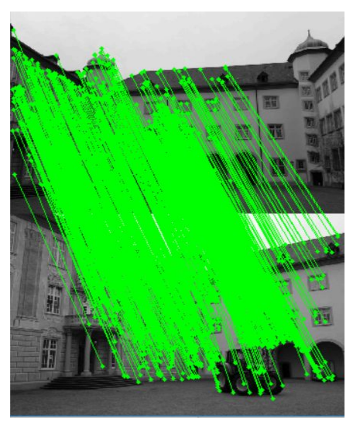


**Image 15: Keypoints and inliners**

Finally, we check projected points as we did in exercice 1, using the fundamental matrix.

**Image 16: Keypoints and correspondences in image 1**

**Image 17: Keypoints and correspondences in image 2**

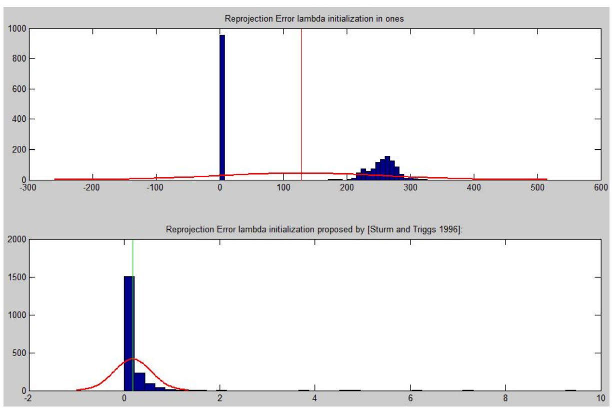



**Image 18: Comparison between projected error of first (top) and the second (bottom) initialization.**

In this real image case we can see how the different initialization has also an importance on the projected error. In the synthetic case however, as it is probably simpler, using the Sturm and Triggs initialization got a value much closer to zero than now, even though the errors obtained now are still much closer to zero than with the initialization with ones.

Numerically, we obtain with the initialization with ones:

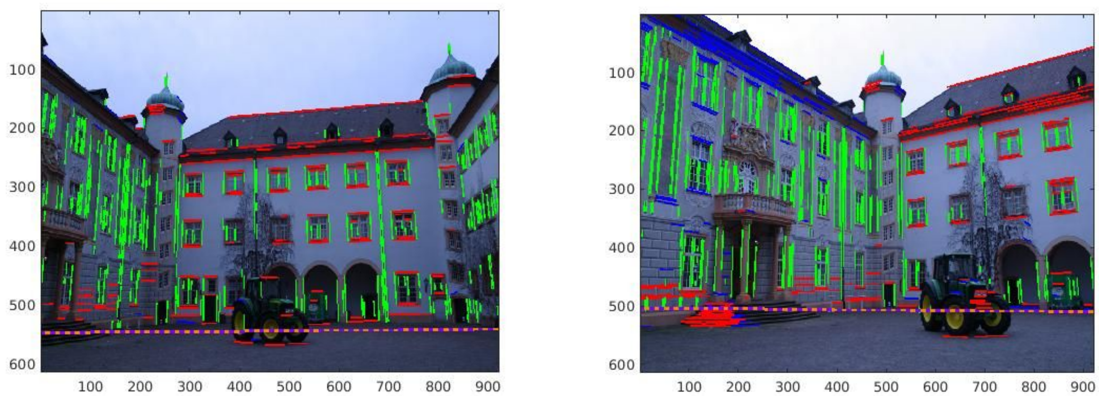
```
meanError1 =  128.6917
```

And with the initialization proposed by Sturm and Triggs:

```
meanError2 = 0.1974
```

## 5. Affine reconstruction (real data)

As we did on previous exercises with synthetic data, we aim here to obtain the transformation matrix *Hp* necessary to update the projective transformation to an affine transformation to the real images. For the detection of the vanishing points, here we'll be using a function provided in [http://dev.ipol.im/~jlezama/vanishing_points/](http://dev.ipol.im/~jlezama/vanishing_points/) to automatically obtain them via primal and dual point alignment detection. Below we can see the automatic detection outputs for the image pairs we'll be using.



**Images 19,20: Automatic vanishing point detection using detect_vps function in image pair 0000.png and 0001.png**

We perform similar steps we already used in previous exercises when working with synthetic data; triangulate, SVD and obtain the transformation matrix *Hp* as we already saw. We obtain the following affine reconstructed results from the data.

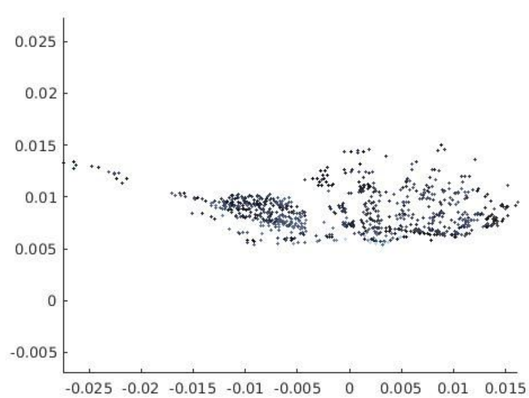


**Image 21: Affine reconstruction results for real data**

## 6. Metric reconstruction (real data)

As we did with synthetic data, we will perform the metric reconstruction but using the real images this time. The procedure followed has already been explained in previous exercises. Compared to what we obtained with the simpler synthetic data, results seem difficult to interpret.

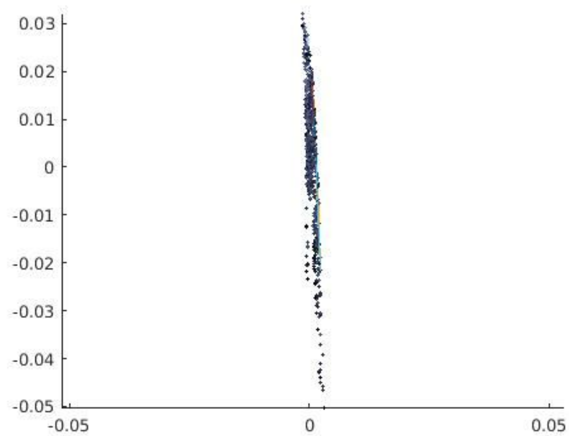



**Image 22: Metric reconstruction results for real data**

**Conclusions:**

- Different initializations of lambda (i.e. ones, Sturm and Triggs) clearly make an impact on the reprojection error, having a mean error values nearly zero for the Sturm and Triggs lambda, as seen on the histograms.

- For simpler cases, such as the synthetic images, the reprojection error is much lower than when using real images. Sturm and Triggs initialization is still however the better for a low reprojection error.

- Going from the simpler synthetic case to the real data makes the interpretation of the reconstructed results more complex, which highlights the utility of using simpler cases before to stepping up to complex cases.