Homography Estimation, Image Mosaics and Calibration

Marc Carné
Universitat Autnoma de Barcelona
Bellaterra, Catalonia/Spain
marc.carne.herrera@gmail.com

Alejandro Nespereira
Universitat Autnoma de Barcelona
Bellaterra, Catalonia/Spain
alejandronespereira@gmail.com

Gonzalo Benito
Universitat Autnoma de Barcelona
Bellaterra, Catalonia/Spain
gonzabenito@gmail.com

Abstract

An image is the projection of the real world on an image plane. Projecting from a 3D world to a 2D world we lose spatial information. Besides, a projective transformation is bound to create some distortions in the image plane. Projective transformation have inherently a lose of lengths, angles metrics or parallelisms. Despite this information loss we can reconstruct scenes in 3D or build mosaics from recovered images. In this work we focus on homographies and their uses. We present one of the methods available to compute them and finally we apply this knowledge to a set of images that comes from different scene views in order to create a single mosaic. Additionally, we explore one of the methods used for auto-calibrating cameras with easily available materials.

1. Motivation

Projecting real world 3D scenes in a 2D images has some limitations and in that process we lose some important information. Due to physical limitations in the cameras, the projection of the world is limited directly by the camera field of view and lens shape. Therefore, it is often that we are not able to capture all what we want, or as detailed as we would like.

An image is formed by a set of pixels with an associated value. Image processing algorithms often focus in addressing these values to enhance features of the images or recognize and detect objects of interest. In this report, however, we are more interested in the position of the pixels in our images and their relation with the real world or other images. We know there is a projective transformation between a plane in the real world and the image plane. This transformation relates the homogeneous coordinates of the world to the homogeneous coordinates of the image plane, that we had seen in the previous session.

Therefore, a point expressed in homogeneous coordinates can be transformed using a transformation matrix, also known as homography. This homography, a 3×3 matrix) contains information about the rotation, translation, affinity and projection between both images. This fact allows us to transform an entire image but also allow us to express to image in the same spatial reference (the same point of view). Conceptually this is the same as if the two images were captured with the same camera. This is possible computing the position of the cameras and moving one camera and its image plane to the position of the other. This method allows us to express different images with a same origin and build mosaics.

In the next sessions we will see how to compute this matrix and how to apply it, and the consideration needed to have into account.

2. Method

Sometimes for a single scene we have different images representing different points of view. With this information it is possible to compute a single mosaic from them. To build a mosaic we need first to compute homographies between all the views. Establishing one image as the central view, we can use the relation between this view and the rest to generate artificial image planes to generate a mosaic of images that share a common point of view.

The main idea of the approach is that having three different images-see Fig. 1-, we want to represent them with the same point of view. This can be achieved by fixing a central image and two lateral images. From that point what we want is to find the relationship between the central image and the lateral ones. Doing that for the two lateral images we can build a mosaic superposing all the images (now three images that share the point of view).

As we explained before, that matrix consist in a 3 by 3 matrix, so it has 8 degrees of freedom (there are nine coefficients but one is a scaling factor). We will use the DLT algorithm (**D**irect **L**inear **T**ransformation) to find a homography from 4 points correspondence. Correspondence points will be found and matched using **SIFT** (**S**cale **I**nvariant **F**eature **T**ransform). As the SIFT will give us a large number of correspondences with plenty of mismatches, we have to use a more robust algorithm for the homography com-

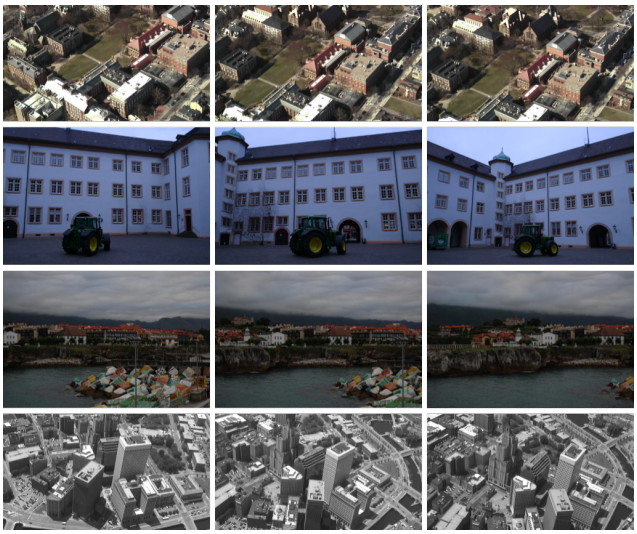

Figure 1: Sets of image triplets used.

putation. For this part we will use **RANSAC** (**R**andom **S**ample **C**onsensus) to select the matrix transformation obtained with 4 correspondences that best represents the transformation.

In this section the DLT algorithm will be explained as a baseline and then how to use it with the RANSAC method to compute the best one.

2.1. DLT algorithm

To compute an homography, we use an algorithm that consists in build a correspondence matrix and compute the singular value decomposition. The result of this process are the homography values.

The first that we have to know is that the relation between correspondent points between images can be expressed as:

$$x' = Hx$$

Where 'H' corresponds to the homography that relates both points.

This can be expressed as product matrix as:

$$\begin{pmatrix} x' & y' & 1 \end{pmatrix} = \begin{pmatrix} h11 & h12 & h13 \\ h21 & h22 & h23 \\ h31 & h32 & h33 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (1)$$

The same if we express each row of the homography as a vector:

$$\begin{pmatrix} x' & y' & 1 \end{pmatrix} = \begin{pmatrix} h1^T \\ h2^T \\ h3^T \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (2)$$

If we expand the product we have the relation of points in homogeneous coordinates, so a direct transformation to euclidean coordinates can be done having as a result:

$$\frac{x'}{w'} = \frac{h1^T * \mathbf{x}}{h3^T * \mathbf{x}}; \frac{y'}{w'} = \frac{h2^T * \mathbf{x}}{h3^T * \mathbf{x}} \quad (3)$$

Note that x' and y' are a dimension coordinate while **x** corresponds to all the dimensions of the point, in homogeneous coordinates. From the last expression we have two equations, that can be expressed as a matrix product:

$$Ai * H = 0$$

Expressing the each row of the homography as a vector, and having being x', y' and w' each one of the coordinates of the reference point and **x** the other point itself, we obtain this two equations:

$$\begin{pmatrix} 0 & -w'\mathbf{x} & y'\mathbf{x} \\ w'\mathbf{x} & 0 & -x'\mathbf{x} \end{pmatrix} \begin{pmatrix} h1^T \\ h2^T \\ h3^T \end{pmatrix} = 0 \quad (4)$$

Having the mathematical expression of the matrix 'Ai', a 2 by 9 matrix, we know for each pair of points (a correspondence) we have one of this matrix, so from the 4 correspondences we will have 4 matrix. Concatenating vertically those matrix we achieve a 8 by 9 matrix that are formed by the 4 correspondences. This matrix will be called **A**.

Once we have the matrix of correspondences **A** built, we have to compute a singular value decomposition (SVD). The result of the SVD is three different matrices: **U** and **V** that corresponds to two matrix containing orthogonal basis of the space and a matrix **D** that contains the eigenvalues (diagonal matrix).

As result of the SVD we obtain the matrix **V** transposed. As the columns in the matrix without transpose are sorted by the value of the correspondent eigenvalue, we look for the column of th matrix **V** that has the lower eigenvalue associated.

That column is a nine dimensional vector that corresponds to the homography values we are looking for. Once we have that values, we have to order them in a matrix form.

2.2. RANSAC algorithm

Previous section showed how to find an homography from 4 correspondences, but when we look for correspondences with SIFT, we are often faced with a large number of points. Moreover, not all the points are valid matches. For that reason not all homographies obtained through this are valid.

RANSAC is a method based on iteration where in each iteration 4 random correspondences (from SIFT detection) are taken and a homography is computed as described before. Then for each SIFT proposed point the homography is applied so a transformed point is computed. The basic step here is to compute the geometrical error, that consists

in the euclidean distance between the transformed point of the image to transform (x) and the correspondence point in the destination domain image (x').

$$d = d(x', Hx) + d(H^{-1}x', x) \tag{5}$$

Once we have calculated the geometrical distance for all the points, the inliers are counted. It is considered an inlier a point that has a geometrical error lower than a threshold. That procedure is done many iterations (specified number or until the updated number of iterations is equal to a certain value). After all the iterations the homography with a large number of inliers is considered valid, but to refine it we compute the updated homography with all the inlier points.

2.3. The Gold Standard Algorithm

When more than 4 correspondences between two images appear, an homography H that can fully fit them may not be achievable. As such, comes the need to find a suitable H to fit the correspondences the best as it can. A loss function is defined so that H minimizes it when fitting the correspondences of the images. There may be various cost functions that can be used as target. Usually, there will be one of these which its minimization H provides the best result according to the data. The Gold Standard algorithm aims to minimize this cost function. The name of this algorithm refers to the fact that its result is the best in terms of H estimating the transformation. The steps for this begin with an initial estimation of $\hat{\mathbf{H}}$ to begin the minimization. DLT algorithm as well as RANSAC can be used for this purpose. Following, comes the estimation of the subsidiary variables $\hat{\mathbf{x}}_\mathbf{i}$ with the measured points. By the implementing LevenbergMarquardt algorithm, the euclidean distance between estimation and measure is minimized over 2n+9 variables: 2n for the n 2D points $\hat{\mathbf{x}}_\mathbf{i}$, and 9 for the homography matrix $\hat{\mathbf{H}}$.

2.4. Camera calibration

In order to fully understand the 3D world through our cameras we need a precise model that can map points with little or no error. To do so, there exist several camera calibration algorithms with different approaches. We have implemented a solid algorithm that, with 3 different views of a planar pattern, is able to obtain not only the camera intrinsics matrix K, but also the relative poses from the different views, characterised as R and t.

The first step is obtaining the correspondences between each view and the real world object. In our case we are using the digital image of a graffiti as our real world object, and different views of a laptop displaying the same image. The planar surface of the screen ensures that the pattern lies in a plane. The pixel measurements from the digital image are taken as the real world measures of our object. As both the planar pattern and the image plane are a plane in projective space, we can establish a homography for each view using the DLT algorithm. Then, for each view i we have $x_i = H * x_T$, where x_i are the points of our image,H is the obtained homography and x_T are the points of our template. From here, it is straightforward to arrive at

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim P \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = K * [R \| t] \sim K * \begin{pmatrix} r1 & r2 & t \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{6}$$

We can safely assign $Z = 0$ at all our planar pattern points, as we will be taking it as our reference. Then it is deduced that

$$\begin{pmatrix} K^{-1}h1 & K^{-1}h2 & K^{-1}h3 \end{pmatrix} \sim \begin{pmatrix} r1 & r2 & t \end{pmatrix} \tag{7}$$

As $r1$ and $r2$ are the first two columns of a rotation matrix R, we can assume that $r1 * r2 = 0$ and $r1 * r1 = r2 * r2 = 1$. Then

$$h1^T K{-}TK{-}1h2 = 0 h1^T K{-}TK{-}1h1 = h2^T K{-}TK{-}1h2 \tag{8}$$

For convenience, we can define $w = K{-}TK{-}1$, which is the image of the absolute conic:

$$w = \begin{pmatrix} w11 & w12 & w13 \\ w12 & w22 & w23 \\ w13 & w23 & w33 \end{pmatrix} \tag{9}$$

or $\Omega = \begin{pmatrix} w11 & w12 & w13 & w22 & w23 & w33 \end{pmatrix}$. From $h_i^T w h_j = v_{ij}\Omega$ with

$$\begin{aligned} v_{ij} = (&h_{1i}h_{1j}, h_{1i}h_{2j} + h_{2i}h_{1j}, h_{1i}h_{3j} + h_{3i}h_{1j}, \\ &h_{2i}h_{2j}, h_{2i}h_{3j} + h_{3i}h_{2j}, h_{3i}h_{3j}) \end{aligned} \tag{10}$$

Now we can define the equations at (8) as

$$v_{11}\Omega = 0; (v_{11} - v_{22})\Omega; \tag{11}$$

Therefore, one view provides us two equations. As Ω has 6 unknowns, it is enough to have 3 different views of our pattern to successfully determine Ω. Using more views would give a more precise output. The resulting system $v\Omega$ can be easily solved using SVD to get w. From here, and knowing that $w = K{-}TK{-}1$, matrix K is retrieved using Cholesky. From here, $r1, r2, t$ can be retrieved for each view using equation (7).

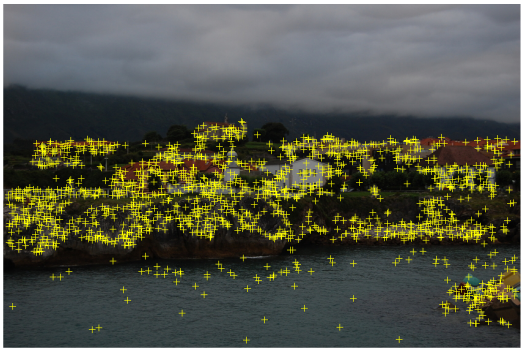

Figure 2: Keypoints found by the SIFT feature extractor. SIFT keypoints are a good starting point for image matching.

3. Results

3.1. Task 1

This task consisted in acquiring a set of correspondences between the images. To do so, a SIFT feature extractor and descriptor was applied over a set of different image triplets in order to visualize the variations in results. The sift points such as seen in Fig. 2, were matched between images in order to find correspondences. Figure 3 shows such correspondences for one of the cases.

3.2. Task 2

Here, we display the results of computing the homography through the DLT algorithm. This implementation looks for correspondences that fit an homography with a marginal deviation. Once we have H calculated, it can be visualized in Fig. 4 how any point in image **a** is matched to another point in image **b** through H.

3.3. Task 3

The third task meant to compose mosaics with the images which homographies were computed. The results vary mostly depending on the possibility to perform a good match the three components of a mosaic. Not all points can be matched correctly through the same homography and this is were non semantically correct details appear. For instance, in the castle sequence, the tractor does not hold the assumption of every object being more or less at the same distance plane from the camera, thus its position varies more than the rest of the scene. This makes it impossible for the same homography H to map points from the background and also from the foreground tractor in a correct way. The result is that the tractor is almost vanished from the scene, although its most brilliant pixels prevail over the dim background ones. The opposite case is

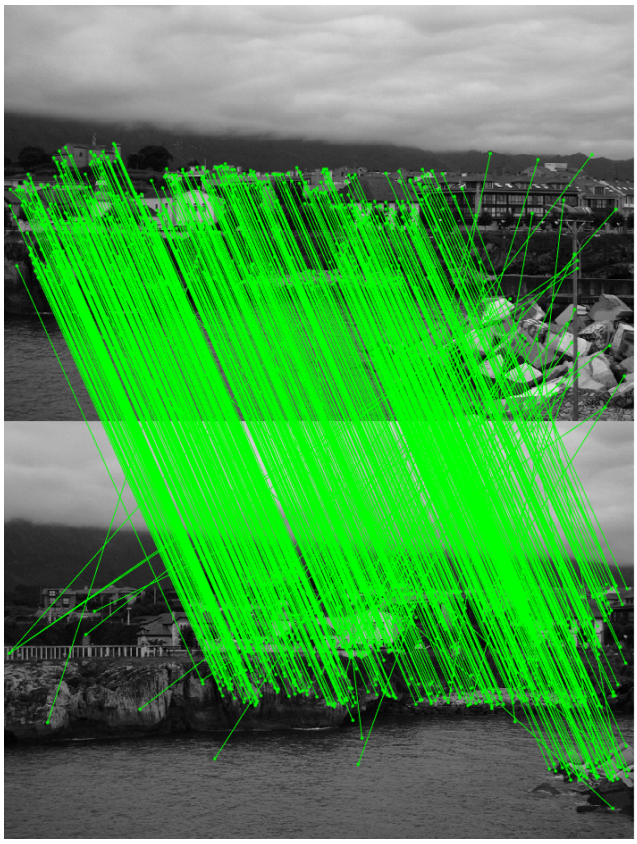

Figure 3: Matches between two images using SIFT descriptors. While most of the lines of the matches present a similar orientation, some mismatches can be seen. These, however correspond to points that are present only in one image due to the motion of the camera.

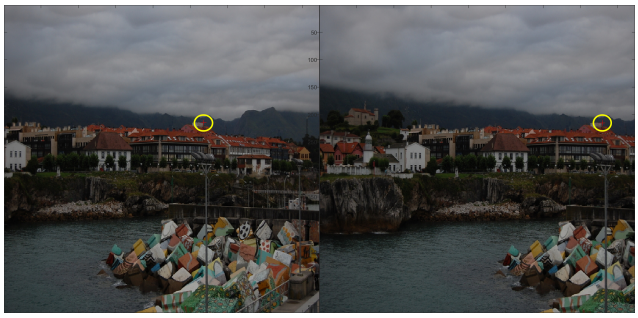

Figure 4: A point match found by directly applying the homography obtained with DLT to a point in the original image.

llanes sequence in which objects are at a sufficient distance for the algorithm to find a suitable homography. Same case of site 22 versus site 13, in this case the tall buildings show the problem. Figures 5, 6, 7 and 8 display such results.

3.4. Task 4

Further improvement of the homographies calculated by the DLT can be achieved through Gold Standard algorithm. Nevertheless, corrections made by this implementation turned out subtle on our experimentation as seen in Fig. 9.

Figures 10, 11, 12 and 13 show the results of GS algorithm. Again, tall buildings in *site 22* and the tractor in *castle* cannot be correctly matched through the homography matrices that best fit each triplet.

3.5. Optional tasks

We have obtained the match for every pair of view-template using the SIFT feature descriptor/extractor provided. To improve the homography and remove outliers we have used a RANSAC approach, which results in a very coherent result, as can be seen in figure 14.

Once we obtained the homographies that relate each view with the template, we were able to build our matrix V, that we used to obtain w using SVD decomposition. From w, applying Cholesky we could obtain our intrinsic parameter matrix K:

$$\begin{pmatrix} 3477.5 & 150.2 & 0.3 \\ 0 & 3386.9 & 0.2 \\ 0 & 0 & 1 \end{pmatrix} \tag{12}$$

The first two elements of the diagonal are the *focal lenghts* for each dimension, x and y. The element at the first row, second column is the *axis skew*, which quantizes the shear distortion present in the image. Lastly, the first two elements of the last column are the *principal point offset*. A perpendicular line to the image plane that passes through the origin point crosses the plane at this point.

As was discussed at section 2.4, we can easily recover the camera poses. This can be illustrated at figures 15 and 16, where the relative position of camera-pattern has been plotted.

Figure 5: Mosaic castle. The position of the tractor w.r.t. the background and the camera does not allow for a single H to match well all pixels.

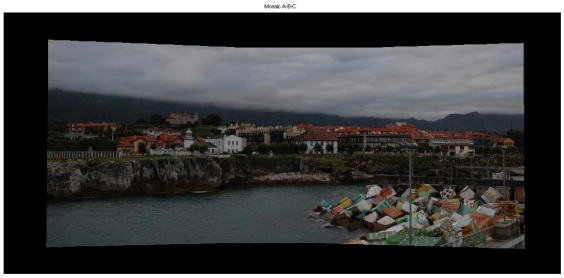

Figure 6: Mosaic of Llanes. The little relative movement between the objects within the scene makes it possible for the homography to perform well.

Figure 7: Mosaic site 13. As in Fig. 6, objects change drastically their position w.r.t. each other and thus, the correspondences can be matched acceptably with H.

Figure 8: Mosaic site 22. The tall buildings of the scene prevent a good correspondence matching as in the tractor in Fig. 5. Several occlusions appear so the non-matching is reinforced. nevertheless low buildings and street are more or less well matched.

Additionally, we have used the computed homographies to plot a very basic 3D shape into the planar patterns. Using the planar pattern as "ground" for our shape, and projecting it from it. To do so we have computed the object points in an ideal case: starting at the origin $(0, 0, 0)$ with a scale of 1, and then transformed by the obtained homographies. A resulting image can be seen at 17

Figure 9: GS points correction.

Figure 10: GS mosaic result on *castle*.

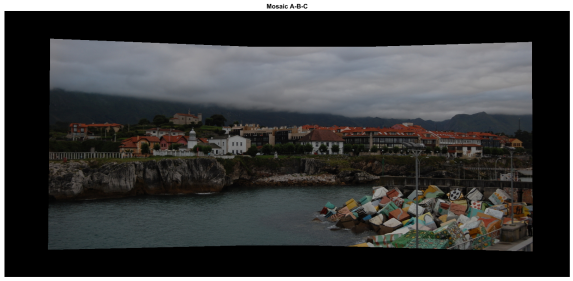

Figure 11: GS mosaic result on *llanes*.

To conclude, we have added the OpenCV logo in one of the graffiti images used for calibration. We have done so manually selecting the points where we wanted the logo to be, and using these points to obtain a suitable homography through the DLT algorithm. The result seems very unnatural, as we are projecting a flat logo in the picture of a slightly tilted wall that can be seen in a tilted laptop in a flat 2D image. This can be seen at figure 18

Figure 12: GS mosaic result on *castle*.

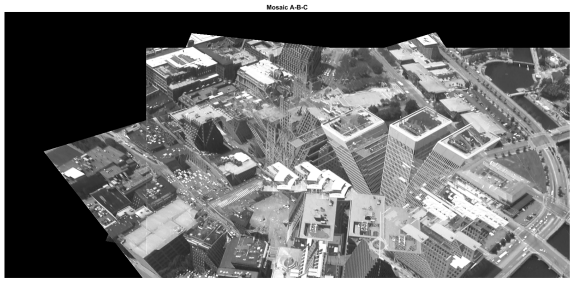

Figure 13: GS mosaic result on *castle*.

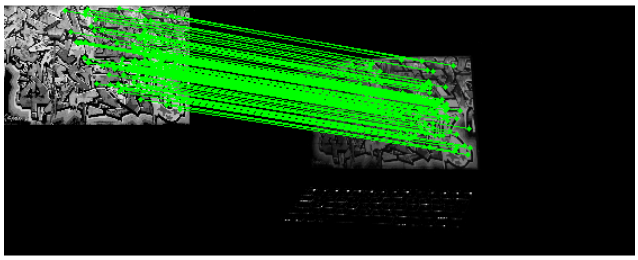

Figure 14: Matches between graffiti3.tiff and the template. The keypoints have been extracted using SIFT.

4. Available code

All the code implemented in this project can be found in the following github repository: https://github.com/marccarne/M6project

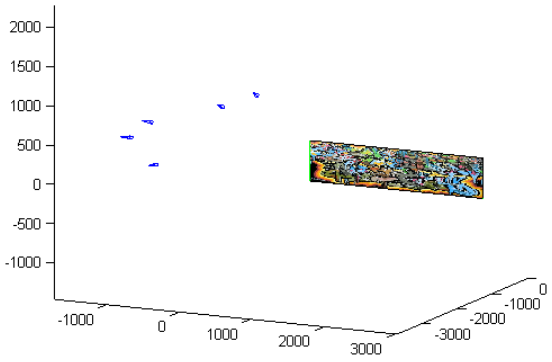

Figure 15: The positions of the cameras with relation to the pattern. The planar pattern is used as the origin point.

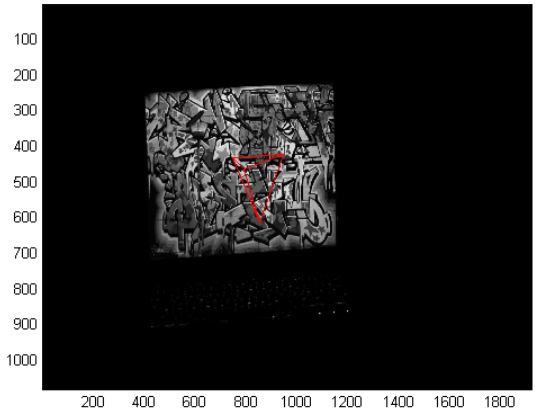

Figure 17: Our simple shape. We decided to go for the pyramid because we have no imagination and are very easy to influence.

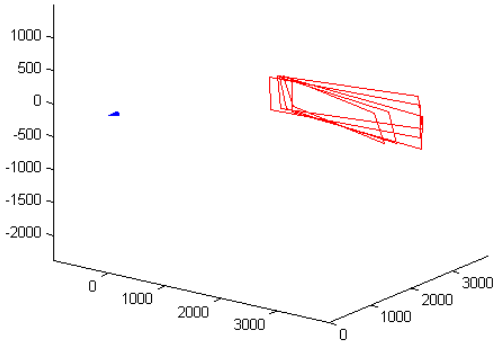

Figure 16: The varying position of the pattern with respect to the camera. In this case the camera is taken as the origin of coordinates and the different positions of the pattern have been registered. The planar pattern has not been rendered to ease the viewing.

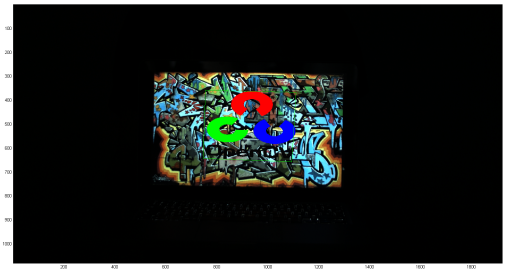

Figure 18: The points selected are the point of union of four different wall tiles. To ease the viewing we have added a green square, as we have added a logo with an alpha channel.