Recovery of 3D Urban Scenes: Planar Transformations and Image Rectification

Álvaro Budria, Alex Carrillo, Sergi Masip and Adrià Molina

Universitat Pompeu Fabra

January 2022

1 Image transformations using Homographies

Homographies, or projective transformations, are a type of linear mapping $h : \mathbb{P}^2 \rightarrow \mathbb{P}^2$ with the only particularity that it preserves straight lines when applied to a set of points in homogeneous coordinates; therefore, homographies may correspond to rotations, translations and scalings to this set of points. A homography may be represented by its *homography matrix* H. When applied on the given set of points x, $x' = Hx$, straight lines formed by $p \in x$ will still be straight in their corresponding $p' \in x'$. This notion of conservation of geometrical properties encoded by a transformation matrix will be discussed later on, in Sec 2.1.
In this work we'll be looking for an *homography matrix* such that when it is applied to the points of a certain image an *affine rectification* is performed as exposed in Section 2.2.

This rectification can be performed by applying a linear mapping on the points of an image I resulting on a new image I' where the projective distortions, such as convergent parallel lines, have been removed. In other words, we'll be looking for a linear mapping H that produces an image I' with a correction of the distortions produced by the projective nature of image perception with respect to the original image I. Several transformations that perform this rectification are tested in Sections 2.2 and 2.3 showing up the performance when rectifying with affine or projective transformations respectively.

1.1 Applying homographies

To correctly apply a homography to an image, the matrix H must be applied to each coordinate of the image. This is called *forward warping* and can lead to holes in the transformed image. A more convenient approach is *backward warping*, which uses the homography H^{-1} to map each coordinate of the transformed image back to the original image, avoiding the creation of holes so that, if a pixel falls in a non-integer location, its value can be interpolated.

To perform backward warping, we must first determine the minimum and maximum coordinates of the transformed image. This can be done by mapping the corners of the original image to the transformed image using H. Once we know the coordinates of the corners of the original image in the transformed image, we can apply backward warping to every coordinate in the transformed image, using the known minimum and maximum coordinates and the size of the transformed image. All coordinates within the rectangle enclosing the transformed image are mapped, even if they fall outside of the original image. Any pixel that falls outside the original image can be filled with black. This process is illustrated in Fig 1.

In the following algorithm Alg 1 we show the procedure for wrapping a certain image into a new projective plane through homographies and homography matrices.

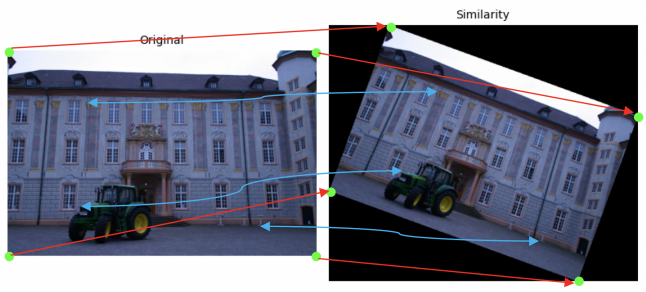

Figure 1: Example of image transformation using homography: (1) Corner mapping with forward warping (red lines). (2) Backward warping applied to all coordinates within min and max values (black frame of image).

Algorithm 1 Applying homographies

Input

$I_{h\times w}$: Input image

$H_{3\times 3}$: Homography matrix

Output

I': Wrapped image

1: $size \leftarrow max(H \times \begin{bmatrix} 0 & h & 0 & h \\ 0 & 0 & w & w \\ 1 & 1 & 1 & 1 \end{bmatrix})$ ▷ Get size by looking at I's corners

2: $I' \leftarrow 0_{x-size \times y-size}$

3: **for** $p' \in I'$ **do**

4: $\quad p \leftarrow H^{-1} \times [p'_{2\times 1}, 1]$ ▷ Find which original pixel corresponds to each new pixel

5: $\quad I'_{p'_x, p'_y} \leftarrow I_{\frac{p_x}{p_z}, \frac{p_y}{p_z}}$ ▷ Assign the corresponding value to the new image

6: **end for**

(a) $\theta = 30$, $S = 1$ (b) $\theta = 175$, $S = 1$ (c) $\theta = 30$, $S = 0.05$

Figure 2: Similarities have four degrees of freedom: angle of rotation, isotropic scaling factor, and two translation coefficients. Notice how angles and relative distances are preserved through the different similarities.

2 Evaluation

2.1 Image transformations

We can categorize geometric transformations based on their invariants. These categorization gives rise to groups of transformations, which are transitive in the sense that *similarities* are a subset of *affinities*, which are a subset for *homographies*.

2.1.1 Similarities

As shown in the results below, similarities are capable of preserving the relative distances and the angles of a set of points. A similarity transformation matrix has 4 degrees of freedom, corresponding to a scaling factor s, how bigger the object will appear, a rotation angle θ, and a translation vector v_x, v_y for moving the object around. These degrees of freedom are reflected in the matrix structure as shown in Eq. 1.

The result of wrapping with this kind of transformation is shown in Figure 2.

$$H_{sim} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

As it's shown in Fig 2 these degrees of freedom can intuitively tweak the resulting image. Note that no variation on the translation vector is shown while results vary the angle θ and the scale S. This is because on visualization we centre the image in the $[0,0]$ of a viewport, therefore no difference would be perceived when translating a complete set of points.

The rest of the parameters in Figure 2 are working as expected, the greater the rotation angle, the more rotated the image appears with a cyclic behaviour from 0 to 360. The scaling factor may seem more confusing to interpret, this 'pixelated' behaviour is kind of related to a sampling process, by scaling the image by a factor of $\frac{1}{20}$ we're actually selecting pixels from the original image Alg. 1 in greater steps and, therefore, undersampling our image.

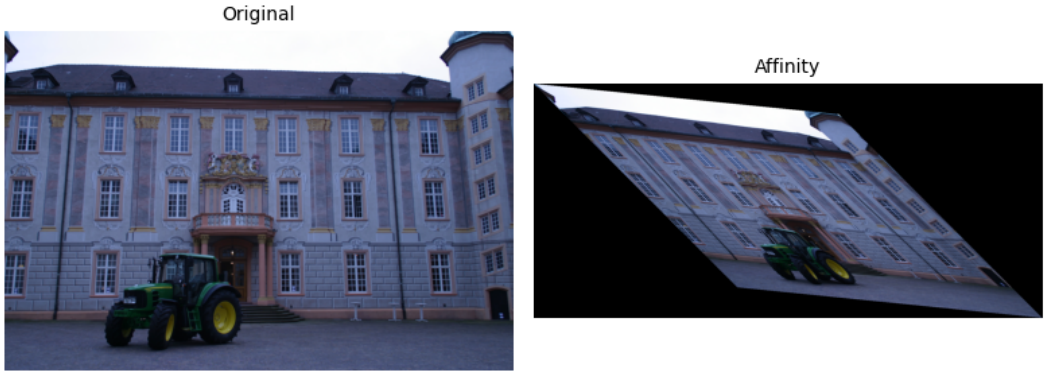

Figure 3: In affine transformations, angles are not preserved, so they may result in perceptually heavier distortions.

2.1.2 Affinities

An affine transformation does not necessarily preserve angles and distances but preserves lines and parallelism. It adds 2 more degrees of freedom with respect to similarities, what could be interpreted as independent rotations and scaling factors for x and y is actually considered as a 2x2 non-singular matrix A_{2x2} which sums up to 6 DOG alongside the translation vector as written in Eq. 2.

Figure 3 shows the result of applying an affinity. Notice how the facade of the building looks 'squeezed' on the vertical axis but elongated on the horizontal. This is a sign that the resulting image isn't a product of a similarity transformation but an affinity with anisotropic scaling factors. Squeezing the image in both directions results in a downscaling; and elongating them, in an upscaling. The fact that S is not constant for every axis may result in this kind of transformation where a different behaviour is shown in each direction, therefore, relative distances cannot be preserved because of non-constant scaling. Actually, this variation on the scaling factor and even the angle is encoded in the matrix $A_{2\times2}$ from Eq. 2 that does not depend on parameters anymore with respect to the analogous similarities.

$$H_{aff} = \begin{bmatrix} A_{2x2} & t_{2x1} \\ 0_{1x2} & 1 \end{bmatrix} \tag{2}$$

Decomposing an Affinity

Affinities may be decomposed into four transformations: two rotations, a scaling, and a translation. The square 2×2 submatrix A in Eq. 2 can be decomposed with a singular value decomposition (SVD): $A = USV^T$. We can rewrite this identity as $A = (UV^T)(VDSV^T) = R(\theta)R(-\phi)SR(\phi)$.

We then expand $R(\theta)$ and $R(\phi)$ with a homogeneous third component, so they become 3×3 matrices. Moreover, we define the translation matrix

$$T = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

With this, we can express H as the product of four matrices: $H = TR(\theta)R(-\phi)SR(\phi)$. We verified that the product of individual steps yields the same image as the original affinity A. We also checked that applying each transformation to the input image step by step and yields the same intermediate results. Fig. 4 illustrates this decomposition. Moreover, the overall decomposition seemed to be numerically stable.

2.1.3 Homographies

As mentioned in the introduction, homographies have the only particularity of preserving straight lines. A result of a applying a projectivity is shown in Figure 5. Homographies present the greatest amount of degrees of freedom (8), since there are no constraints on the 3x3 transformation matrix other than being defined up to scale, therefore $h_{3,3} \in H_{3x3}$ is usually constrained. Note that the increasing amount of DOGs points up to the concept of generalization or transitivity of this transformations, meaning that homographies can express any transformation listed above.

$$H = \begin{bmatrix} h_{1,1} & h_{1x2} & h_{1,3} \\ h_{2,1} & h_{2x2} & h_{2,3} \\ h_{3,1} & h_{3x2} & h_{3,3} \end{bmatrix} \tag{3}$$

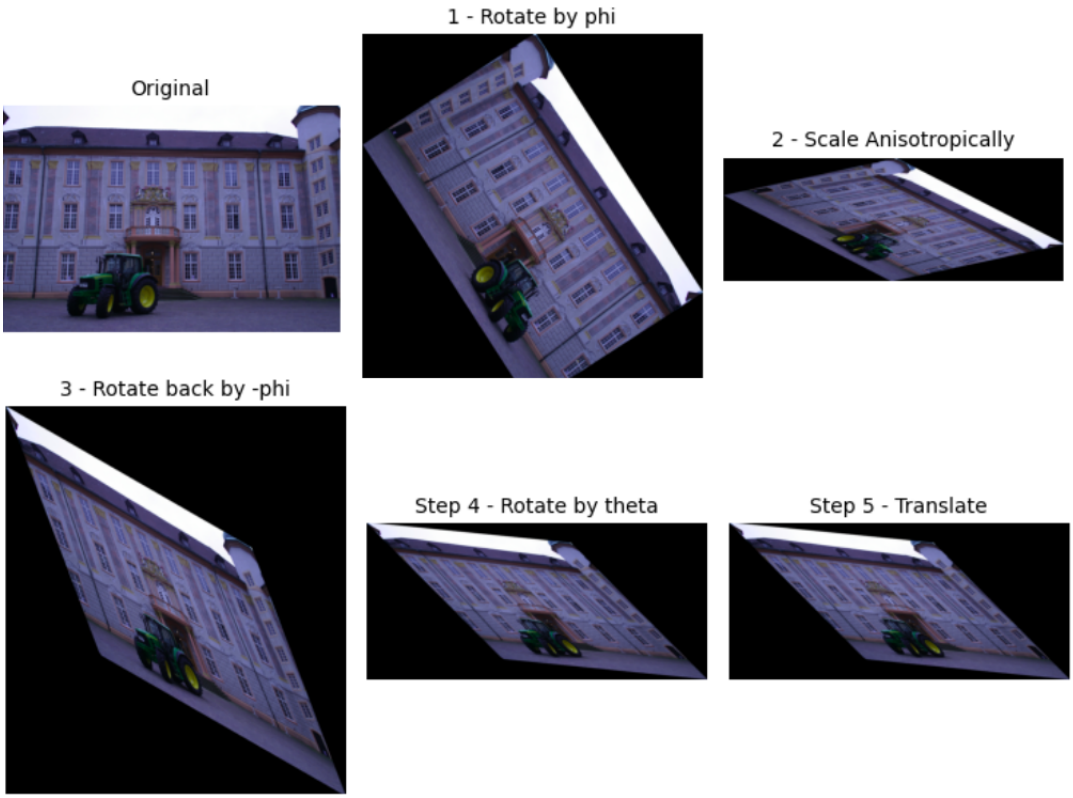

Figure 4: Step-by-step application on an image of the different components of that make up an affinity.

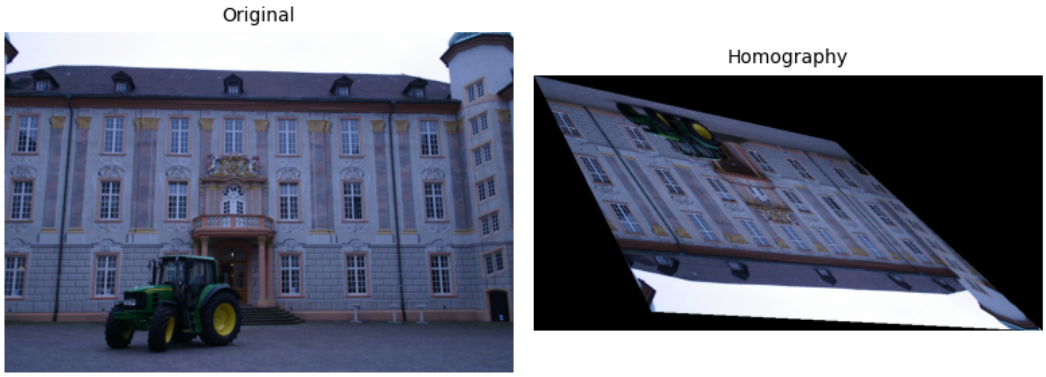

Figure 5: Example of a projective transformation. As it's the generalization for the previous transformation it offers 8 degrees of freedom as it can represent any transformation above as a subset of its potential transformations.

2.2 Affine Rectification

A projective transformation can be decomposed into three components: the similarity H_S, affine H_A and projective H_P components. These components can be written as:

$$H = H_S H_A H_P = \begin{bmatrix} sR & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} K & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} I & \mathbf{0} \\ \mathbf{v}^T & v \end{bmatrix} \tag{4}$$

The goal of affine rectification is to extract the H_P component so that only an affine transformation is applied to the original image. This can be achieved by using a property of affine transformations: they preserve the line at infinity. Since the H_P component can map points $(x, y, 0)$ and lines at infinity to any other point, we can search for the vanishing at infinity in the transformed image and find the linear mapping that moves it to $l_\infty = (0, 0, 1)$.

Given that all parallel lines intersect at a point in the line at infinity, if we have two pairs of lines that are parallel in the original plane, we can define a line using their intersections points. This is the vanishing line $l = (l_1, l_2, l_3)$.

Therefore, we apply transformation H_1

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & l_3 \end{bmatrix} \tag{5}$$

which maps the vanishing line onto the line at infinity. Once applied to the original image, we get the affine rectified image.

2.3 Metric Rectification

As described in the previous section, we used affine rectification to make parallel lines appear parallel in the rectified image.

When an image is transformed only by a similarity, its metric structure is preserved. Metric rectification is the process to restore the image up to a similarity transformation.

Similarly to the affine rectification, we can use the fact that circular points are unchanged under similarities but not under affine transformations. But, instead of working directly with circular points at infinity, we work with their dual conic $C_\infty^* = IJ^T + JI^T = \texttt{diag}(1, 1, 0)$. This is because C_∞^* allows us to calculate angles between lines that are invariant under projective transforms using the following equation:

$$\cos\theta = \frac{l^T C_\infty^* m}{\sqrt{(l^T C_\infty^* l)\,(m^T C_\infty^* m)}} \tag{6}$$

According to Equation 6, orthogonal lines in the original plane should have a cosine of 0, or namely $l^T C_\infty^* m = 0$.

With the definitions from Equation 4 and the conic dual matrix C_∞^*, it is clear how circular points and their dual conic remain fixed under a similarity transformation. The conic is transformed as follows:

$$C_\infty^{*\prime} = (H_A H_S)\, C_\infty^* \left(H_S^T H_A^T\right) = H_A C_\infty^* H_A^T = \begin{bmatrix} \mathrm{KK}^T & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \tag{7}$$

and we do not need to consider the H_P component of Equation 4 because we are assuming that an affine rectification has already been performed.

In order to perform metric rectification, we must find two pairs of orthogonal lines that are linearly independent of each other. When can then name the matrix $S = KK^T$ to rewrite Equation 7 as the following constraint $l^{T\prime} C_\infty^{*\prime} m' = 0$ or:

$$l^{T\prime} C_\infty^{*\prime} m' = \left(l_1' m_1', l_1' m_2' + l_2' m_1', l_2' m_2'\right)\left(s_{11}, s_{12}, s_{22}\right)^T = 0 \tag{8}$$

where s_{11}, s_{12} and s_{22} are the elements of the symmetric matrix S and the system 8 only needs 3 equations to be solved. By fixing one of the elements of S to, for example, $s_{11} = 1$ (an arbitrary scale), we can find the remaining last 2 constraints of the equation by selecting two pairs of orthogonal lines and solving for s. Then, we can use the Cholesky decomposition to obtain the final matrix K from $S = KK^T$. The matrix that we use for metric rectification, $H_{MR} = H_A^{-1}$, is obtained by performing the inverse of the affinity component as shown in Equation 4. The metric rectification process is illustrated in Figure 6.

While visualizing the results of this section, we had some trouble when drawing the rectified lines. For instance, in the left facade (Figure 7) they appeared at the top of the image, though they looked rectified. In order to solve this issue, we updated the c parameter of the lines considering the top left corner coordinates calculated in the forward warping.

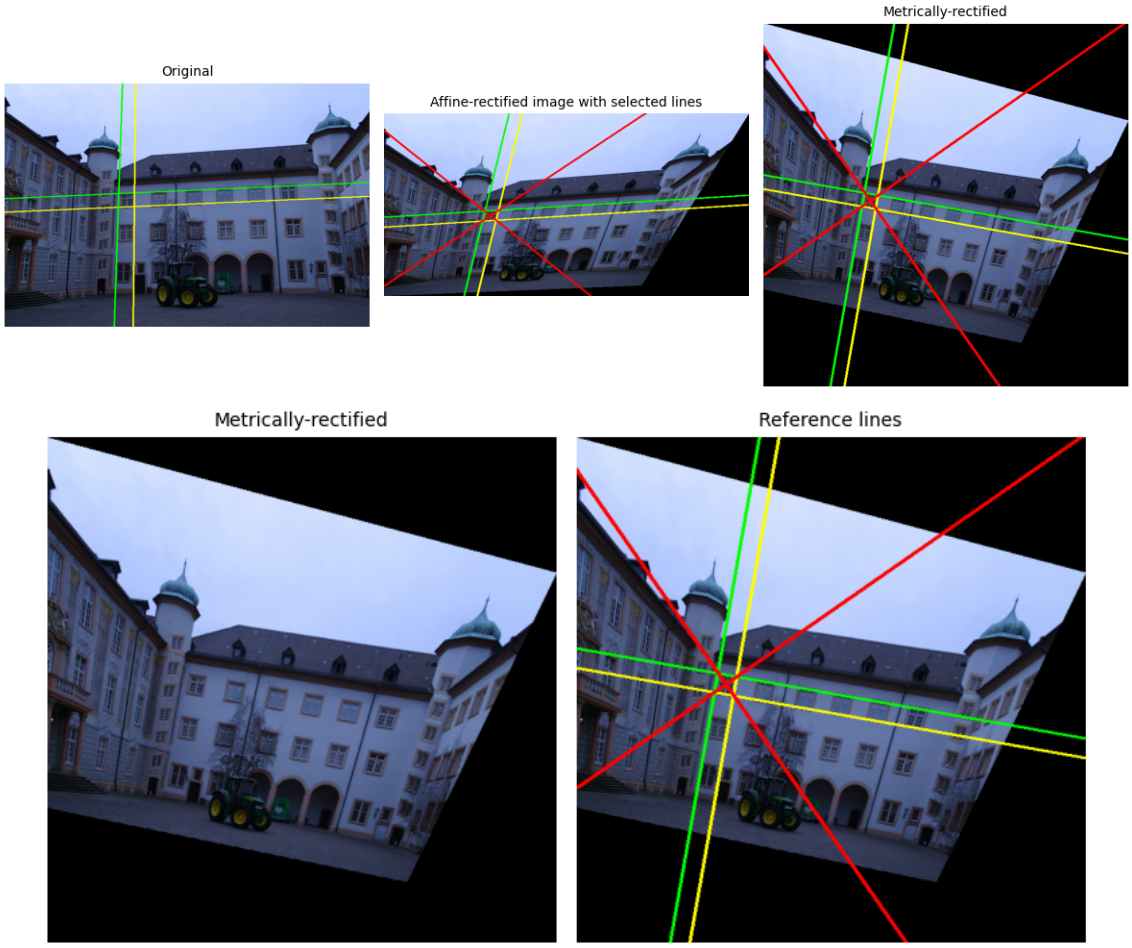

Figure 6: Comparative view of the original, affine-rectified, and metrically rectified images. Notice how in the original image lines are not perpendicular nor orthogonal (though they are in the real world). With affine rectification, parallelism is recovered, and with metric rectification, angles are respected.

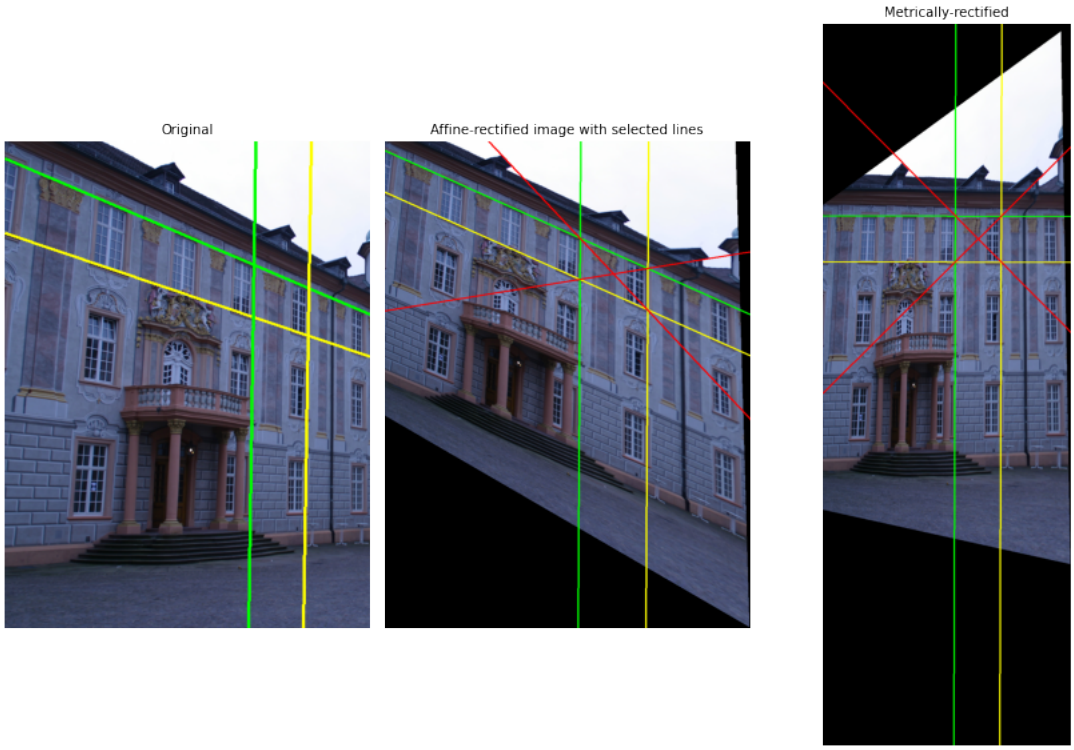

Figure 7: Optimization of the left facade rectification problem.

Image	L1 / L2	L3 / L4	L4 / L5	L5 / L6
Original	4.42	0.12	-	-
Affine	0	0	43.6	124.14
Projective	$8e^{-7}$	0	135	89.99

Table 1: Resulting angles for the left facade image. Green lines: L1 and L3. Yellow lines: L2 and L4. Red lines: L5 and L6

2.4 Affine and Metric Rectification of the left facade of the building.

So far we have rectified the image with respect to straight lines belonging to the frontal facade of the proposed image. This is slightly less challenging due to the fact that the projective distortion is small and the facade is well-posed, resulting in an easy way to rectify the image thanks to being able to properly identify reference lines.

In this section, we rectify with respect lines in the left facade, shown in Figure 7. As it's demonstrated in the figure, the results are surprisingly good despite being a more challenging task. A more precise evaluation is done in Table 1, where we can observe an almost perfect rectification of the selected angles in the image. As expected, affine transformation preserves the parallelism in horizontal and vertical lines, nevertheless, the crossing is not perpendicular as it should be. This is easily achieved by projective transformation where it gets a perfect 90 (with some rounding issues due to using floating point decimals).

2.5 One-Step Metric correction

In this section, we tackle the problem of metric rectification in one step. Unlike in previous sections where we take a stratified approach, here we intend to metrically rectify the image with one single transformation.

The dual conic C^*_∞ compresses all the information required for a metric rectification. We aim at recovering the 8 coefficients that define a dual conic (though the matrix has 9 entries, only ratios are important, so there is one less degree of freedom). Thus, we identify 5 pairs of orthogonal lines and set up an equation per pair of the form

$$l^{T\prime} C^{*\prime}_\infty m' = \left(l'_1 m'_1, (l'_1 m'_2 + l'_2 m'_1)/2, l'_2 m'_2, (l'_1 m'_3 + l'_3 m'_1)/2, (l'_2 m'_3 + l'_3 m'_2)/2, l'_3 m'_3\right) (a,b,c,d,e,f)^T = 0. \quad (9)$$

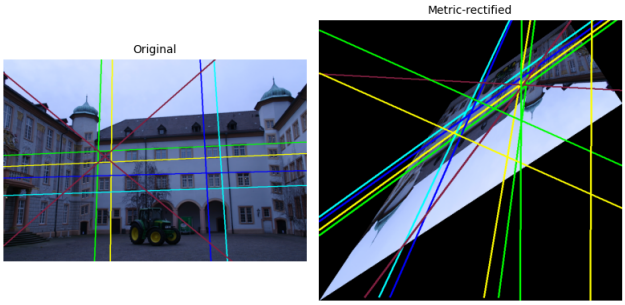

Figure 8: Selected lines and result of the non-stratified metric correction.

We solve the resulting system of equations by computing the null space, obtaining the matrix for the dual conic. Since it is invariant to similarities,

$$C_\infty^{*\prime} = (H_P H_A H_S) C_\infty^* \left(H_S^T H_A^T H_A^T\right) = (H_P H_A) C_\infty^* \left(H_A^T H_A^T\right), \tag{10}$$

and taking the SVD decomposition,

$$C_\infty^{*\prime} = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} U^T \tag{11}$$

we get that the rectifying projectivity is $H = U$.

Despite the diagonal matrix theoretically being of the form

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

we found that due to numerical errors, our estimate did not exactly match this template. This would account for the rather defective results that the homography produces when applied to the projected image.

We also realized that the final result was very sensitive to the set of lines chosen at the beginning of this process, regardless of whether we used the whole image or the left-cropped one. Figure 9 shows the results of applying the non-stratified correcting procedure on the two studied images with different sets of points.

3 Conclusion

Despite the fact that it's been shown a proper rectification of the proposed images, the proposed methods have multiple limitations. It's important to recall that both metric and affine rectification need *ad-hoc* information in order to perform, being this information the segments used as reference. Because of this, the performance of the method in real-world applications is either limited to those circumstances where a human input can be expected or to the performance of the line detection algorithm alongside its own performance.

Nevertheless, in contrast to DL solutions; it provides a robust rectification based on the foundations of image perception and geometry instead of a probabilistic approach which returns feasible or convincing outputs given an image.

Finally, it is worth noting that most of the images produced as a result of an homography present a significant portion of black pixels. Therefore, for a neater result, more post-processing would be required, such as cropping or rotating the image. Although, usually the goal is to take measurements of certain objects in the image, that would not be necessary.

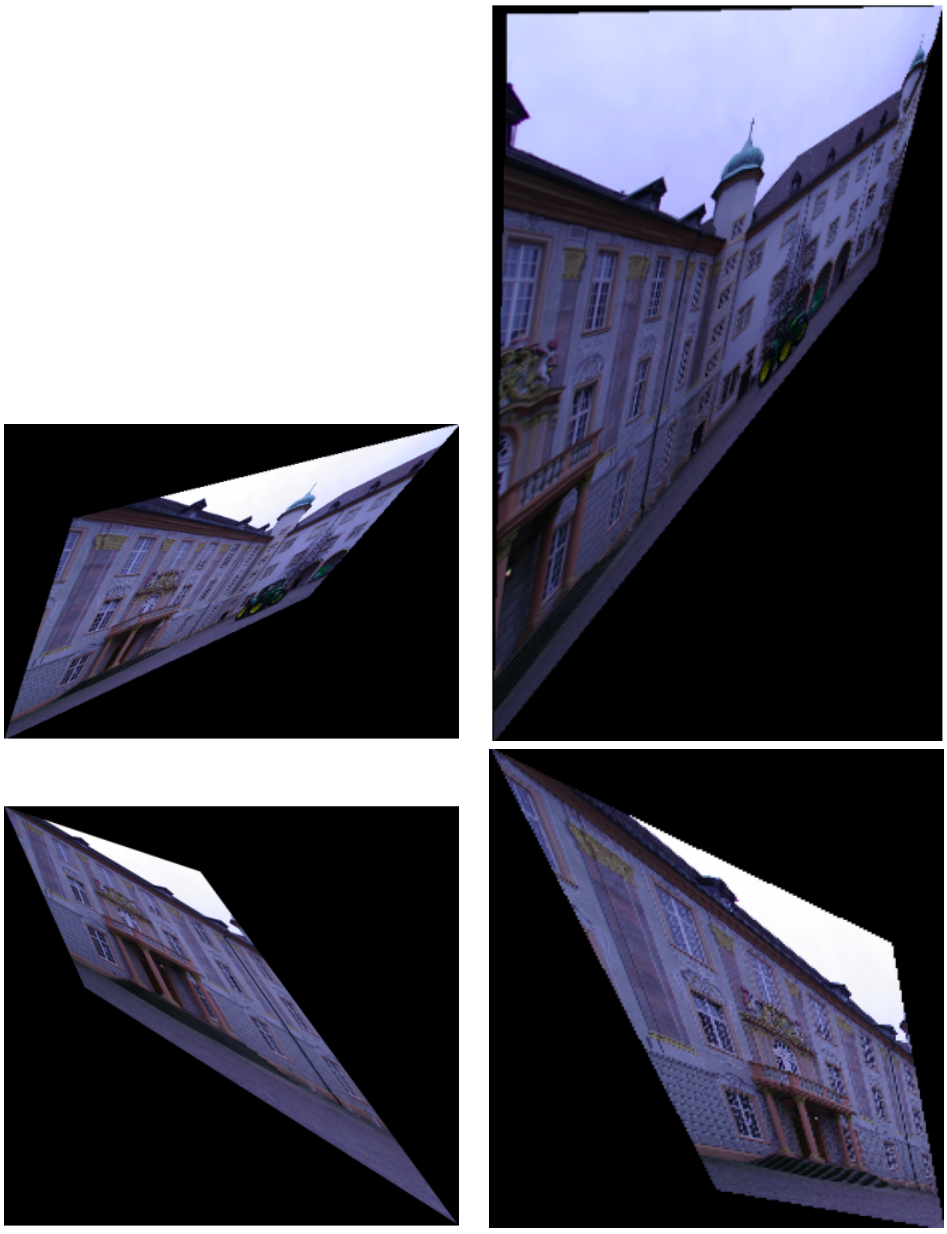

Figure 9: Due to numerical instability and rounding errors in the computation of the null space and the SVD decomposition, as well as errors introduced in the measurement of the lines, the resulting transformation varies sharply depending on the chosen set of lines (left vs. right) and the scene (top: complete vs bottom: left crop).