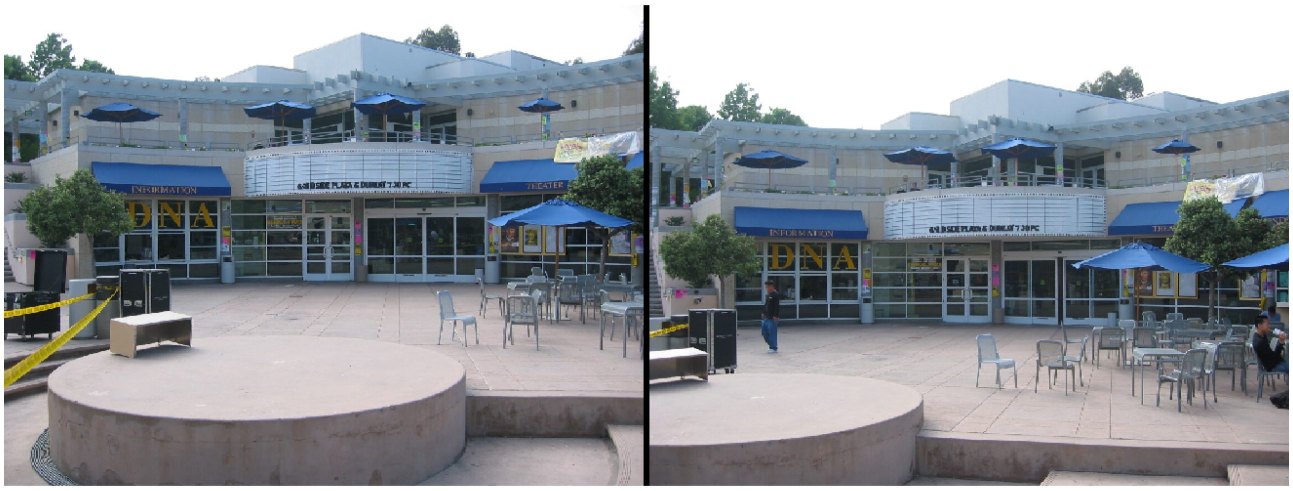
Estimating the Fundamental Matrix for Triangulation

Myron Liu

March 18, 2015

Description

Figure 1: *image*$_1$ (left) and *image*$_2$ (right). Both are 1024 pixels wide and 768 pixels tall.



Suppose we are given the two images above. Our objective is to estimate the fundamental matrix $\boldsymbol{F}$ that maps points in *image*$_1$ to lines in *image*$_2$. To summarize, the action of the fundamental matrix is as follows. For a given point $\boldsymbol{x}$ in *image*$_1$ and letting $\boldsymbol{C}$ be the camera center of *image*$_1$, $\boldsymbol{F}$ projects the ray $\overline{\boldsymbol{Cx}}$ to a line in *image*$_2$. Our general approach for computing $\boldsymbol{F}$ is:

- Detect trackable features in both images.
- Match corresponding features between the two images.
- Perform outlier rejection to throw out false correspondences.
- Estimate $\boldsymbol{F}$ via linear optimization.
- Refine said estimate for $\boldsymbol{F}$ via nonlinear optimization.

1 Automatic estimation of the fundamental matrix

(a) **Feature detection**

As outlined, the first step is to find trackable features. But what exactly does it mean for a feature to be trackable? To gain insight on this matter, let's consider a hypothetical image comprised entirely of straight lines. In this example, it is clear that points along edges are not trackable. After all the point could be anywhere on the line segment in our feature detection window; it is impossible to pinpoint it. In contrast, points at corners (where two lines intersect) are unambiguous. If we translate our window by a bit, or zoom in and out, the corner is still easily picked out. Therefore, we seek *corner-like* features. Of course, images in general cannot be reduced to a collection of lines. However, we can generalize this idea of corner versus edges. What we really want are features that have bidirectional texuredness. The mathematical description of this property is framed in terms the of gradient in pixel intensities (within our feature detection window). The degree of bidirectional texturedness is given by the minor eigenvalue λ_{minor} of:

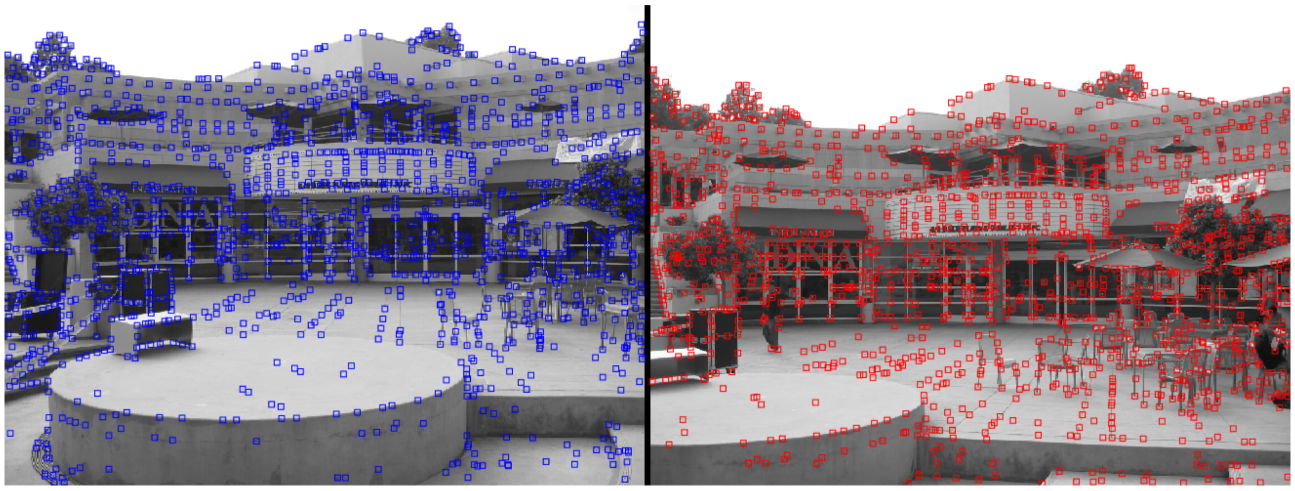
$$\boldsymbol{N} = \begin{bmatrix} \sum_w I_x^2 & \sum_w I_x I_y \\ \sum_w I_x I_y & \sum_w I_y^2 \end{bmatrix}$$

where the summation is over the window, and I_x, I_y are pixel intensity gradients in the x, y directions. We set a threshold t such that features with $\lambda_{minor} < t$ are considered untrackable. In addition, since our window is finite, a given feature may appear in multiple windows. For instance, if our window were a square $w_{detect} = 9$ pixels wide, then a given feature may appear in up to $w_{detect}^2 = 81$ windows. But we only want to track this feature once. Therefore, we perform nonmaximum suppression using a statistical filter of size $w_{suppress}$. If the feature isn't the most corner-like within its statistical window (that is, if its eigenvalue isn't the maximal one), then we do not count it as a new feature. For our implementation, we picked:

- Feature detection window size: $\boxed{w_{detect} = 9px}$
- Statistical maximum filter window size: $\boxed{w_{supress} = 9px}$
- Minor eigenvalue threshold for gradient matrix: $\boxed{t_1 = 0.00016, t_2 = 0.00013}$
  - Note that we scaled our pixel intensities to lie between 0 and 1.
  - As a preliminary guideline for setting t, it is helpful to look upon the mean and standard deviation of the minor eigenvalues.
- note: In general, we may pick different window sizes and thresholds for each image. The motivation for using the same $w_{detect,1} = w_{detect,2}$ and $w_{suppress,1} = w_{suppress,2}$ is that at a glance, the two images seem to be on roughly the same scale. However, we picked $t_1 < t_2$ because $image_1$ appears slightly darker than $image_2$ (see gray-scale images). Assuming that the ratio of intensities is roughly a constant multiplicative factor, and since the length scales are roughly equal, the gradients in $image_1$ will be smaller. It makes sense then to scale t accordingly.

All of this gives $\boxed{1394, 1399}$ trackable features in $image_{1,2}$ (respectively). Going back to our simplified image of straight lines, corners will not generally lie exactly on a pixel. Partly, this is due to the finite accuracy of any image; partly this is due to noise, which will tend to *round* off the corners. To obtain the best estimate for the coordinates of the feature to subpixel precision, we employ the Förstner corner point operator [4]. Shown below are the resulting features to subpixel precision.

Figure 2: Detected corner-like features in $image_1$ (left) and $image_2$ (right) using the aforementioned parameters. The boxes around the features are the detection windows of size w_{detect} centered on that feature.



For additional reference on feature tracking to subpixel accuracy, please refer to Shi and Föerstner [3, 4].

(b) **Feature matching**

To match the trackable features we found, we compare the normalized correlation coefficient (NCC) between every possible pair of windows centered on our features (one feature from each image). Since our features are computed to subpixel accuracy, we interpolate the window image such that the feature being compared is at the window center. This probably won't make a noticeable difference for larger comparison windows. However, for smaller windows such as our choice of $w_{compare} = 9$, the boundary of our comparison window constitute a substantial fraction of the pixels being compared. Moreover, it would not be prudent to make the window so large that it far exceeds the length scale of the features themselves. so this interpolation step is advised.

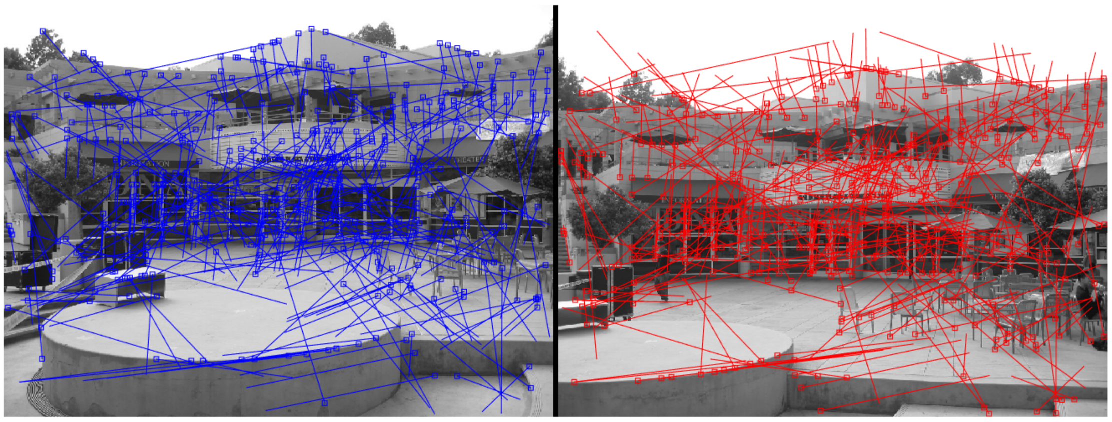
Finding the putative correspondences is similar to the stable marriage problem. However, due to the symmetry of the NCC, this special case is even simpler. Ultimately, the optimal solution is to simply match the two features with the highest NCC, followed by the two features with the second highest NCC, and so on. We also set a threshold on the minimum NCC allowed for a match. Otherwise, if there were a pair of features that are not actually matched, our algorithm may pair them together anyways simply because they have no better matches.

Sometimes, with repeated patterns (*e.g.* window panes) it is difficult for our algorithm to determine which features to match. This is because it does not take the whole context into account. Therefore, we intervene and set a maximum distance d allowed between correspondences in the two images. That being said, there aren't too many such repeated features in our images and the features in the foreground are significantly displaced. Therefore, we were generous in setting d. For our implementation, we chose the following parameters:

- Comparison window size: $\boxed{w_{compare} = 9px}$
- Minimum allowed normalized correlation coefficient between corresponding windows: $\boxed{c = 0.88}$
- Maximum distance allowed between correspondences: $\boxed{d = 500px}$

which gives $\boxed{408}$ matched features.

Figure 3: Matched features. Line segments in $image_1$ (left) point to the matched feature in $image_2$ (right). We subsequently discard outlier correspondences with MSAC in part(c).



(c) **Outlier rejection**

As can be seen in the figure above, some of the matches are fallacious. The way we determine which correspondences are actually valid is via the MSAC algorithm. On each iteration, we randomly pick seven correspondences $\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}'_i$, which we presume to be inliers (*i.e.* true correspondences). Using these, we compute the candidate $\boldsymbol{F}$ via the seven point algorithm. For reference, one can refer to Section 11.1.2 of Zisserman [1]. We use the resulting model of the seven point algorithm in MSAC. We will spare the details here, but for more on MSAC, one can refer to section 4.7 of [1] for a detailed account of RANSAC. MSAC differs from RANSAC in the way the cost function is computed and for this, one can refer to the paper by Torr[2].

In MSAC, the maximum number of random trials before termination of the algorithm is adjusted adaptively via $maxTrials = \log(1-p)/log(1-w^s)$. Below, we give a description of the relevant variables:

- s is the sample size used to compute the model. Here, $s = 4$ since we pick four point correspondences.
- w is the fraction of the number of inlier-pairs over the total number of data-pairs, which varies for each iteration. Note that as w increases, $maxTrials$ decreases, which is the behavior we desire. Namely if at some point we find a model that gives a large number of inliers, then we are probably close to the optimal MSAC solution (really we are trying to minimize the cost, but this constitutes another sort of metric on quality of model), so we decide that it is satisfactory to run fewer total trials.
- p is the assumed probability that at least one of the random samples of three correspondences results in a model with no outliers. Note that if we pick a larger p, $maxTrials$ increases, which is the

behavior we desire. Namely if there exists a superb random sample, then we will probably want to run more random trials to increase the likelihood of finding this treasure. For our implementation, we picked a conservative value of $\boxed{p = 0.99}$.

For the error tolerance, which we use to determine whether a pair of 2D-3D points is an inlier or outlier, we used the following: given the seven-point-computed-model-fundamental-matrix, if the Sampson Error of a given correspondence exceeds $\boxed{F_{m=1}^{-1}(\alpha = 0.95)(\sigma^2 = 1) = 3.8415}$ then the pair is considered to be an outlier. The definition of the Sampson Error (and Correction) can be found in Section 4.26 of Zisserman [1]. Here $F_m^{-1}(\alpha)$ is the inverse chi-squared cumulative distribution function with m degrees of freedom at the probability α, and σ^2 is the variance of the measurement error. $m = 1$ in this case because $\boldsymbol{F}$ has codimension 1. α is the assumed probability that a pair is an inlier. $\boxed{\alpha = 0.95}$ is typical for applications such as ours (although for other applications, α is more appropriately assumed to be other values). Finally, we assumed a value of $\boxed{\sigma^2 = 1}$. In other words, for whatever units (in this case pixels) we used to record the feature coordinates $\tilde{\boldsymbol{x}}_i$, the variance in the measurement error is one such unit.

With this assignment of parameters, we typically find about 200 inliers out of the 408 matches. Since MSAC is random, for the rest of the report we will focus on one particular instance in which our implementation returns the following:

- Consensus set of 209 inlier correspondences (out of 408 matches) found in 496 random trials
- The points correspondences used to compute the consensus model fundamental matrix were:

$\boldsymbol{x}_1$	$\boldsymbol{x}_2$
(298.151552982406,335.812949440606)	(291.877117851226,403.482538645534)
(384.301160899456,355.708362312422)	(369.047460009693,423.135323114275)
(707.671253052960,195.363234417487)	(716.389991952165,263.489730505544)
(832.468414999576,396.729992895593)	(848.295214043126,466.773064896454)
(505.302763025737,69.8886492759519)	(59.4487907340367,164.844345295784)
(751.289169607856,625.238346333362)	(441.489305894843,690.506001096283)
(487.420191364419,429.181841050455)	(476.432500124758,495.719461769196)

where the origin is at the top left of the image, the first coordinate is along the rightward axis, and the second coordinate is along the downward axis. This gives resulting seven point algorithm solution:

$$\boldsymbol{F}_{MSAC} = \begin{bmatrix} 2.11567237784636 \times 10^{-8} & 1.43593286239337 \times 10^{-6} & -0.000669013803210115 \\ -3.66079709065288 \times 10^{-7} & 1.15103565797012 \times 10^{-7} & -0.0111196089973112 \\ 0.0003450241014739 & 0.0103180688946042 & 0.999884655911233 \end{bmatrix}$$

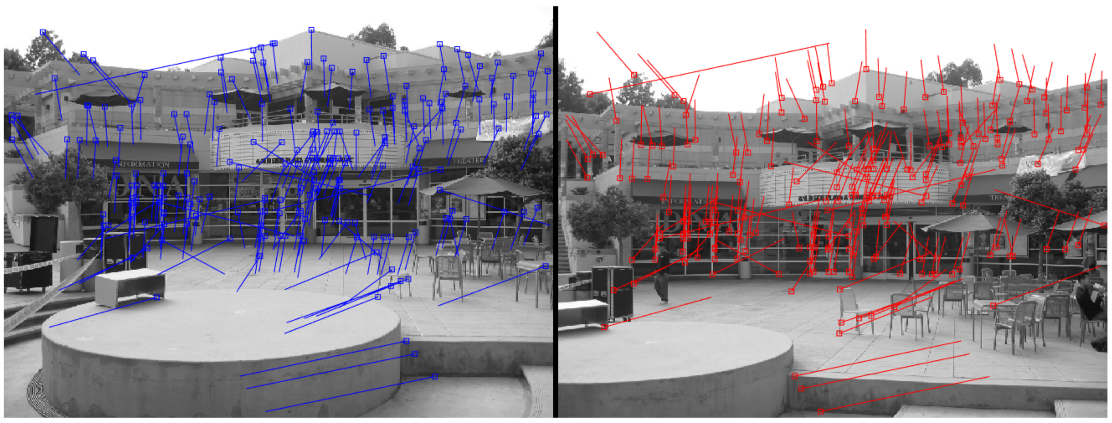


Figure 4: Matched features with outlier correspondences thrown out via MSAC.

(d) **Linear estimation**

Once we have a set of inlier correspondences from running MSAC, we want to refine our estimate for $\boldsymbol{F}$. Some improvements can be made because while MSAC fits the model perfectly to the random sample of seven correspondences, it does not take into account the other inlier points when computing $\boldsymbol{F}$. Therefore, it is in our interest to modify the model such that $\boldsymbol{F}$ may not fit any of the points perfectly, but on the whole minimizes the reprojection error. Before we get there though, it is useful to first compute the model that minimizes the algebraic error $\|\boldsymbol{Af}\|$. Here $\boldsymbol{A} = [\boldsymbol{x'_i}^\top \otimes \boldsymbol{x}_i^\top]$ is the design matrix and $\boldsymbol{f} \equiv vec(\boldsymbol{F})$ such that constraining a given row of $\boldsymbol{Af}$ to zero corresponds to the condition that $\sum_{j=1,2,3}(\boldsymbol{x'}_i^\top \boldsymbol{F}\boldsymbol{x}_i)_j = 0$. To this end, we use the Direct Linear Transformation algorithm, with the intention of using its output to seed the nonlinear optimization; except for some pathological cases, this will place us in the basin of the global optimum for the reprojection error. The DLT algorithm for $\boldsymbol{F}$ is given in section 11.3.1 of Zisserman [1]. With our set of inlier correspondences, the resulting DLT fundamental matrix is:

$$\boxed{\boldsymbol{F}_{DLT} = \begin{bmatrix} -1.85548763329073 \times 10^{-8} & -1.40819946655058 \times 10^{-6} & 0.000653353672989866 \\ 3.8501544574324 \times 10^{-7} & -1.95425036753899 \times 10^{-7} & 0.010935091433029 \\ -0.000335902859635542 & -0.0100866092086065 & -0.999889066039879 \end{bmatrix}}$$

(e) **Nonlinear estimation**

We now use $\boldsymbol{F}_{DLT}$ to seed our Levenberg Marquardt algorithm. The parameter vector that is adjusted at each iteration is $\hat{\boldsymbol{v}} = [\hat{\boldsymbol{p}'}^\top, \hat{X}_1, \hat{Y}_1, \hat{Z}_1, \hat{X}_2, \hat{Y}_2, \hat{Z}_2 ..., \hat{X}_n, \hat{Y}_n, \hat{Z}_n]^\top$. Here, $\hat{\boldsymbol{p}}'$ (eleven degrees of freedom) is the parameterization of overdetermined camera projection matrix $\hat{\boldsymbol{P}'}$ for $image_2$ such that with canonical $\boldsymbol{P} = [\boldsymbol{I}|\boldsymbol{0}]$ for $image_1$, $\boldsymbol{F} = [\boldsymbol{e}']_\times \boldsymbol{P}'\boldsymbol{P}^\dagger$ ($\boldsymbol{e}'$ is the epipole in $image_2$). $[\hat{X}_i, \hat{Y}_i, \hat{Z}_i]^\top$ is the parameterization of the 3D point that projects to the i^{th} inlier pair, adjusted at each step such that together with $\hat{\boldsymbol{p}'}$, the reprojection error is minimized.

We initialize $\hat{\boldsymbol{p}'}$ to the decomposition of $\boldsymbol{F}_{DLT}$ under the canonical camera assumption. Using $\{\boldsymbol{P}, \boldsymbol{P}'_{DLT}\}_{canonical}$, we initialize $[\hat{X}_i, \hat{Y}_i, \hat{Z}_i]^\top$ to the triangulated, optimally-corrected features in the images. Because in the homogeneous representation points at infinity have zero as the last coordinate, we specifically avoid the usual inhomogenous parameterization. Rather, for both $\hat{\boldsymbol{p}'}$ and $\hat{\boldsymbol{X}}_i$, we use the parameterization as outlined in Appendix 6.9.2 of Zisserman [1]. At each step, the LM algorithm modifies the parameter vector such as to minimize the reprojection error (see section 4.2.3 of Zisserman [1]).

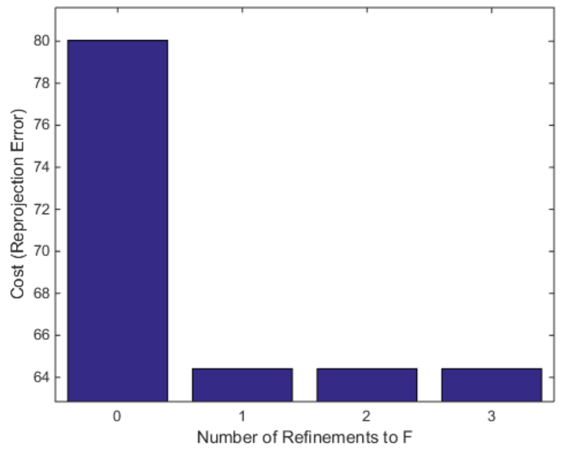
For $n = 209$ inlier correspondences, the Jacobian of $\hat{\boldsymbol{p}'} \mapsto [\tilde{x}_1, \tilde{y}_1, ..., \tilde{x}_n, \tilde{y}_n, \tilde{x}'_1, \tilde{y}'_1, ..., \tilde{x}'_n, \tilde{y}'_n]^\top$ is 836×638, where $[\tilde{x}_1, \tilde{y}_i]^\top$ and $[\tilde{x}'_1, \tilde{y}'_i]^\top$ are the inhomogeneous projected $\hat{\boldsymbol{X}}_i$ in the respective images (Not the measured points $\boldsymbol{x}_i, \boldsymbol{x}'_i$. Our notation is somewhat degenerate). As can be seen, the situation quickly gets out of hand due to the large amount of memory required. Fortunately, the Jacobian is sparse such that it is not necessary to store the entire matrix. Our sparse implementation follows exactly that outlined in Appendix 6.5 of Zisserman with identity covariances [1]. We normalized the data in the same way as is done in the DLT algorithm; so the covariances will be transformed accordingly, but this normalization is for numerical stability and may not be necessary in all cases. Our resulting $\boldsymbol{F}_{LM}$ upon termination is:

$$\boxed{\boldsymbol{F}_{LM} = \begin{bmatrix} -1.89679786464535 \times 10^{-8} & -1.44899815802076 \times 10^{-6} & 0.000663529145278562 \\ 4.04526679133871 \times 10^{-7} & -2.12280456942332 \times 10^{-7} & 0.0109873334793566 \\ -0.000343401630875977 & -0.0101267828847284 & -0.999888078023487 \end{bmatrix}}$$

The nonlinear optimization converges to our termination criteria of $cost_{previous}/cost_{current} > 0.999999$ in three steps with cost sequence:

iteration	cost
0	80.03892
1	64.40920
2	64.40912
3	64.40912

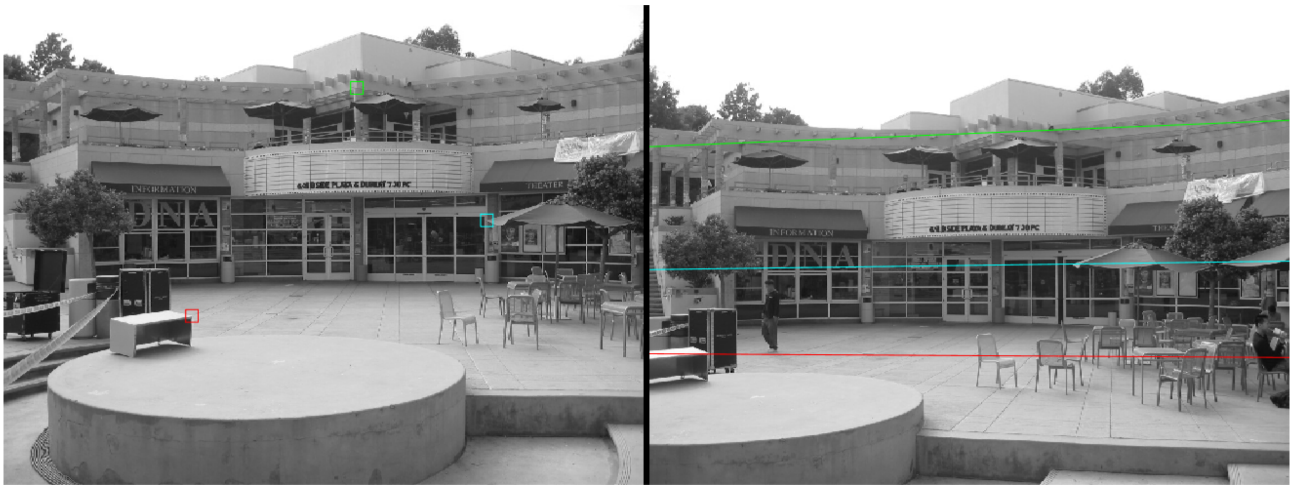
Figure 5: Cost (Reprojection Error) as a function of the number of refinements made to $\boldsymbol{F}$. Refinement zero corresponds to the cost at initialization. Our termination criteria is $cost_{previous}/cost_{current} > 0.999999$.



(f) **Point to line mapping**

As aforementioned, $\boldsymbol{F}$ maps point $\boldsymbol{x}$ in $image_1$ to line $\boldsymbol{\ell}'$ in $image_2$ by *projecting* the ray $\overline{\boldsymbol{Cx}}$ onto $image_2$. Therefore, $\boldsymbol{\ell}' = \boldsymbol{Fx}$ must pass through the corresponding feature $\boldsymbol{x}'$ in $image_2$. To check the quality of $\boldsymbol{F}_{LM}$, it suffices to verify that this behavior holds. Below, we demonstrate that our final estimate for the fundamental matrix is satisfactory.

Figure 6: Three points $\boldsymbol{x}_{1,2,3}$ in $image_1$ and the corresponding epipolar lines $\boldsymbol{\ell}'_{1,2,3} = \boldsymbol{F}_{LM}\boldsymbol{x}_i$ in $image_2$. As it should, $\boldsymbol{\ell}'_i$ passes through $\boldsymbol{x}'_i$ (the feature corresponding to $\boldsymbol{x}_i$) in $image_2$.



References

[1] Richard Hartley and Andrew Zisserman *Multiple View Geometry in Computer Vision Second Edition* 2003: Cambridge University Press.

[2] P.H.S. Torr and D.W. Murray (1997). "The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix". International Journal of Computer Vision 24 (3): 271–300. doi:10.1023/A:1007927408552

[3] J. Shi and C. Tomasi (1994) "Good Features to Track". Computer Vision and Pattern Recognition, 1994. Proceedings CVPR 1994, pp 593 - 600

[4] W. Förstner and E. Gülch (1987) "A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centres of Circular Features". ISPRS Intercommission Workshop, Interlaken, June 1987