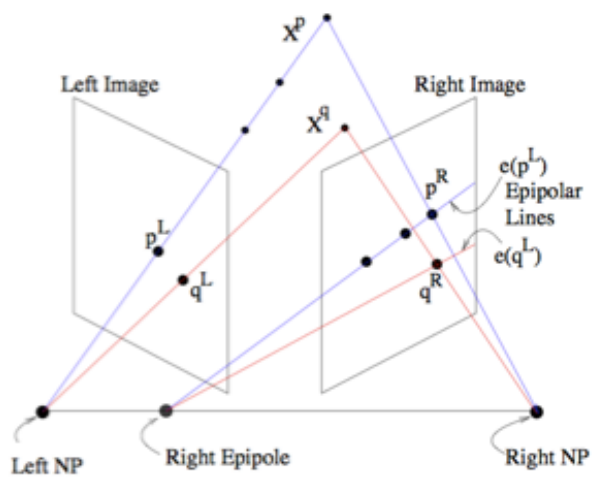
Projection, stereo and panoramic images

X^p and X^q are the physical location of objects.



http://www.cs.toronto.edu/~jepson/csc420/notes/epiPolarGeom.pdf
(note, the image is posted on an educational site and copied here without following up on permissions. Any further use of the image should follow up on the origins and permissions.)

NP are the nadir points of the cameras.

The line connecting the left and right NP is the baseline.

The projection of the left nadir, NP is seen as a the right epipole with w.r.t. the right image. The epipoles may or may not be within the border of the images.

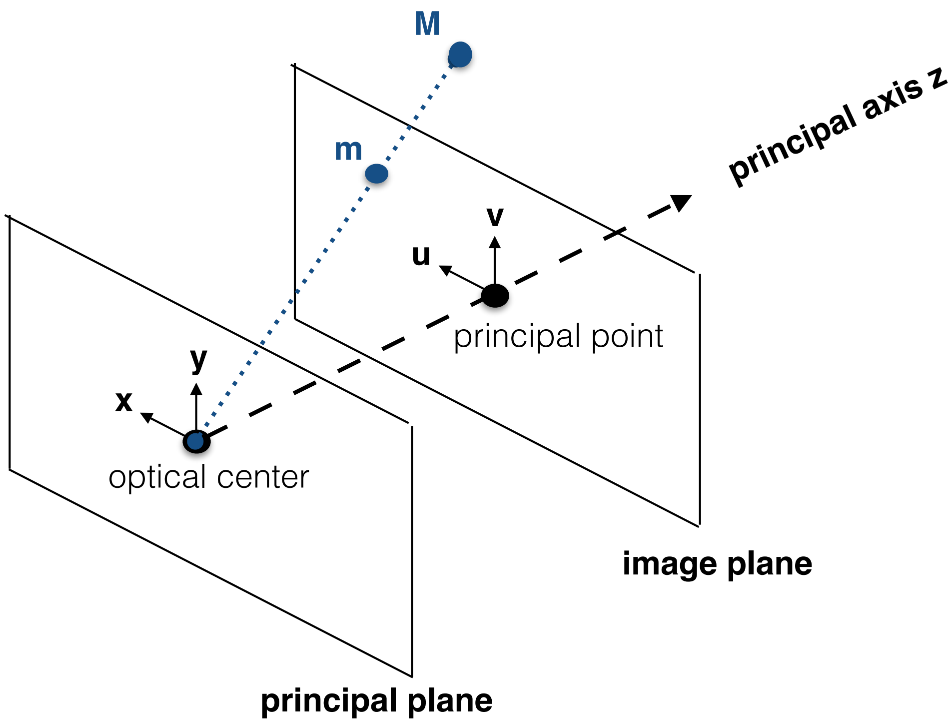
The projection of X^p to left nadir, NP is seen as an epipole line in the right image. If more than one epipole line is present in the right image, they converge at the right epipole (which might not be within the boundaries of the image).

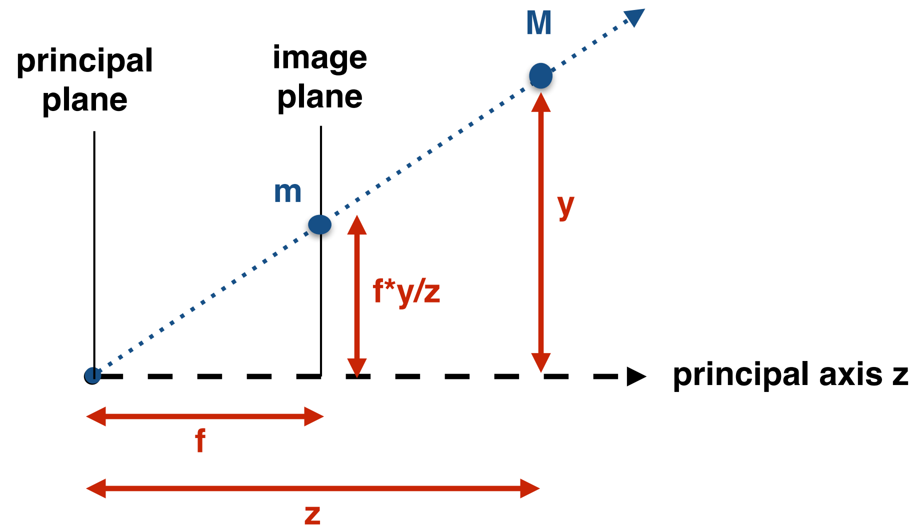
Once the points in the left image, X_L are matched with points in the right image, X_R, if there are at least 7 points, one can determine the "bifocal tensor",
a.k.a. "fundamental matrix, relating the points in the 2 images using a 3x3 matrix of rank 2.

$(X_L)^T * F * X_R = 0$ for any pair or points in the images.

Note: the "Essential matrix" is a matrix used if the camera details are known. The "bifocal tensor", a.k.a. "fundamental matrix" does not need camera details.

Pinhole camera geometry





camera projection matrix P is defined in z*$\mathbf{m}$ = P*$\mathbf{M}$
where M = $(x,y,z,1)^{\mathbf{T}}$ and m = $(f*x/z, f*y/z, 1)^{\mathbf{T}}$

The right side of the projection eqn can be modified by rotation matrix **R** and translation vector **t** to put the coordinates in the (external) world coordinate system. The matrix is $\begin{vmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{vmatrix}$
The six parameters are called *external parameters*.
rows of **R** and the *optical center* describe the *camera reference frame* in world coordinates.

The left side of the projection eqn can be modified to change coordinates in the image frame.

$$K = \begin{vmatrix} f/s_x & f/s_x * \cot\theta & o_x \\ 0 & f/s_y & o_y \\ 0 & 0 & 1 \end{vmatrix}$$

K is the *camera calibration matrix,*
result is pixel coords in image plane.

Pinhole camera geometry (cont.)

intrinsic parameters:
f is the focal distance in mm
o_x, o_y is the principal point in pixel coordinates
s_x is the width of the pixel in mm
s_y is the height of the pixel in mm
ø is the angle between the axes, usually 90 degrees
The aspect ratio s_x/s_y is usually 1

P can be rewritten as P = **K***[I | 0]***G** = **K***[**R** | **t**]

A scale factor ƛ applied is P = ƛ***K***[**R** | **t**]
This is a 3x4 full rank matrix and can be factorized by QR factorization.

P can be rewritten as P = [$\mathbf{P_{3x3}}$ | $\mathbf{p_4}$] where $\mathbf{P_{3x3}}$ is the first 3 rows of P and $\mathbf{p_4}$ is the 4th column.

If ƛ=1, the matrix is normalized and the distance of **M** from the focal plane of the camera is *depth*.

For the image plane at infinity, the image of points doesn't depend on camera position.
The angle between 2 rays is cos θ = $\mathbf{m1^T * \omega * m2 / (}$sqrt$\mathbf{(m1^T * \omega * m1) * (}$sqrt$\mathbf{(m2^T * \omega * m2)}$

extrinsic parameters are the position and orientation of the camera w.r.t. a coord frame.

(Notes on pinhole camera followed by a few notes on multi-view geometry are from http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FUSIELLO4/tutorial.html) multi-view geometry

With point correspondences $\mathbf{m_i} \longleftrightarrow \mathbf{M_i}$, , can solve for camera projection matrix P. Using the Kroenecker delta product and vec operator, coefficients are rewritten to their linearly independent forms of a matrix of size 2n × 12 called the coefficient matrix A

From a set of n point correspondences, we obtain a 2n × 12 coefficient matrix A, where n must be > 6.

In general A will have rank 11 (provided that the points are not all coplanar) and the solution is the *1-dimensional right null-space of A*.

The linear system of equations for inexact data is solved using least squares.
The least-squares solution for vec $P^{\mathbf{T}}$ is the singular vector corresponding to the smallest *singular value* decomposition of A and the direct linear transform.
The equation is usually written in form A * h = 0 where h is vec $P^{\mathbf{T}}$ (its the one dimensional reshaping of the homograph matrix).

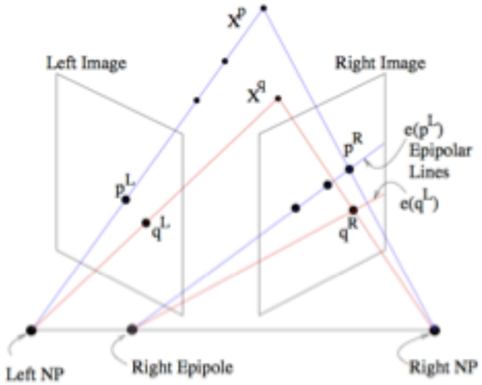
multi-view geometry (cont.)

In general:
— dot product of a point and a line is zero if the point lies on the line
— if the *intrinsic parameters* are known, the relationship between corresponding points is given by the *essential matrix*: $\mathbf{m_r^T * E * m_l^T}$
— when no camera details are available, one can use the *fundamental matrix*.
It's a 3x3 rank 2 homogeneous matrix. It has 7 degrees of freedom.
For any point $\mathbf{m_l}$ in the left image, the corresponding epipolar line $\mathbf{l_r}$ in the right image can be expressed as $\mathbf{l_r = F * m_l}$ and vice versa $\mathbf{l_r = F^T * m_r}$
— the *essential and fundamental matrices* are related through $\mathbf{F} = \mathbf{K_r^{-T}} * \mathbf{E} * \mathbf{K_l^{-T}}$ where Kr and Kl are the left and right camera matrices

Regarding 3D reconstruction:
— when both *intrinsic* and *extrinsic* camera parameters are known, the reconstruction is solved unambiguously by triangulation.
— when only *intrinsic* parameters are known, extrinsic parameters can be estimated and the reconstruction can be solved up to an unknown scale factor, that is **R** can be estimated, but **t** can be only up to a scale factor. the epipolar geometry is the essential matrix and the solvable projection is euclidean (rigid+ uniform scale)
— when neither *intrinsic* nor *extrinsic* parameters are known, i.e., the only information available are pixel correspondences, the reconstruction can be solved up to an unknown, global projective transformation of the world.

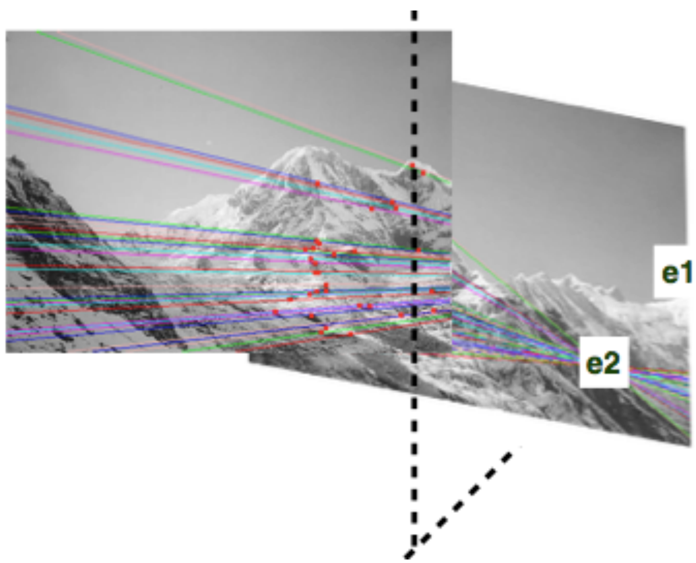
In Trifocal geometry, a trifocal tensor is used.

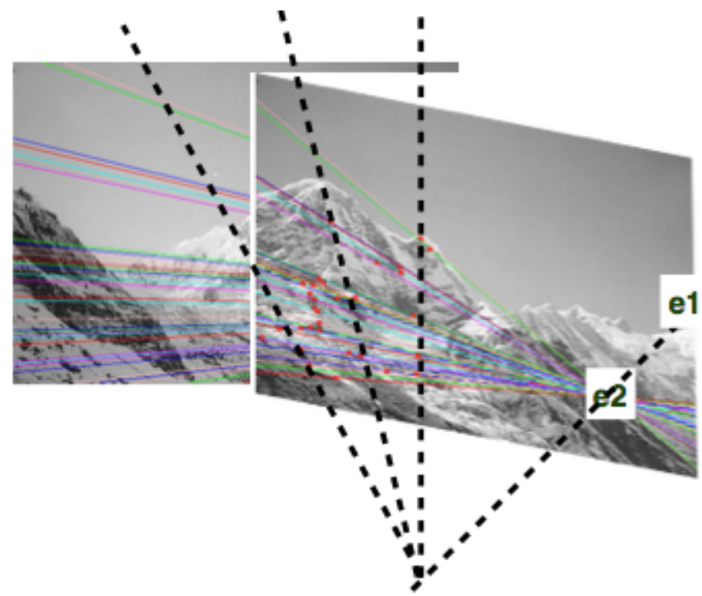
Projection, stereo and panoramic images



http://www.cs.toronto.edu/~jepson/csc420/notes/epiPolarGeom.pdf
(note, the image is posted on an educational site and copied here without following up on permissions. Any further use of the image should follow up on the origins and permissions.)

panoramic images here are from Brown & Lowe 2003



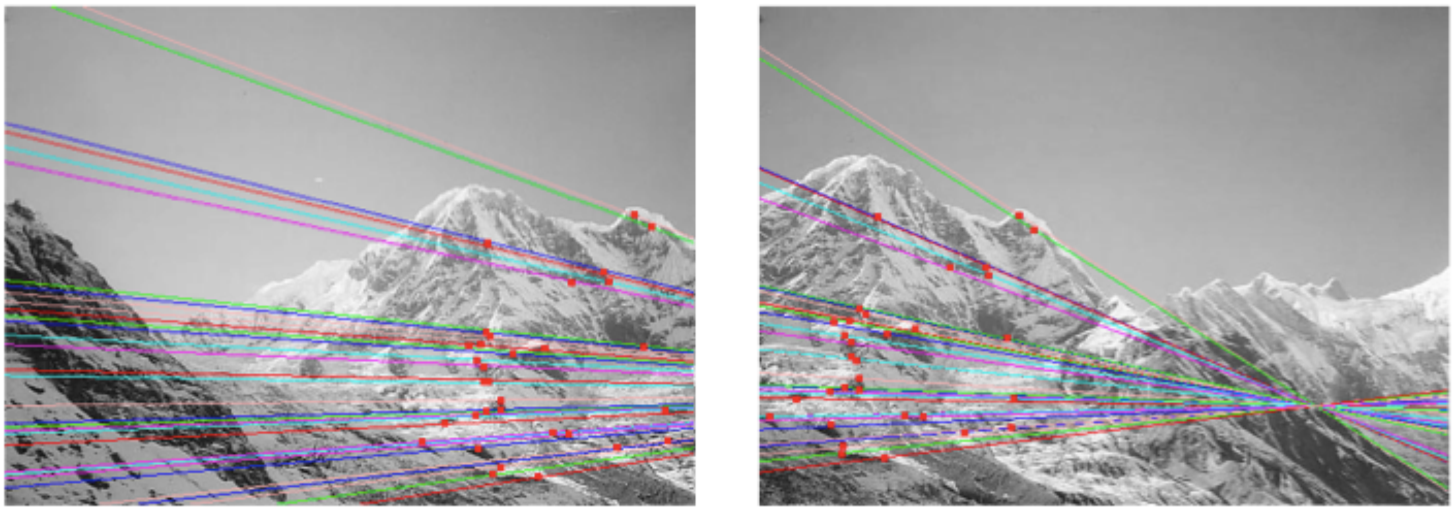


same camera objectives (nadirs), but different orientation for the 2 images (that is, rotated around the same nadir)

Point Correspondence

Need to create list of matched points between the images in order to solve for the epipolar projection

manually making a point list from the corners from the edge extractor used with “outdoor mode”:



epipolar fit to 32 points already known to match shows what the epipolar projections should be when the corner find + corner match + stereo projection solve are correctly automated.

```
nMatched=32
avgDist=0.281
stDev=0.508
```

tried various means of creating point lists and solving for matches between two images.
— regardless of the means of extracting the points, the coordinates of the points alone were not enough information to quickly solve the transformation between one image frame and another consistently.

"feature descriptors" were needed. greyscale image intensity descriptors are usually enough for matching, and gradient images help to determine the "orientation angle" of a coordinate. the "oriented features" of one image are compared to those of another image to find the best matches (a match being the sum of squared differences (SSD) of their oriented descriptors, and that result being less than their individual auto-correlation errors).
— the methods tried for creating points to match between images were inflection points from scale space contours, corners from canny edge filter edges, corners from the perimeters of blobs extracted from specially tailored segmentation, and points extracted as the largest value points from second derivative gaussians of the greyscale images.
— various filters were tried to reduce the points of interest to the most unique for matching.

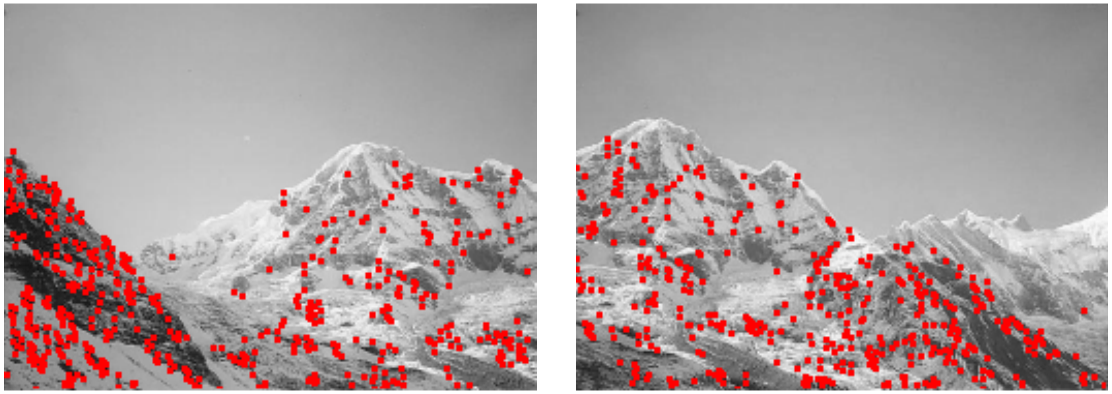
— a filter to remove points which have a close 2nd best matching point (close being an SSD within a factor of 0.8).

— a filter to remove difficult to localize features made using the eigenvalues of the descriptors works well too.

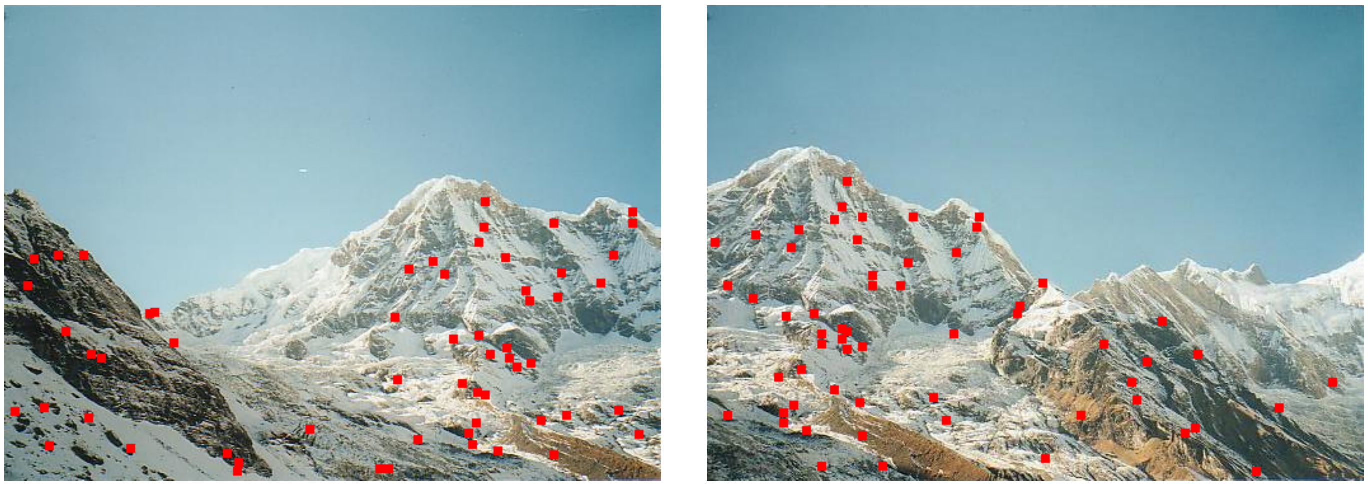
— a filter to remove points with close values of cosine similarity was tried.

The 2nd best filter and localizability filter were found to be very effective.
— the ransac algorithm was then created to use a euclidean or epipolar model for random small samples of points with an evaluator of various distance methods both of which work for rotated, translated, and scaled images.

the results are matched correspondence lists.

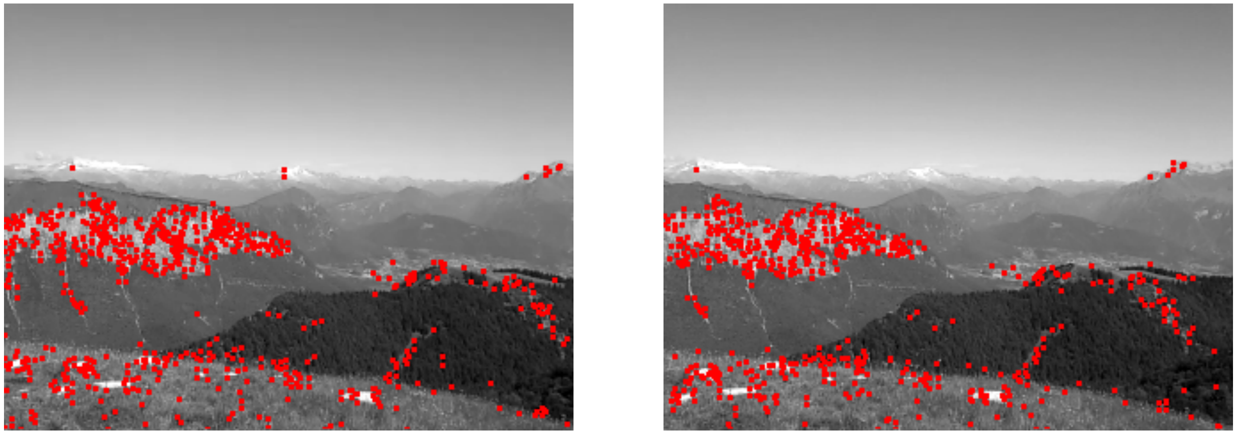


2nd deriv points

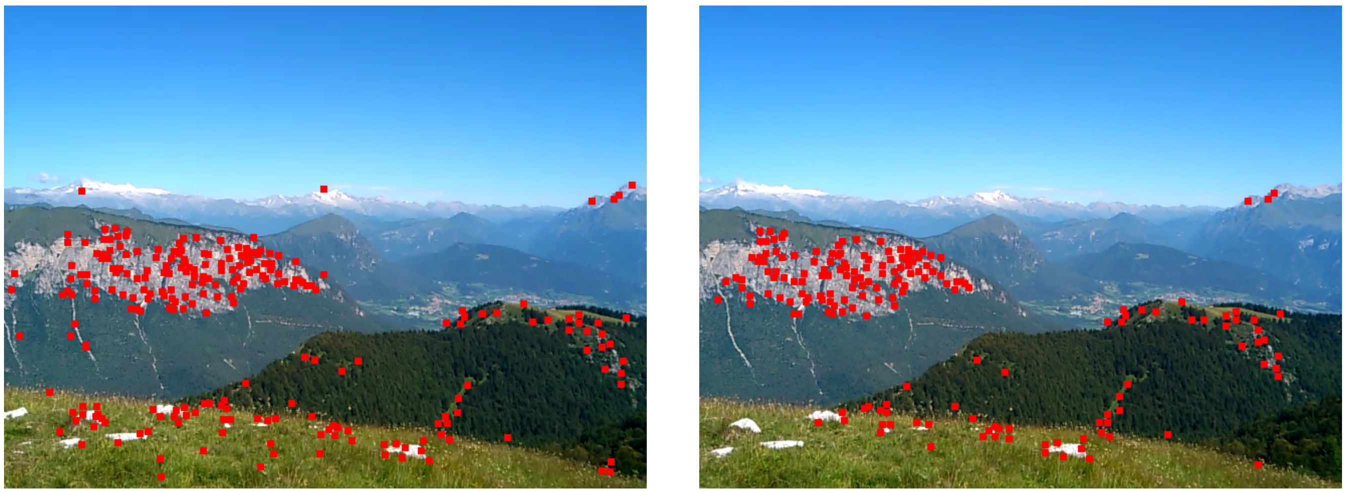


points filtered for those w/o close 2nd best match, then filtered
for localizability.

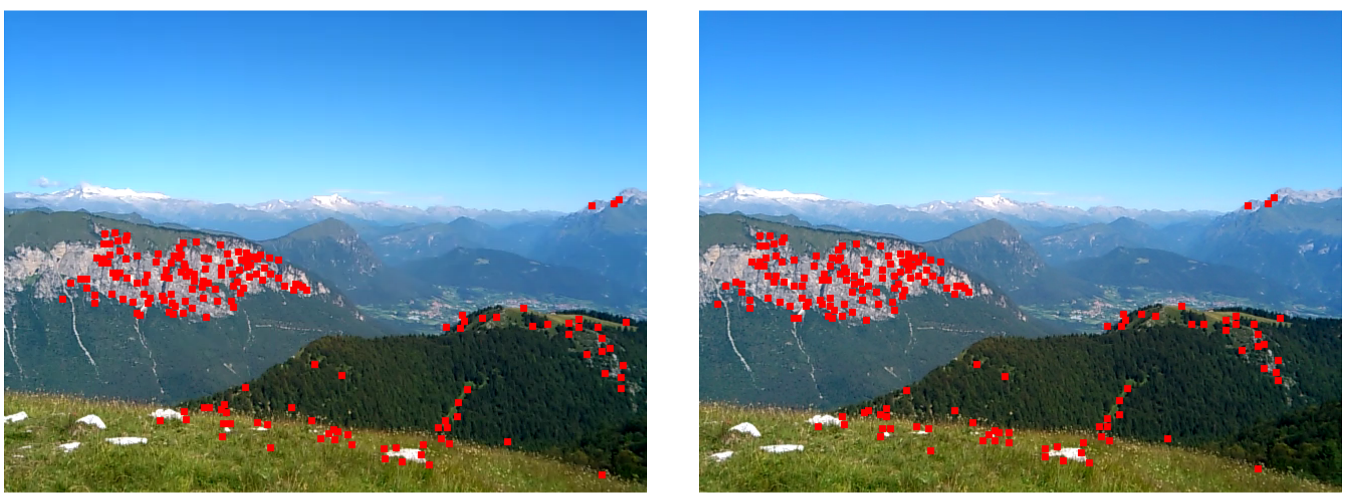
matched by RANSAC algorithm



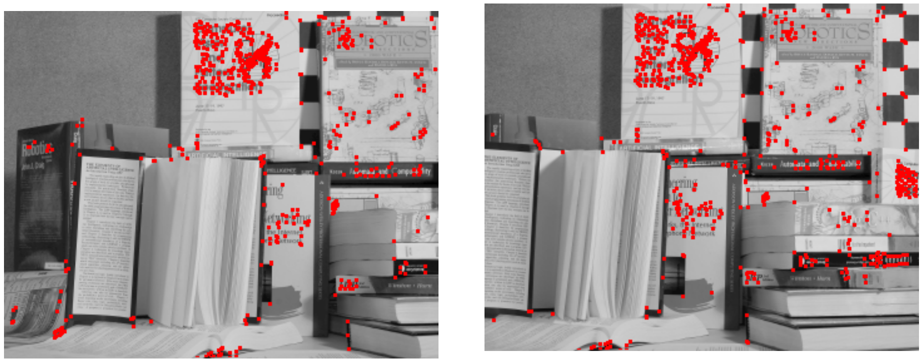
2nd deriv points



points filtered for those w/o close 2nd best match, then filtered for localizability.



matched by RANSAC algorithm

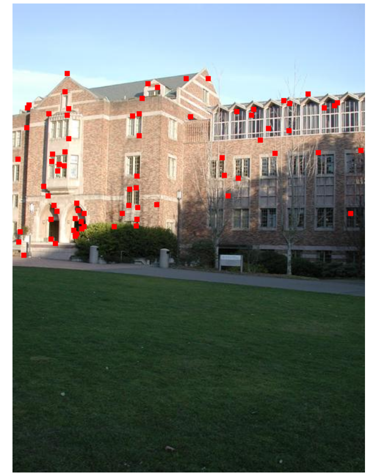


2nd deriv points

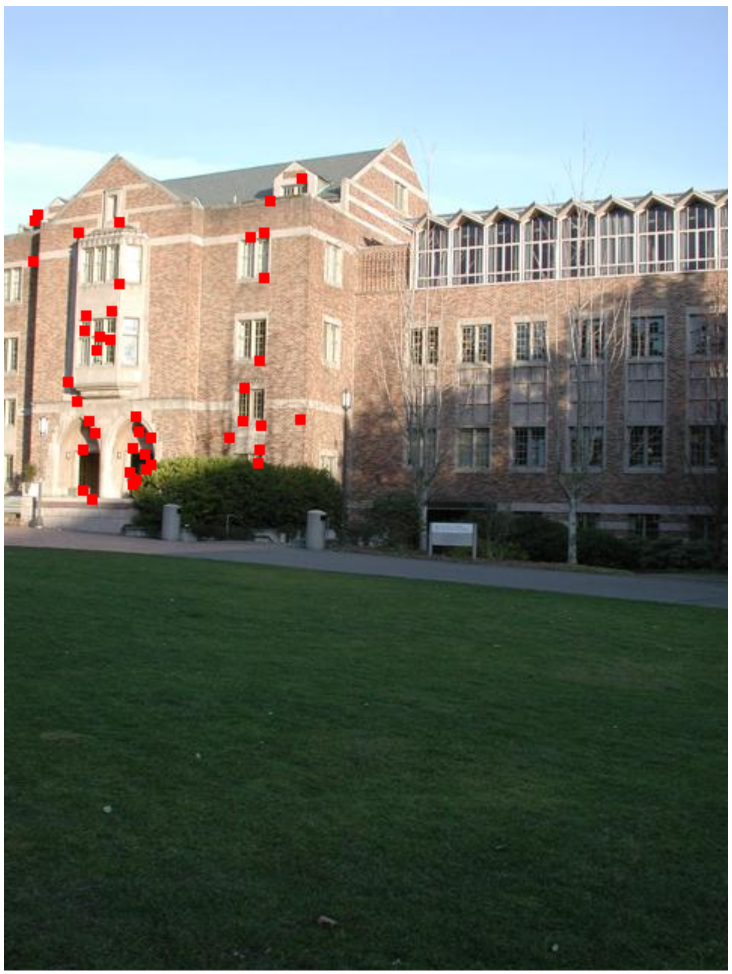
points filtered for those w/o close 2nd best match, then filtered for localizability.

matched by RANSAC algorithm


2nd deriv
points



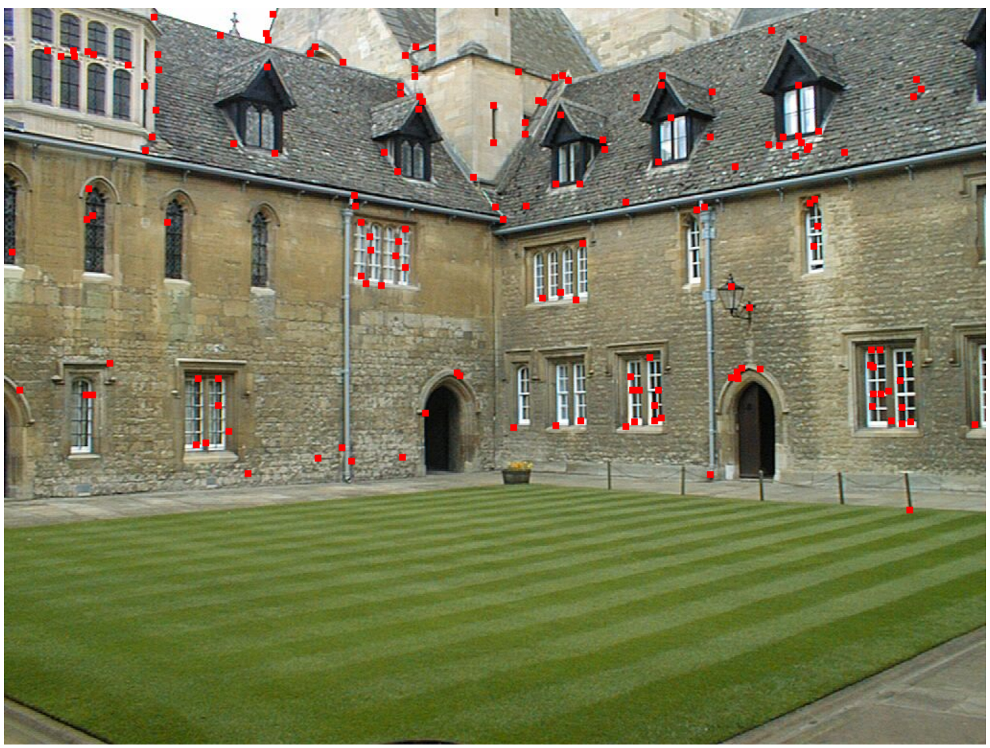

points filtered for
those w/o
close 2nd best
match,
then filtered
for localizability.

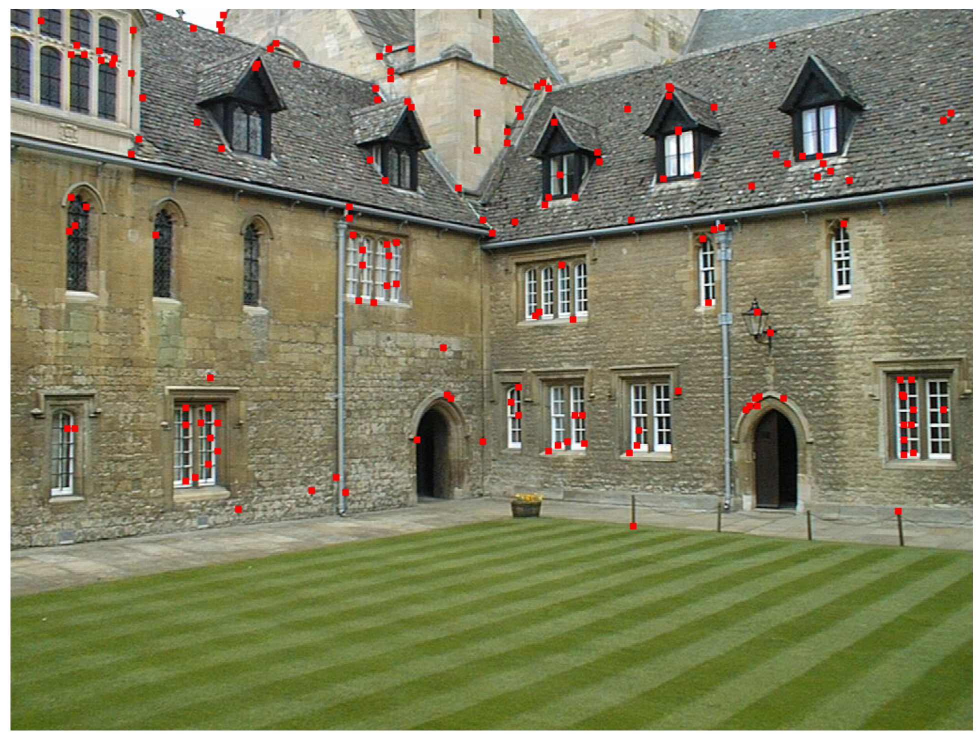



matched by
RANSAC
algorithm

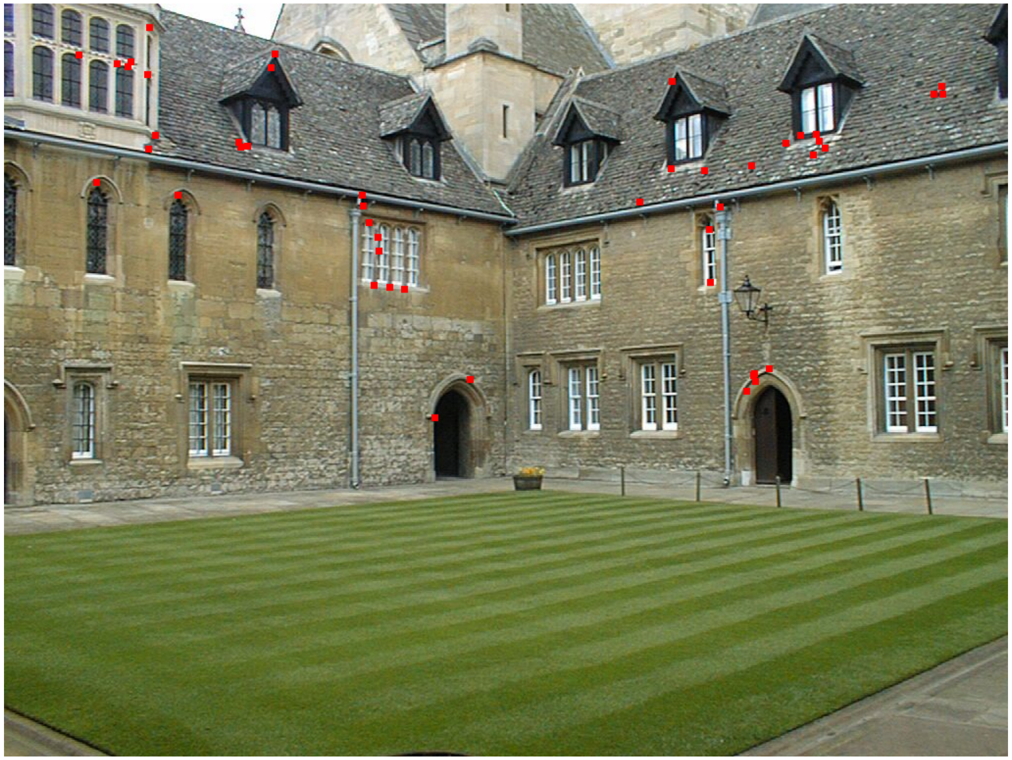
2nd deriv points

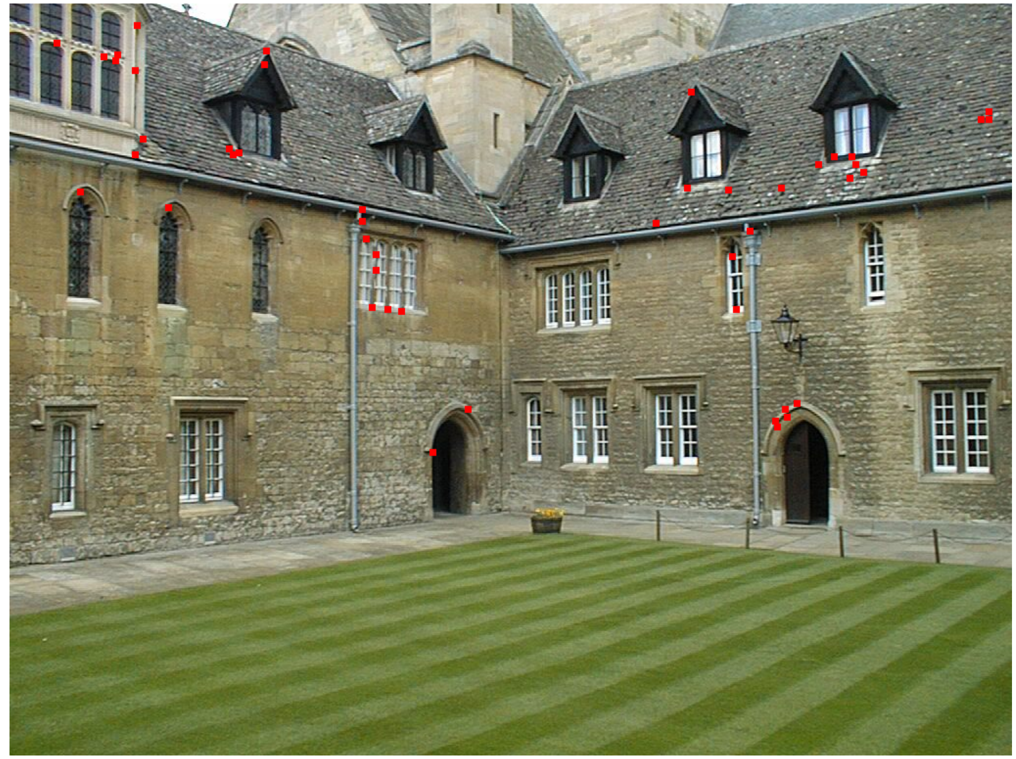
points filtered for those w/o close 2nd best match, then filtered for localizability.



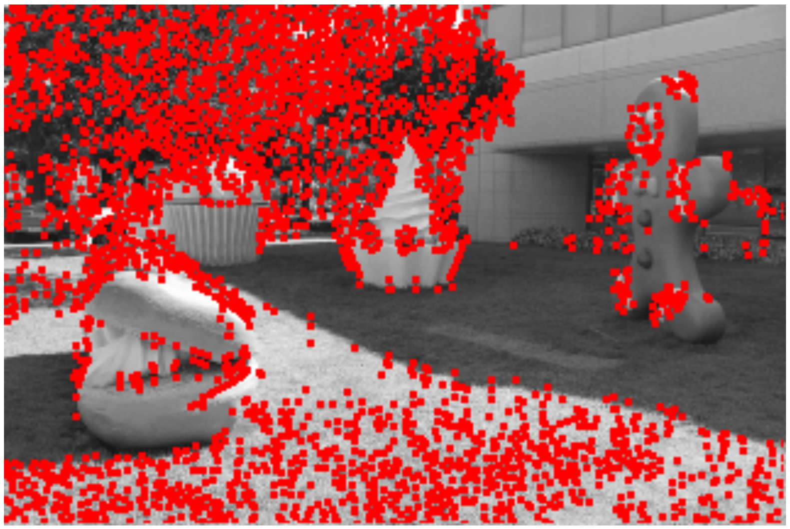


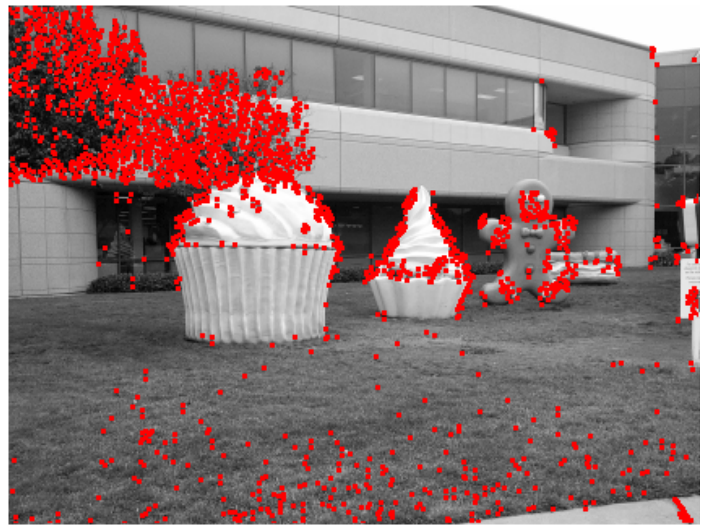
matched by RANSAC algorithm



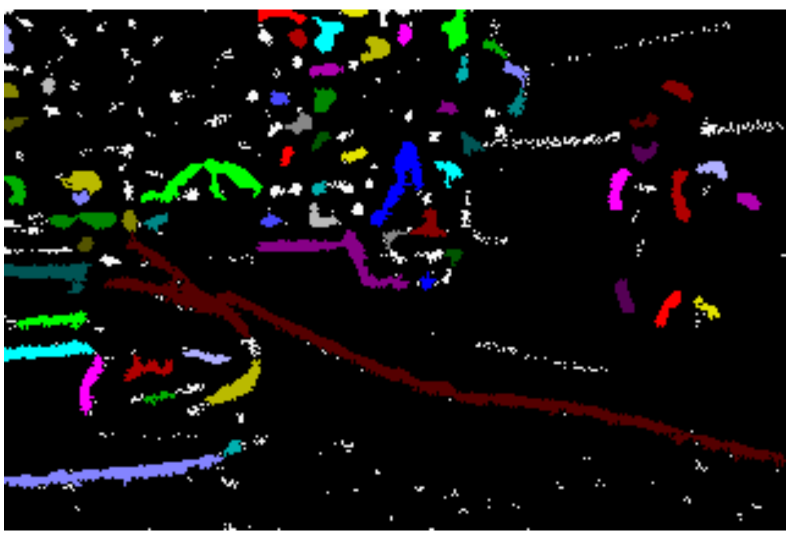


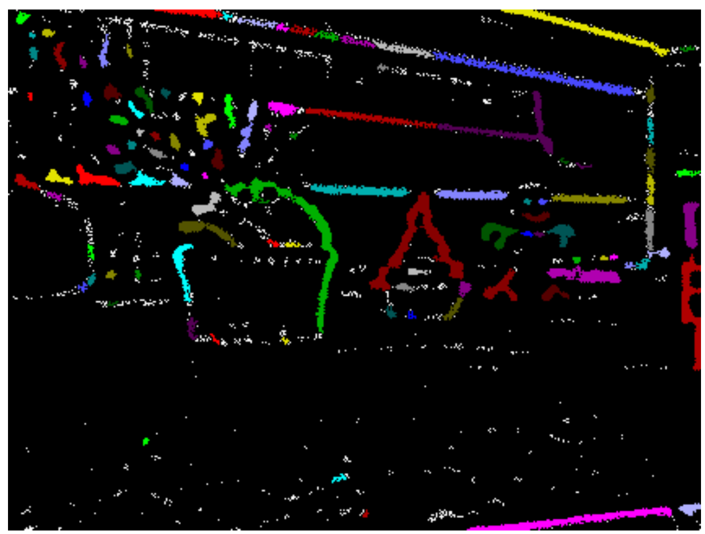
for these images w/ many many points in textures such as the trees and in grass, it's necessary to increase the limit for 2nd derivative points to get points in the less textured areas, but then it is also necessary to limit the textured points by using association with wavelet based segmentation or another technique.



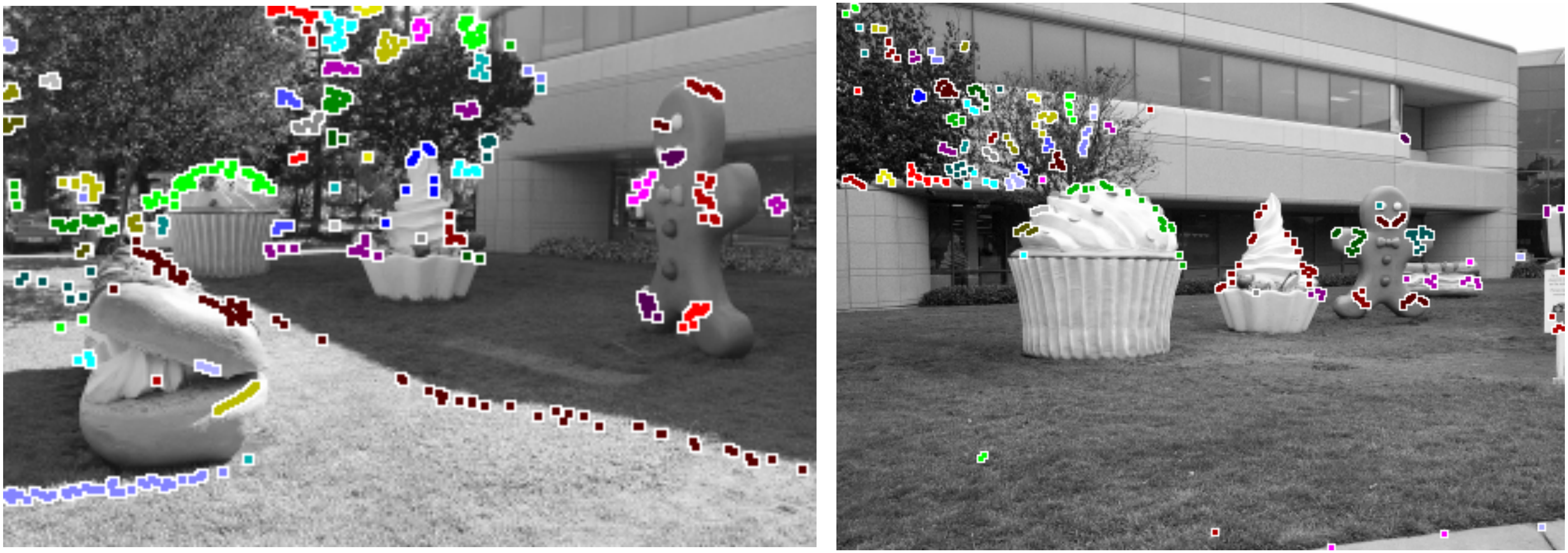


extracting about 5000 2nd derivative points in both images





blobs from a trous wavelet based segmentation

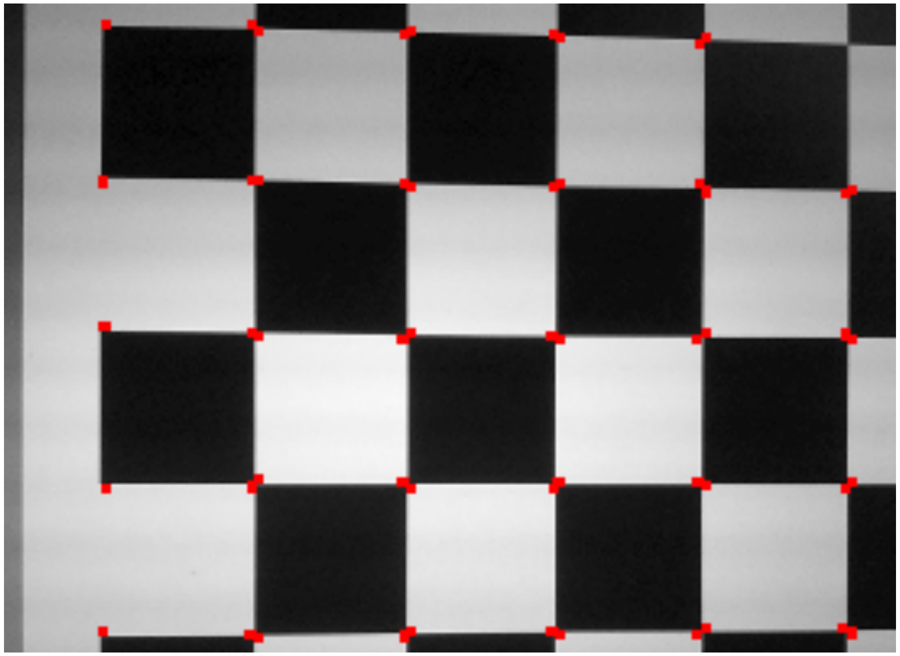
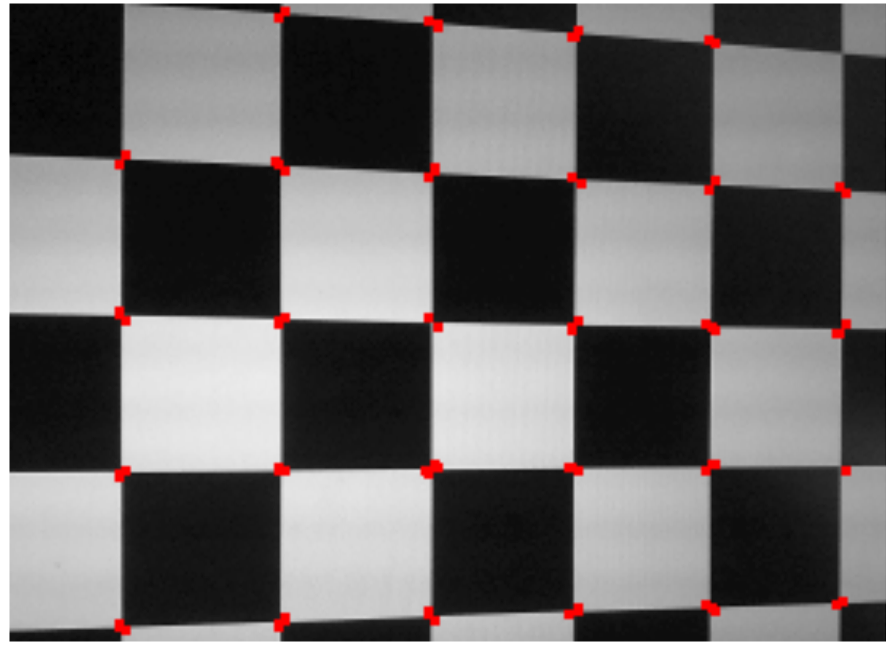


filtered to only points associated with the blobs.

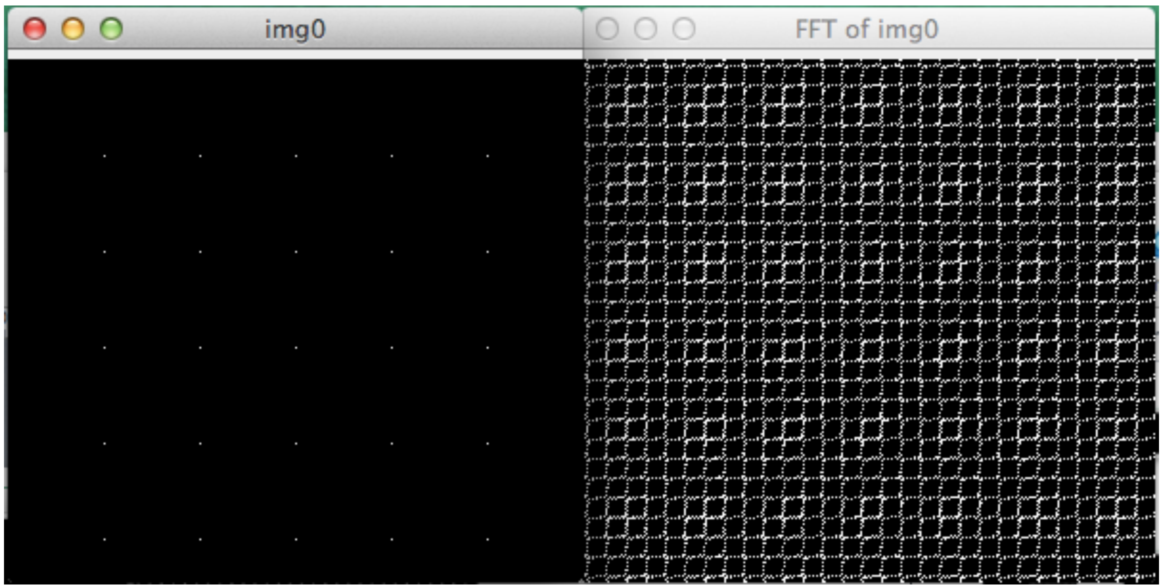
— can see that with color, it should be possible to match the gingerbread man.
using color instead of greyscale would presumably bring out points in the tie and buttons,

note that further filtering for localizability leaves too few true matches.

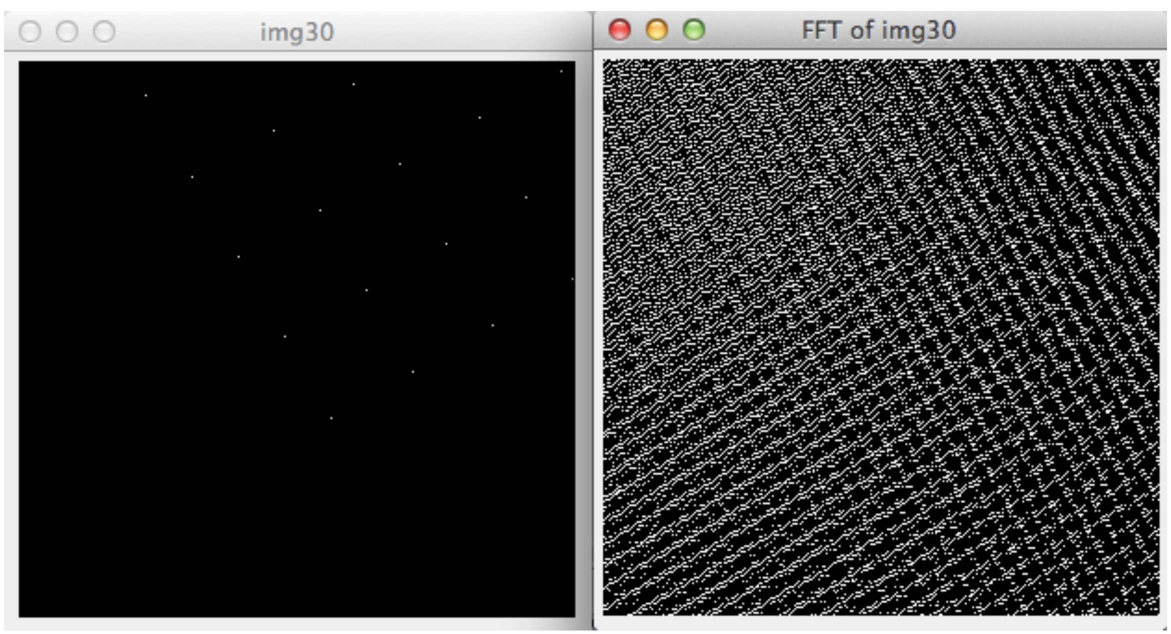
so changes are still needed

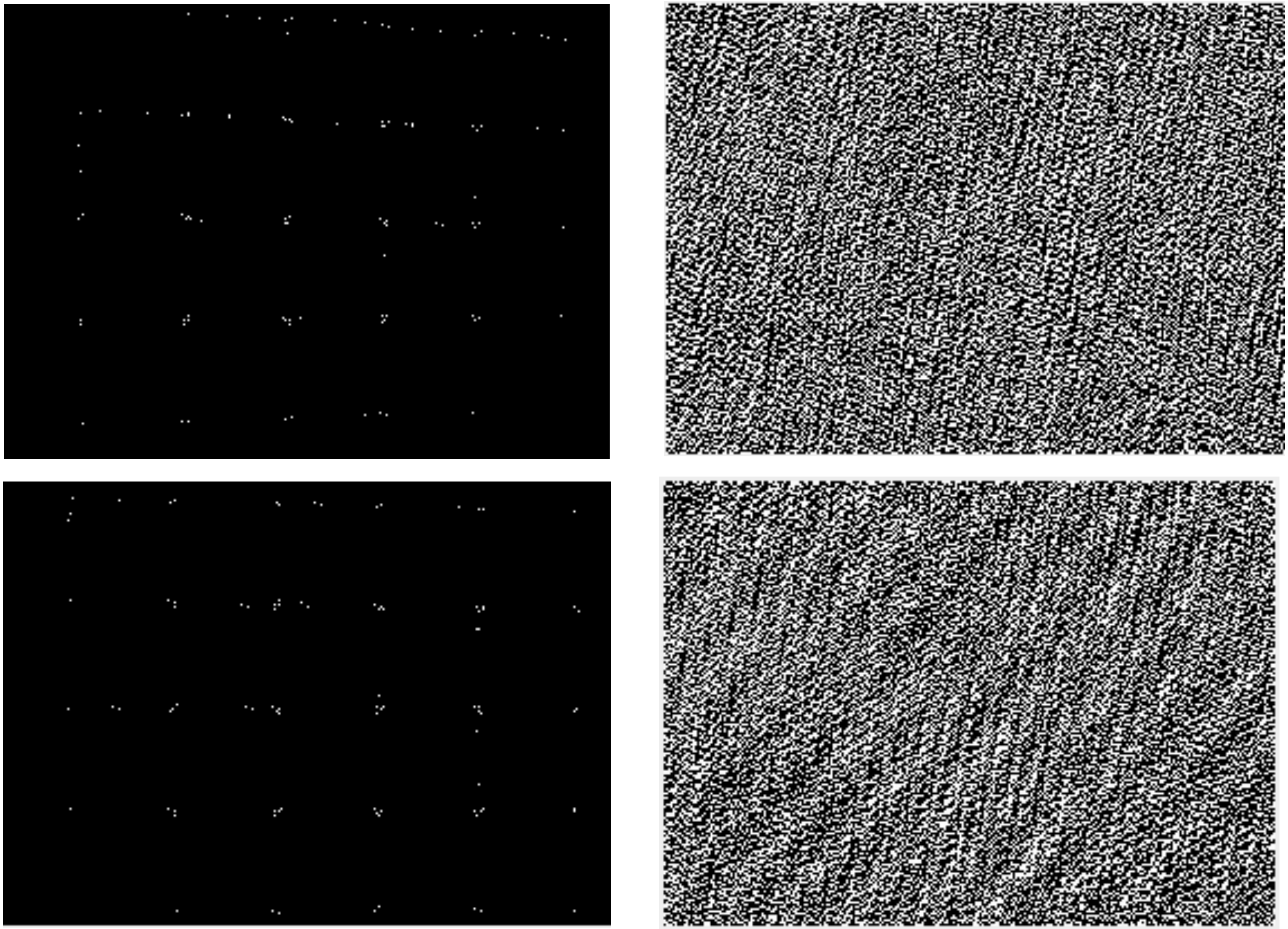
quick look at complexity of using FFT to find textures. needs images to be aligned along axes of interest for simplest extraction of spatial components.



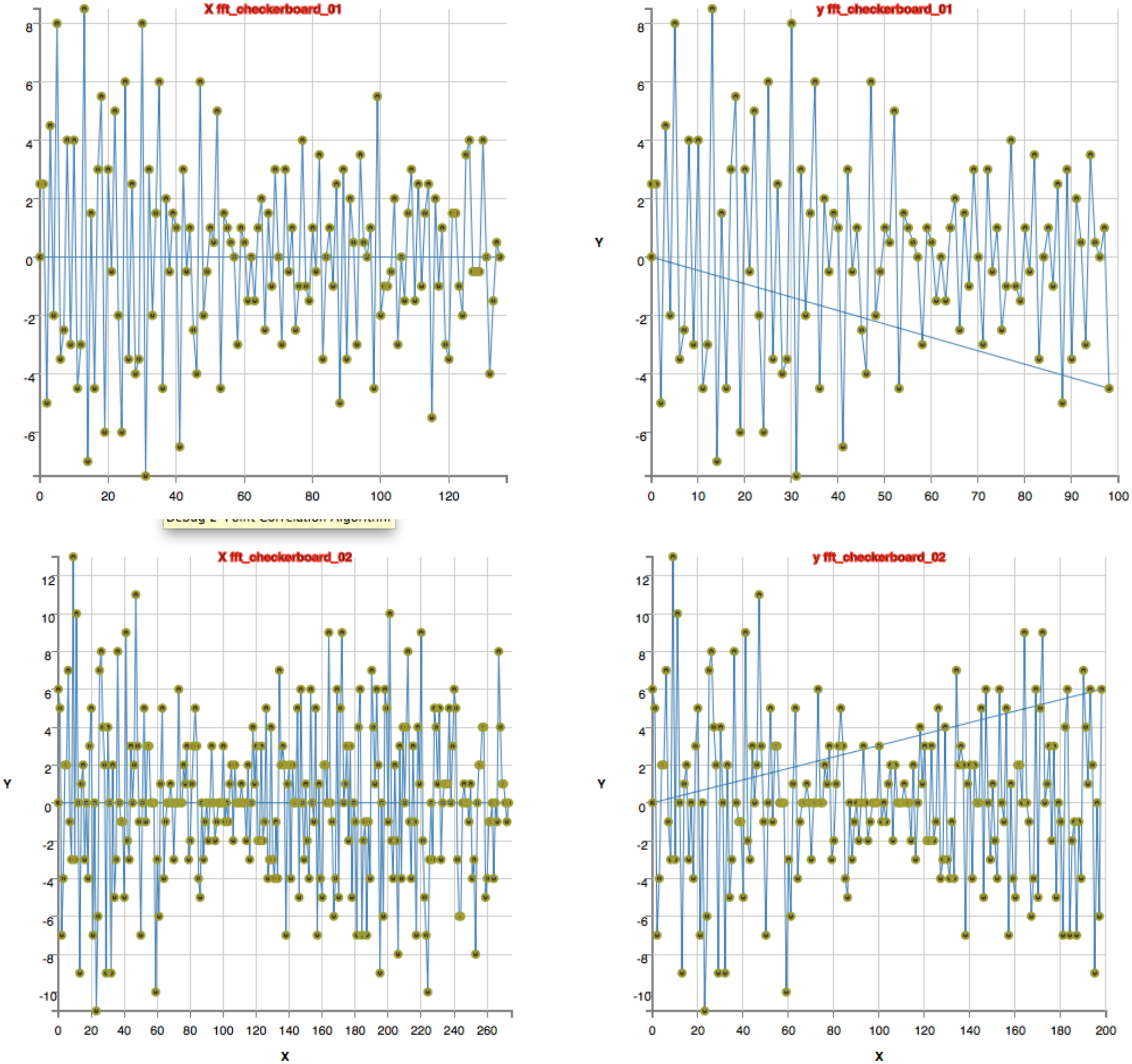
2D FFT of a grid of points



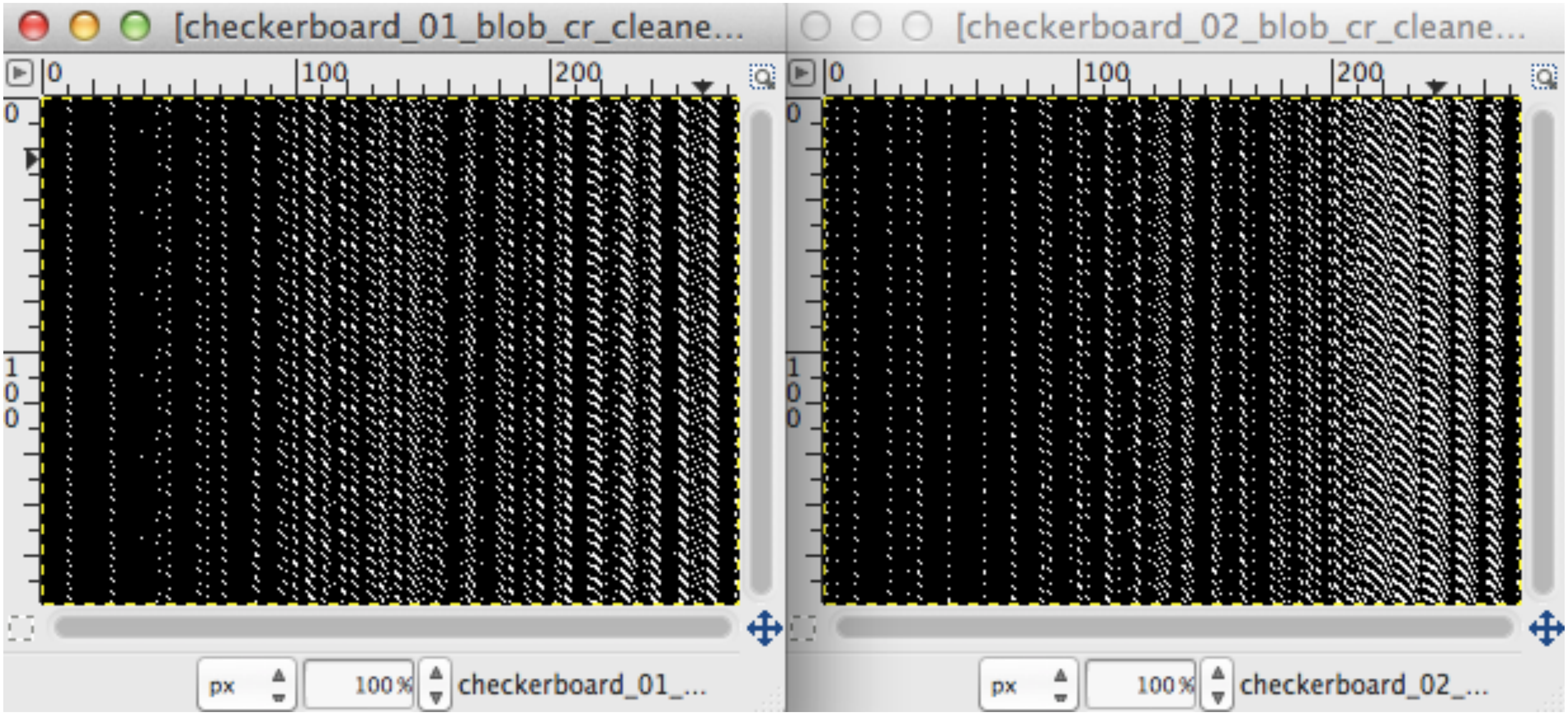
FFT of a same grid of points rotated by 30 degrees



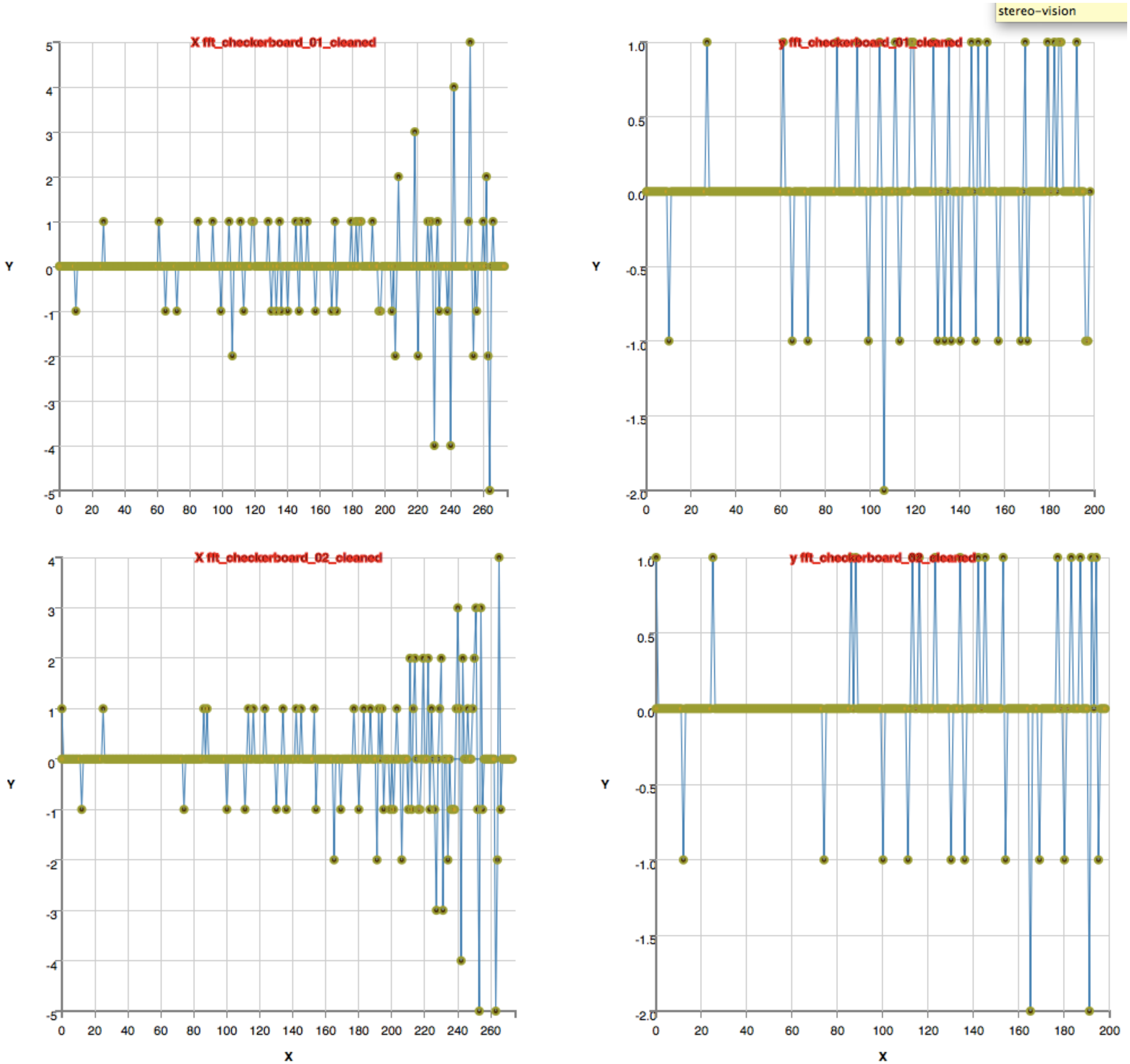
FFT of the corner regions



FFT
of the
corner
regions

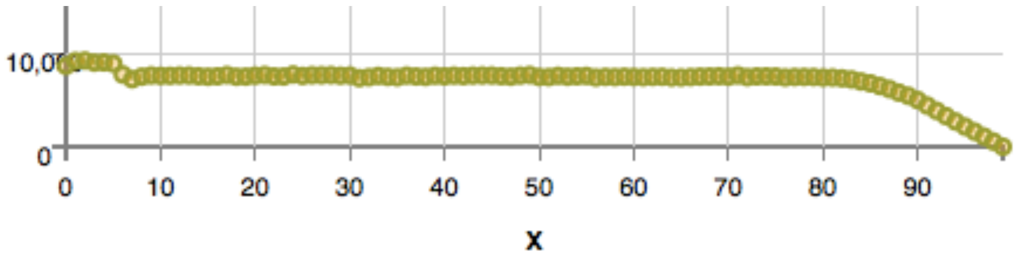


FFT of corner regions (artifact corners removed)

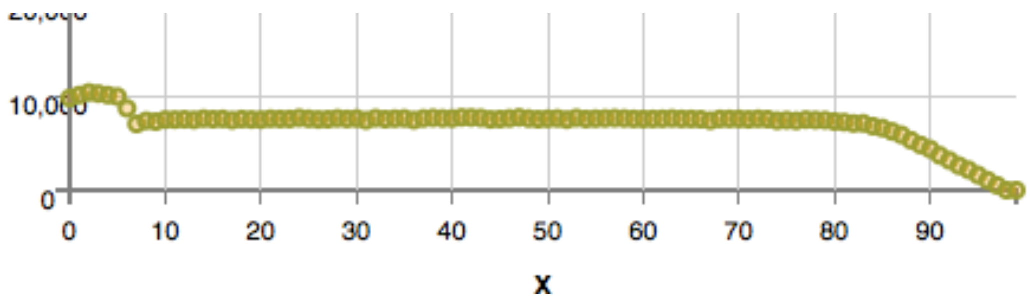


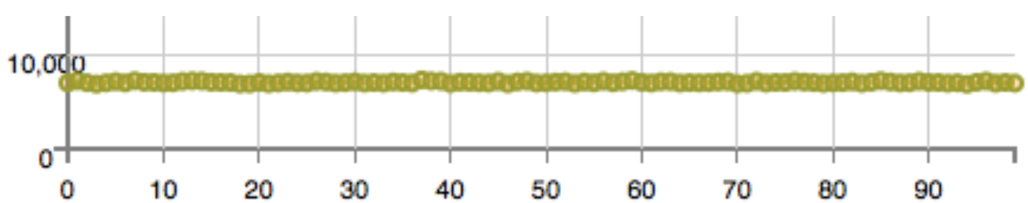
FFT of corner regions (artifacts removed)

On a different topic, looking at improving the random sampling for the RANSACSolver.
Created a k bits random number selector for a distribution of indexes and
created the same for a distribution of indexes altered to replicate the number by its value.
Plotted is a look at the numbers drawn from 100,000 throws using these methods.



The new methods have a few biases due to a bug in BigInteger's random. (filed a bug report accepted by oracle)





randomly choosing k integers from SecureRandom, in contrast to the above, shows a very uniform distribution

On a different topic, looking at peaks and silhouette matching

For the outdoor images, can find the sky and create a sky mask and also create corners from just the skyline. (see skyline_extraction.pdf)

This sky mask can help to pre-process the image before feature finding and correspondence. Note that the skyline corners will be improved before real use, this is a quick look.

Might adapt the contour matcher to work with open curves, especially for mtn silhouettes such as half dome test images.