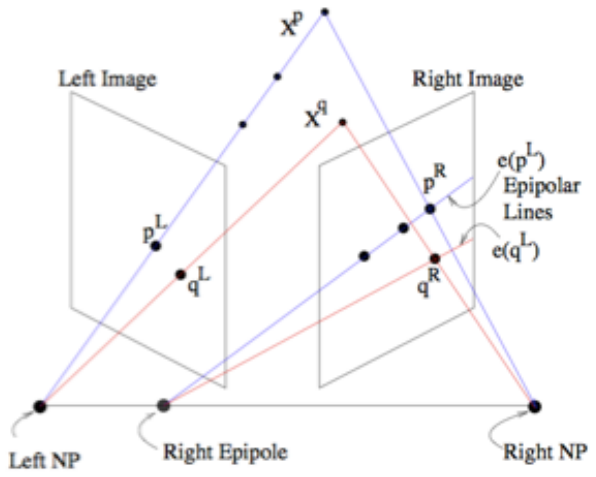
Projection, stereo and panoramic images

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.

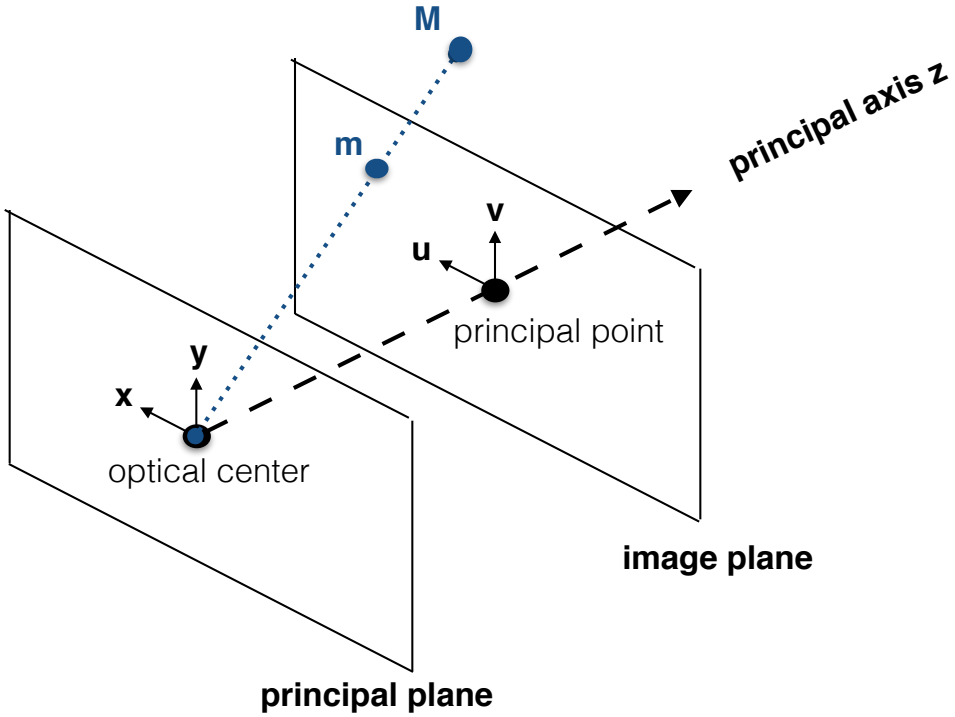
The projection of X^p to left nadir, NP is seen as an epipole line in the right image. If more than one epipole line is present in the right image, they converge at the right epipole (which might not be within the boundaries of the image).

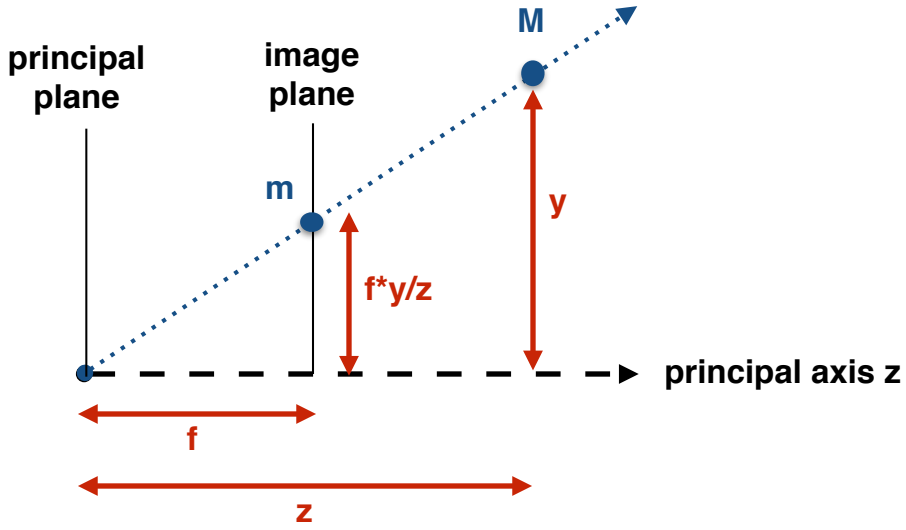
Once the points in the left image, X_L are matched with points in the right image, X_R, if there are at least 7 points, one can determine the "bifocal tensor",
a.k.a. "fundamental matrix, relating the points in the 2 images using a 3x3 matrix of rank 2.

$(X_L)^T * F * X_R = 0$ 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***m** = P***M**
where M = $(x,y,z,1)^T$ and m = $(f*x/z, f*y/z, 1)^T$

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 | **R t** |
The six parameters are called *external parameters*. | 0 1 |
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 **λ** applied is P = **λ*****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 **λ**=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 $2n \times 12$ called the coefficient matrix A

From a set of n point correspondences, we obtain a $2n \times 12$ 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^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^T (its the one dimensional reshaping of the homograph matrix).

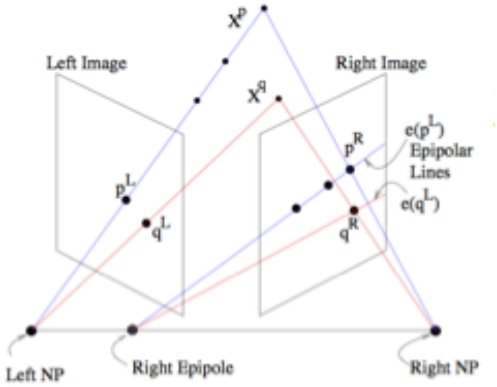
multi-view geometry (cont.)

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.

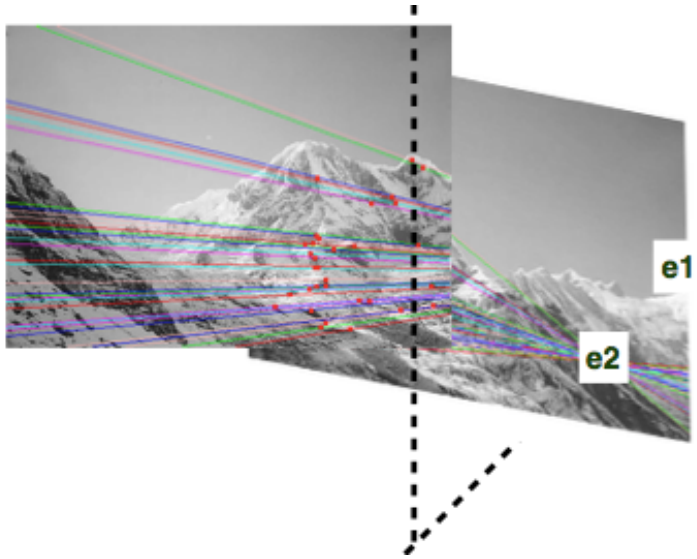
In Trifocal geometry, a trifocal tensor is used.

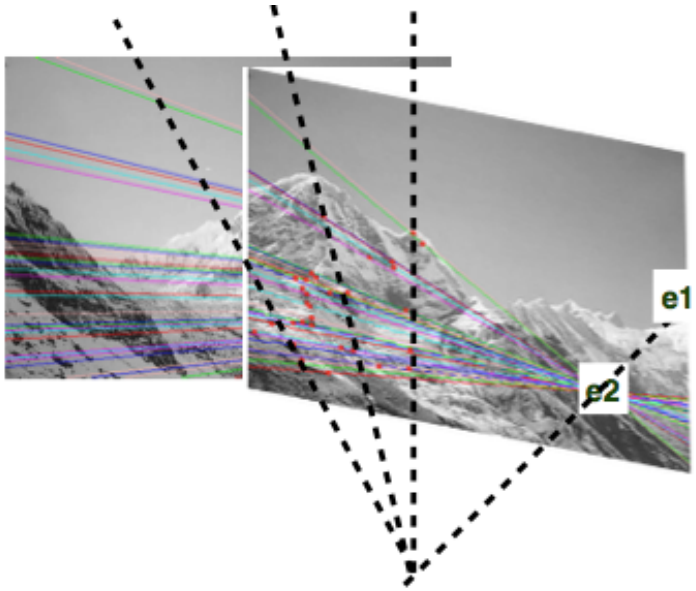
Projection, stereo and panoramic images



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



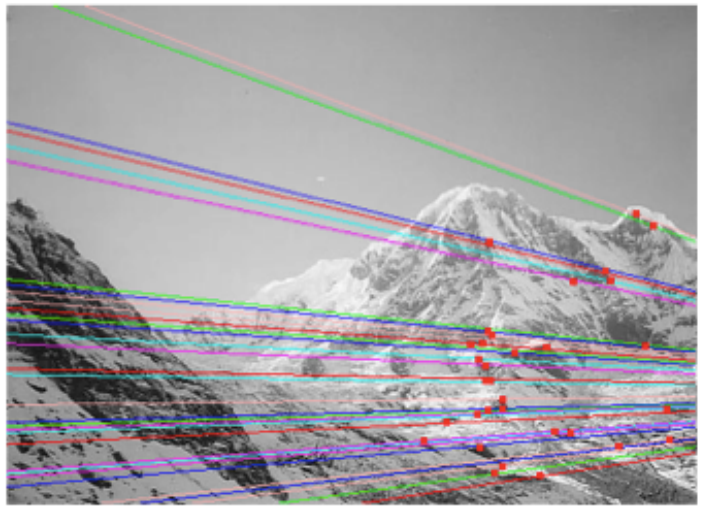


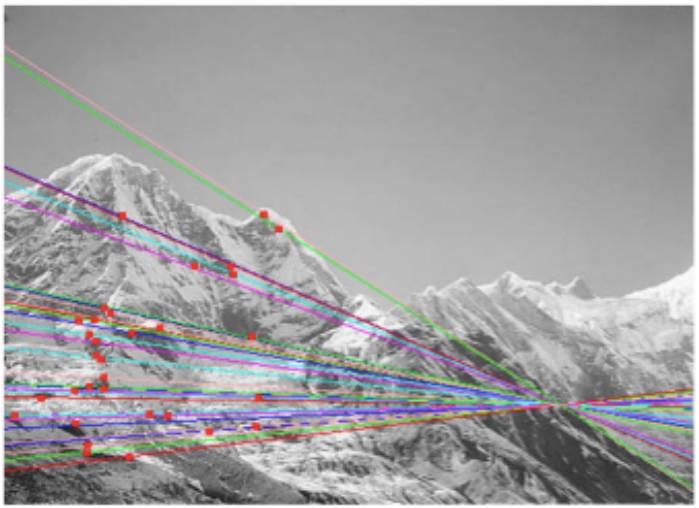
same camera objectives (nadirs), but different orientation for the 2 images (that is, rotated around the same nadir)

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":





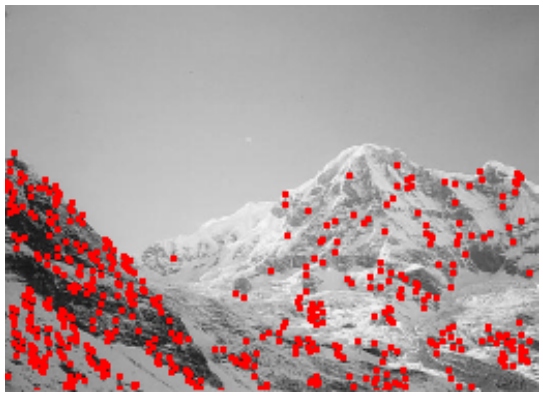
epipolar fit to 32 points already known to match shows what the epipolar projections should be when the corner find + corner match + stereo projection solve are correctly automated.

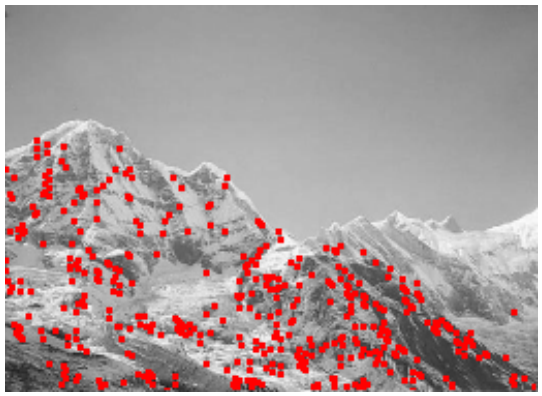
```
nMatched=32
avgDist=0.281
stDev=0.508
```

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.

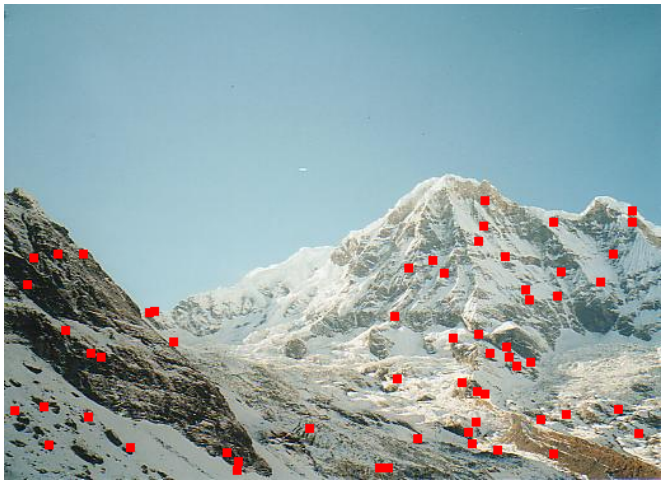
"feature descriptors" were needed. greyscale image intensity descriptors are usually enough for matching, and gradient images help to determine the "orientation angle" of a coordinate. the "oriented features" of one image are compared to those of another image to find the best matches (a match being 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 was made using the harmonic mean of the auto-correlation matrix of each point.
— 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.

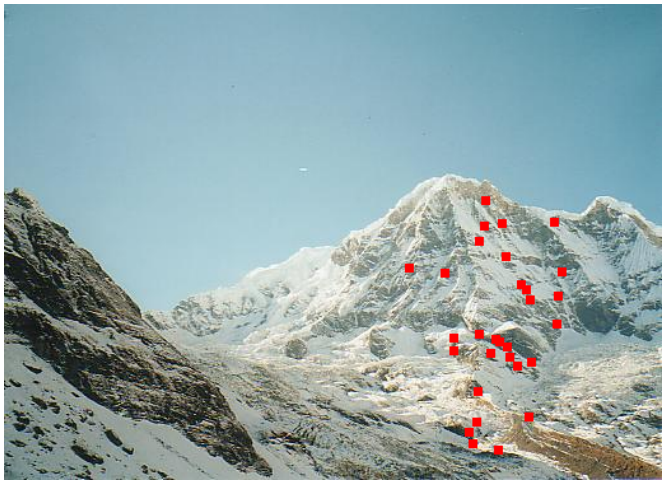




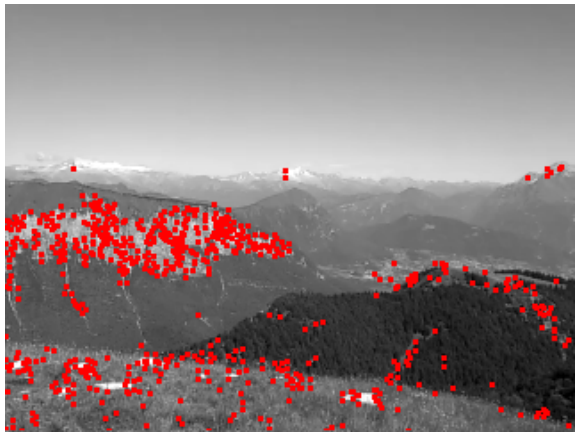
2nd deriv points

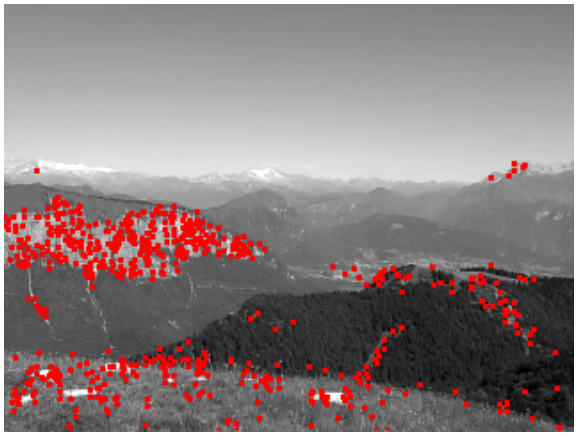


points filtered for those w/o close 2nd best match, then filtered for localizability.

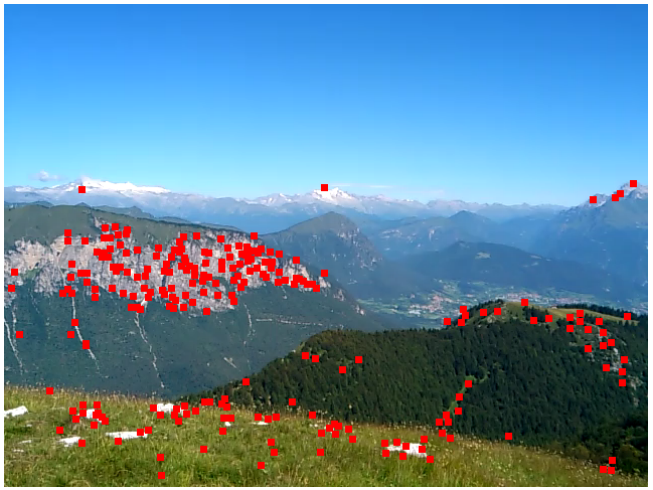


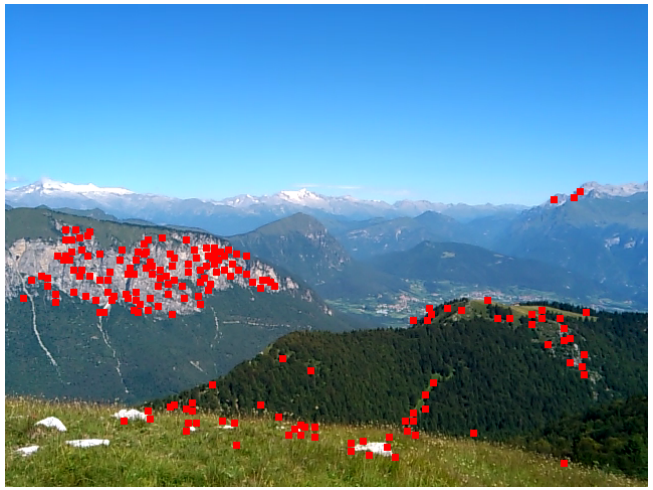
matched by RANSAC algorithm



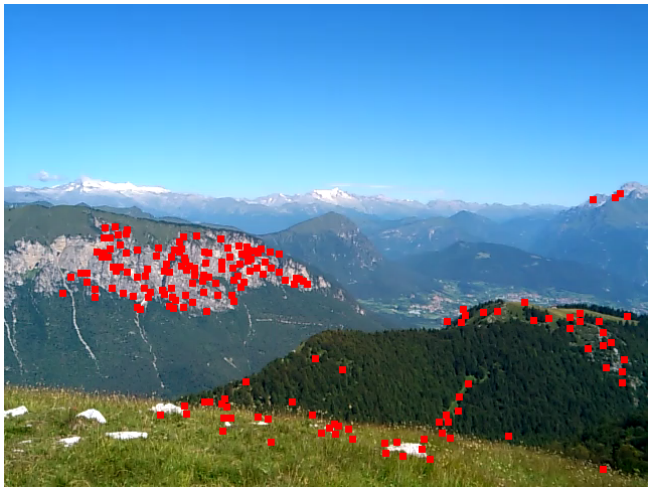


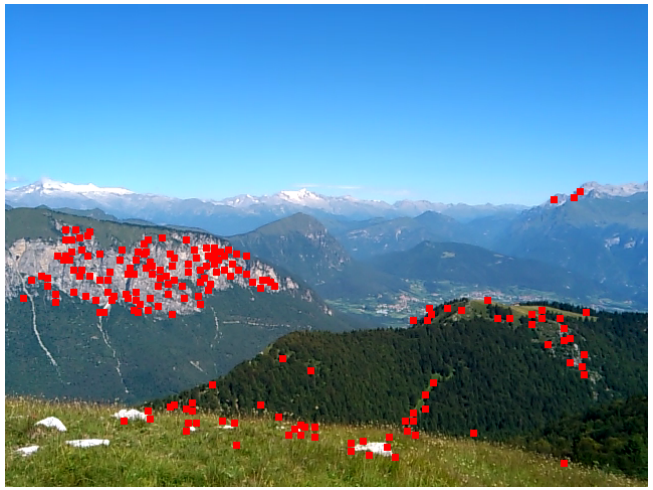
2nd deriv points



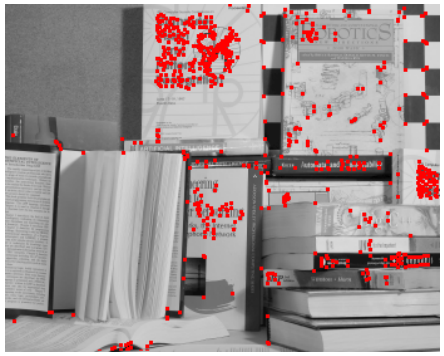


points filtered for those w/o close 2nd best match, then filtered for localizability.



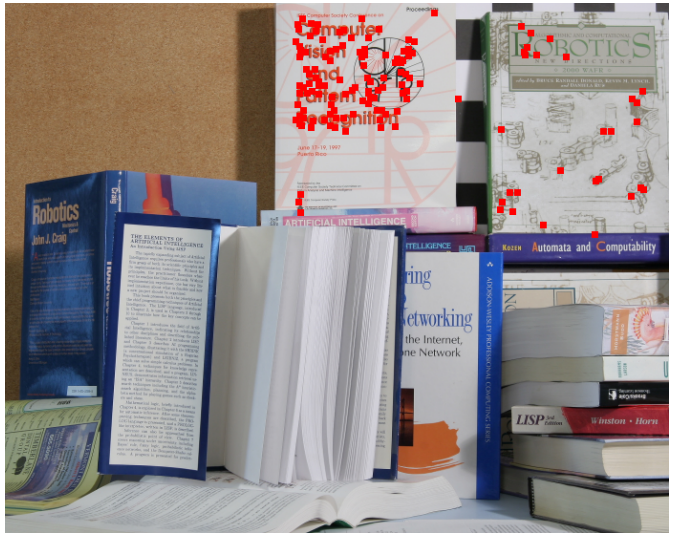


matched by RANSAC algorithm



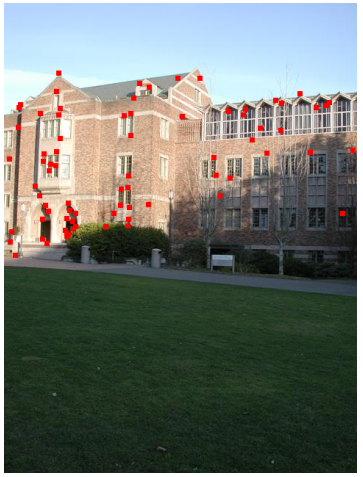
2nd deriv points

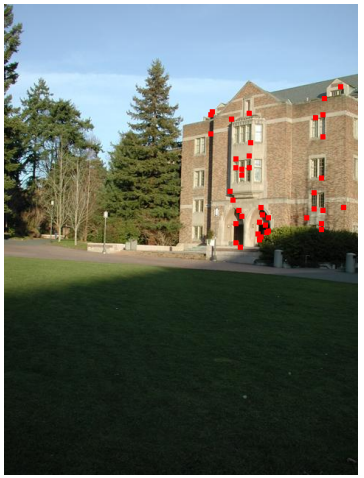
points filtered for those w/o close 2nd best match, then filtered for localizability.



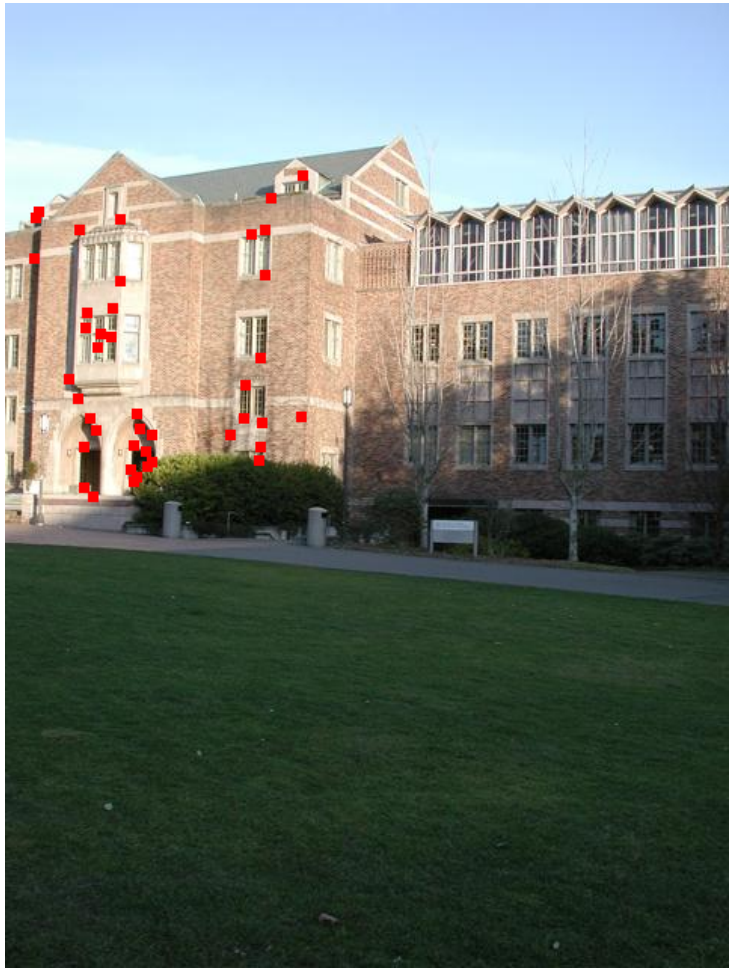
matched by RANSAC algorithm

2nd deriv
points

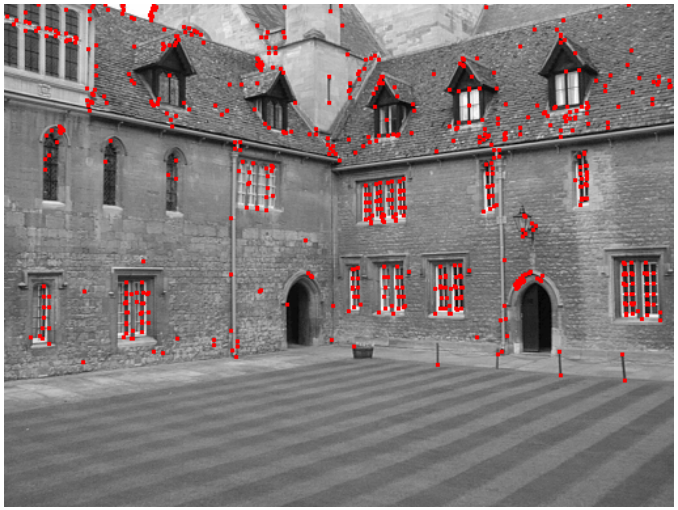




points filtered for
those w/o
close 2nd best
match,
then filtered
for localizability.

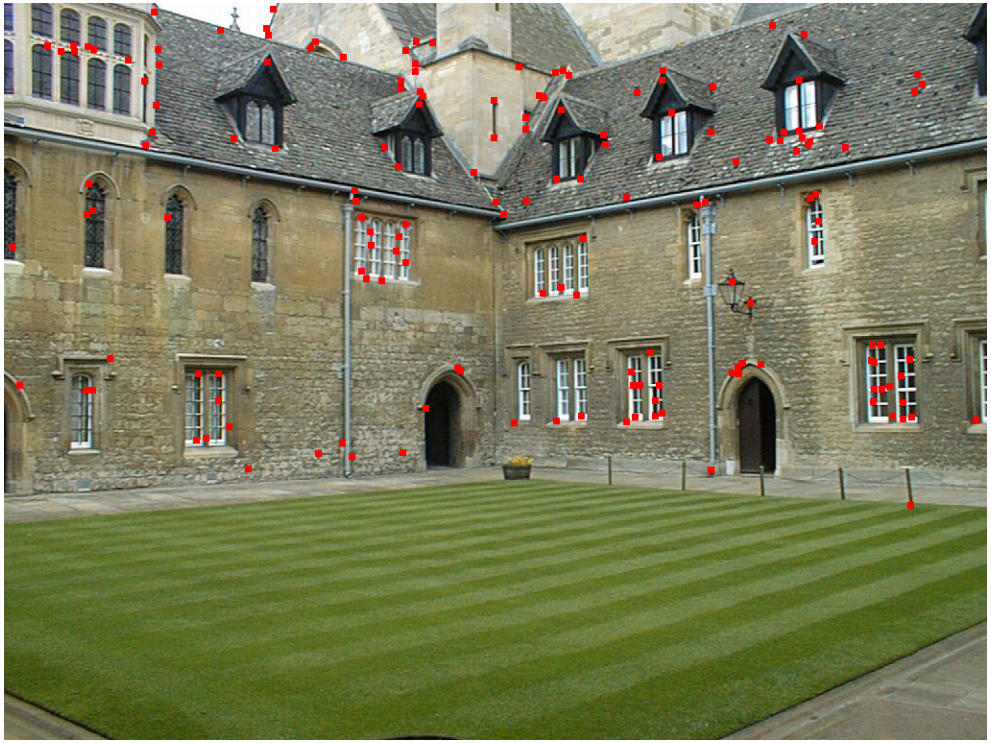


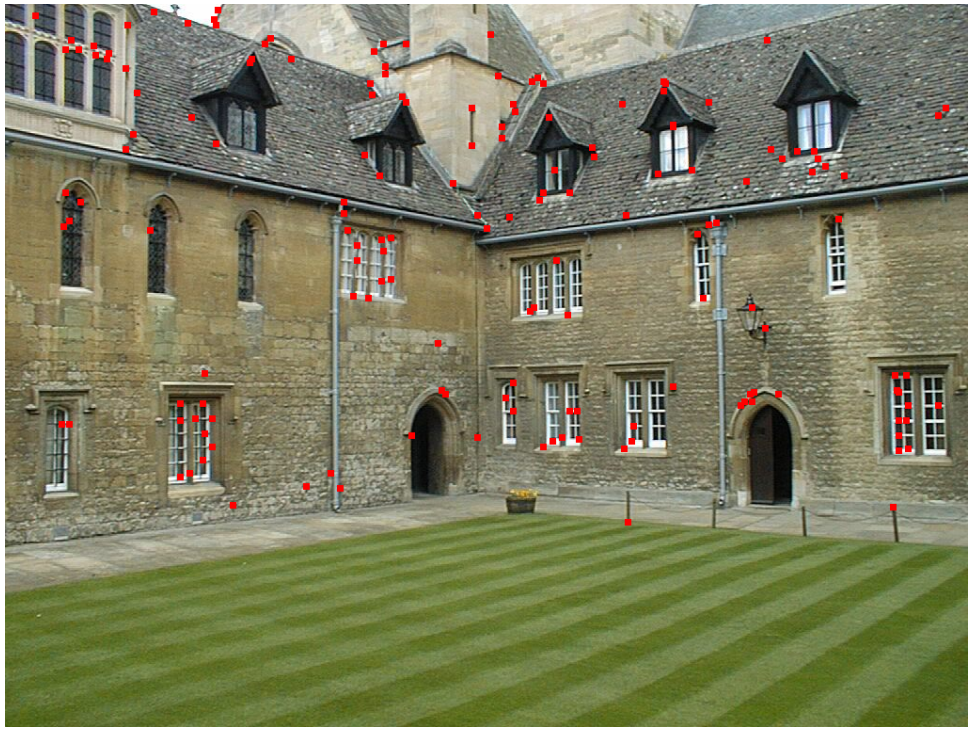
matched by
RANSAC
algorithm



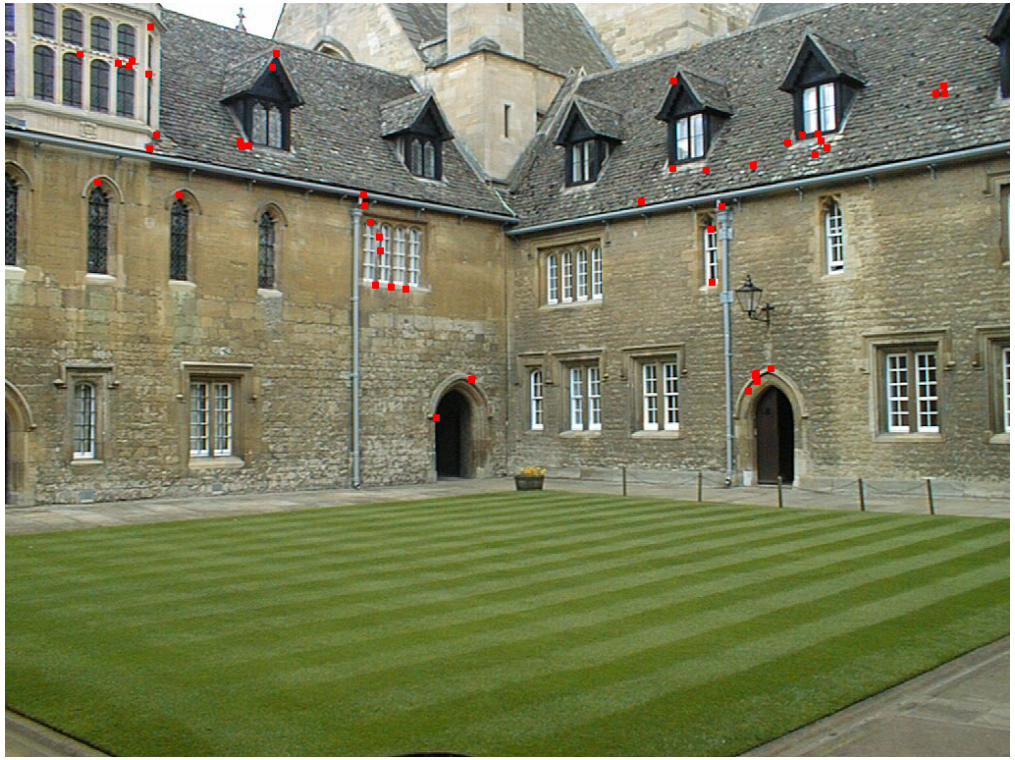
2nd deriv points

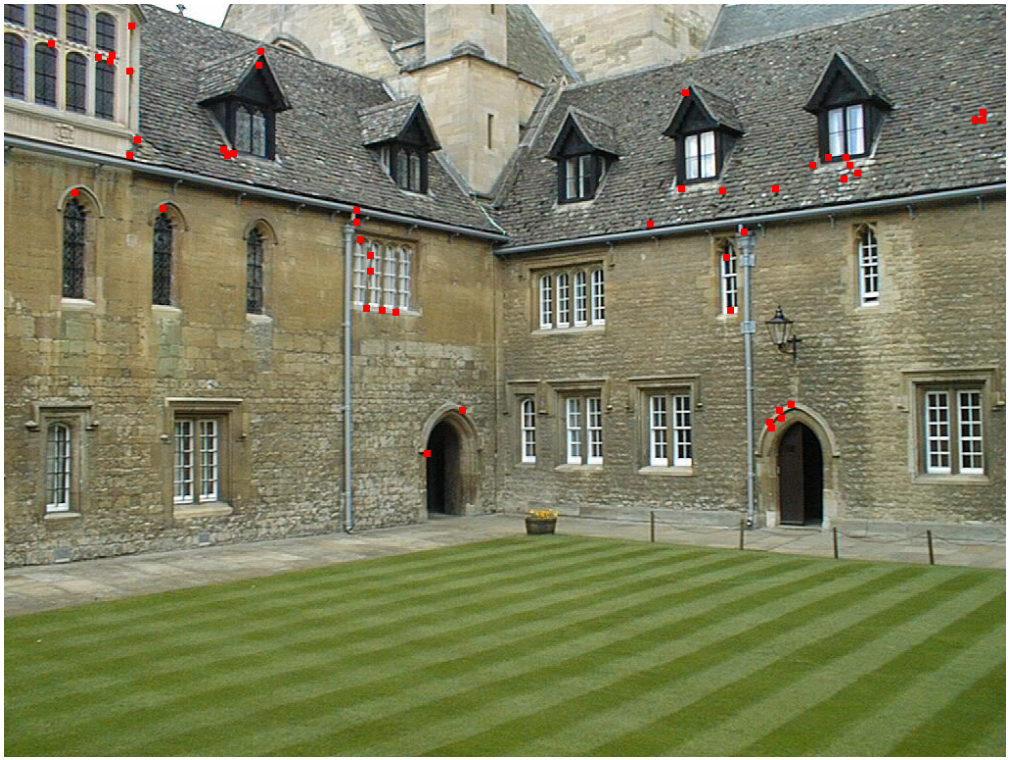
points filtered for those w/o close 2nd best match, then filtered for localizability.



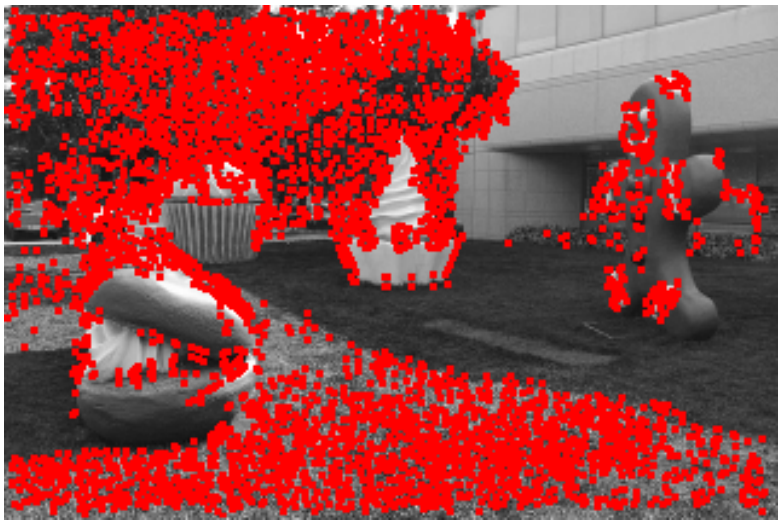


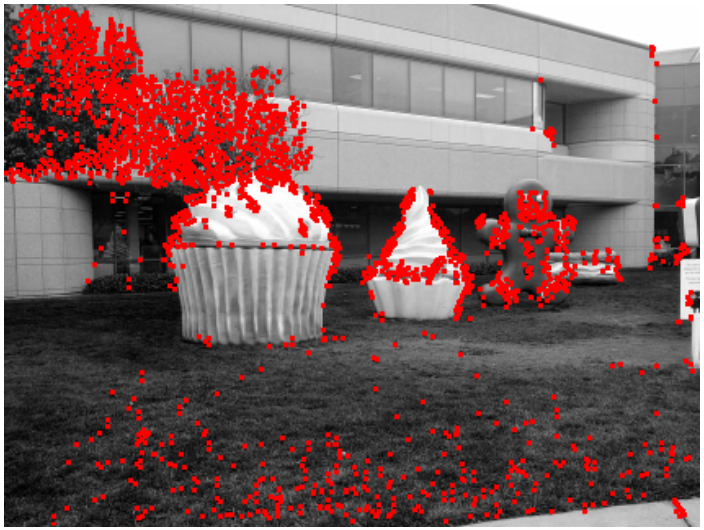
matched by RANSAC algorithm





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.





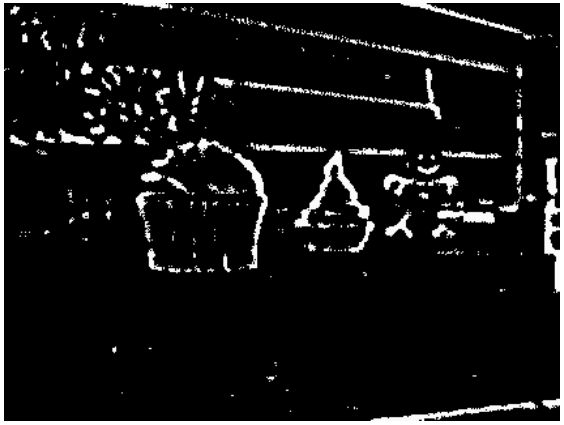
extracting about 5000 2nd derivative points in both images

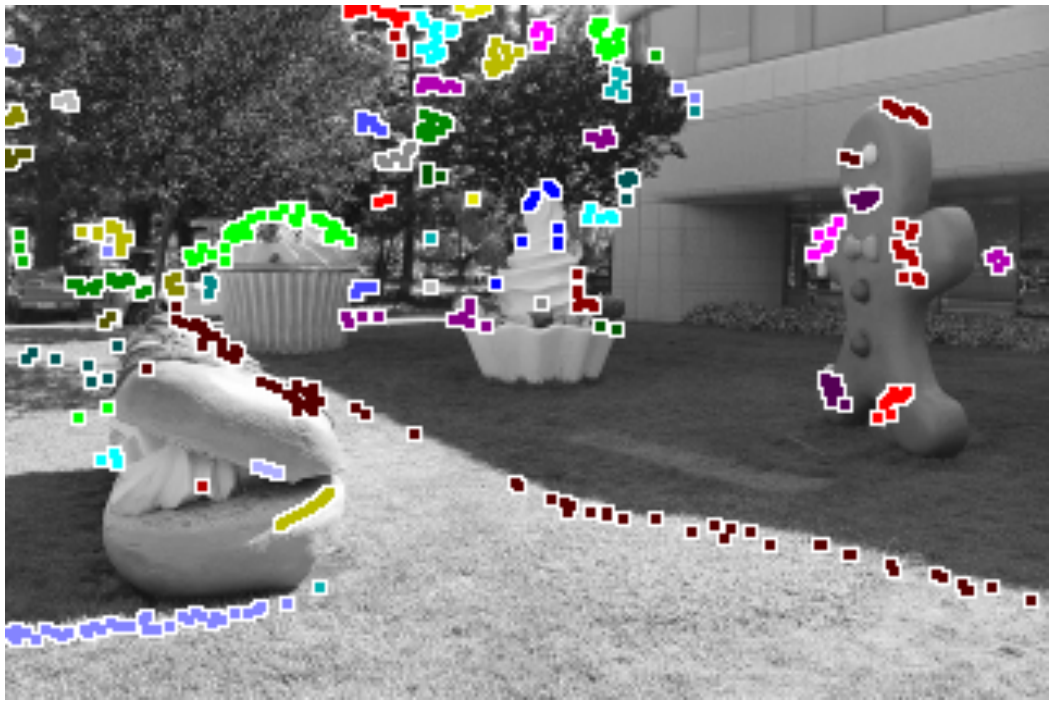
Next, the points need to be matched and color seems to be necessary.

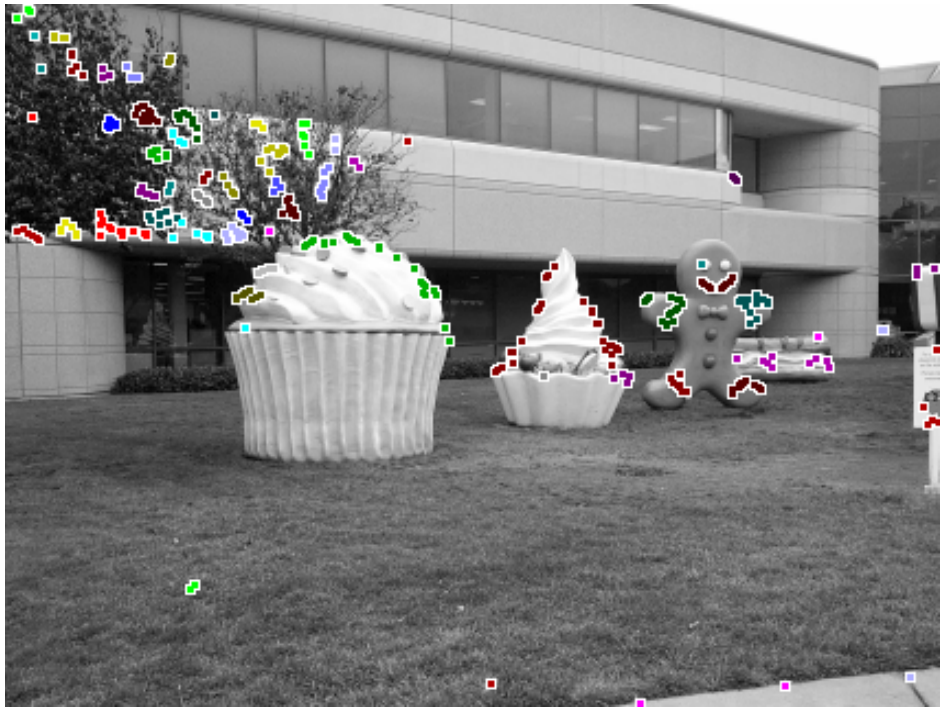
Because the objects have moved over time, the best feature matches followed by ransac w/ an epipolar solver are not going to provide the best solutions.
The two largest differences between the images are the sidewalk and grass, followed by the eclair.

So, matching points at an object level is required and the ransac with epipolar solver should be fine for that, but may need to use features along with distances in the ransac epipolar model evaluation.

Note: tried the use of a segmentation filter based upon the atrous wavelet to limit the points to be matched, but the *results were not the best* because the points also need to be matched as groups of points within objects larger than the atrous segmentation boundaries.







Added descriptors for CIE LAB color space and a delta E 1994 calculation for the difference in the CIE LAB colors. These descriptors were useful for general matches between tree leaves, but missed the gingerbread man. gingerbread man is then extracted from each image, isolated.

filtered corners

Filtered corners show that there should be many points that could be matched (same positions in both image).

Differences in features are present due to illumination differences (shine and non-grey changes) and due to adjacent backgrounds from different perspectives.

changing the close 2nd best threshold to a higher 0.9, helps keep a tie match, but the remaining are mostly false matches.

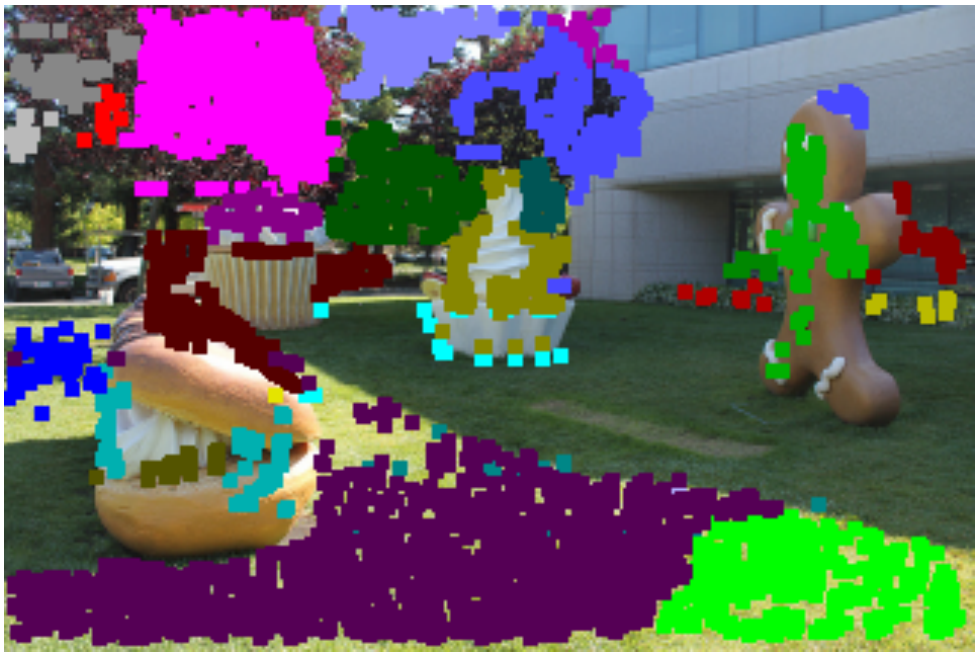
some consideration for homology or groups of points larger than 1 looks necessary.

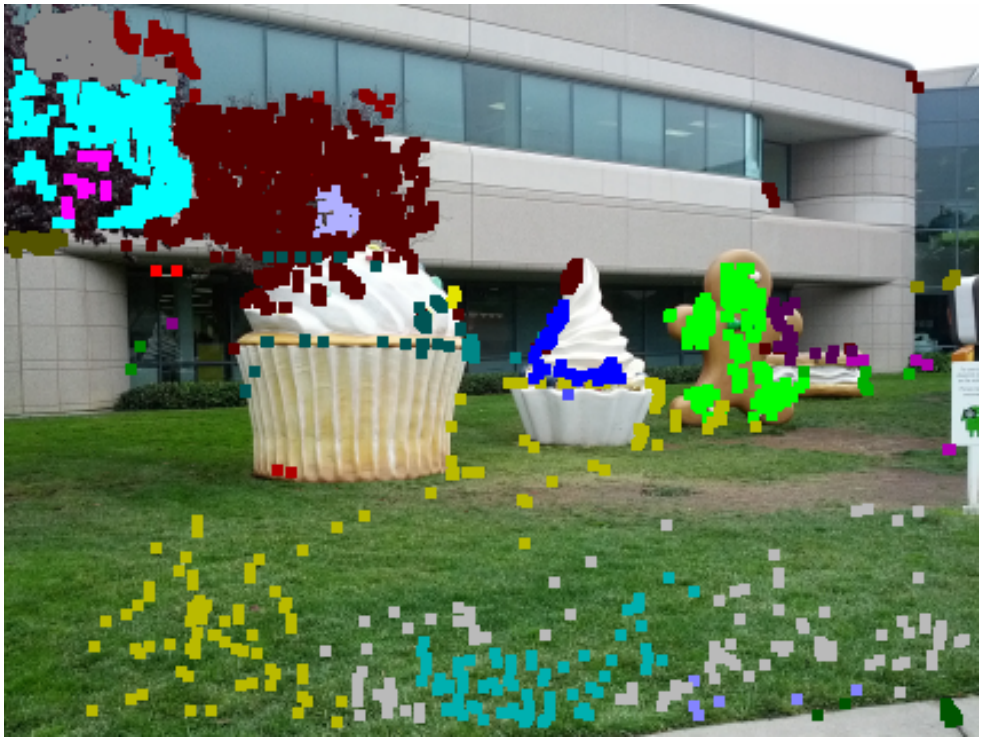
```
bin image down to 75 x 75 (or some other size); then ImageSegmentation.applyUsingCIEXYPolarThetaThenHistEq(img, 16, false)
```




this binning followed by ciexy polar theta segmentation looks like it might be a good way to find boundaries containing larger groups of points to use in matching.

next, would use a dfs pattern to find contiguous regions of same color, then the perimeters of those regions, then scale that to the reference frame in progress, then merge groups that are within bounds of the perimeter polygons. then use group based matching.



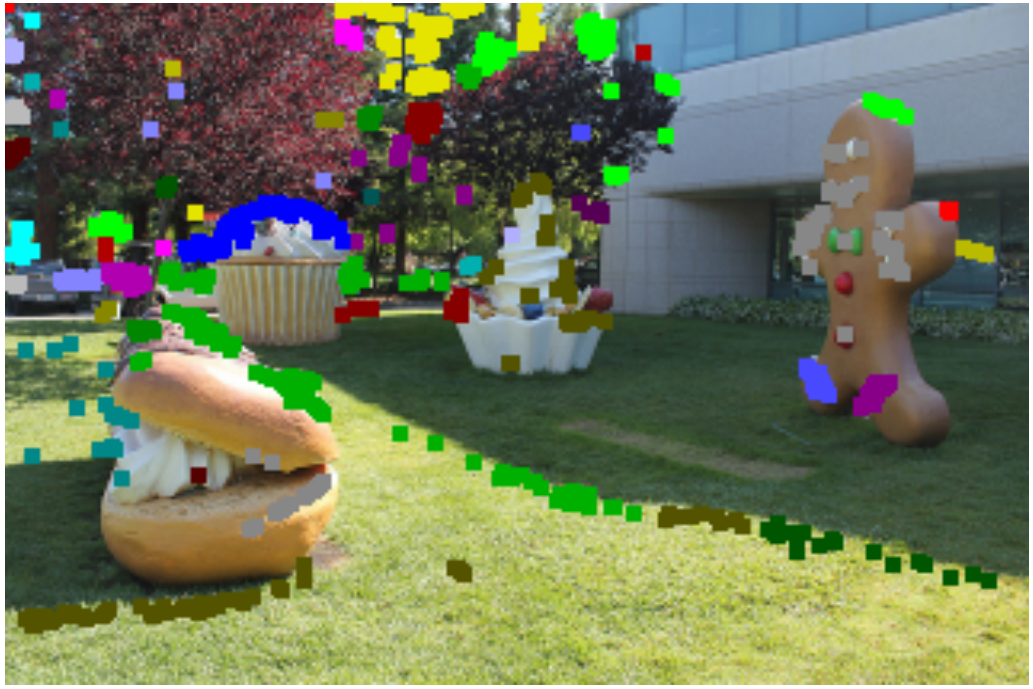


Using the larger bounds just extracted from lower resolution segmentation, placed the points into groups.

Because some of the objects in the image have moved, need to compare the points on a group by group basis (ideally, the group would be the complete object, like the gingerbread man).

Using feature matching to make a preliminary list, then ransac with an epipolar model and an evaluator that uses features and distance for cost calculations to make a preliminary inliers list, then re-matching the unmatched by features and using those with the inliers as input for ransac again.
(see next pages).

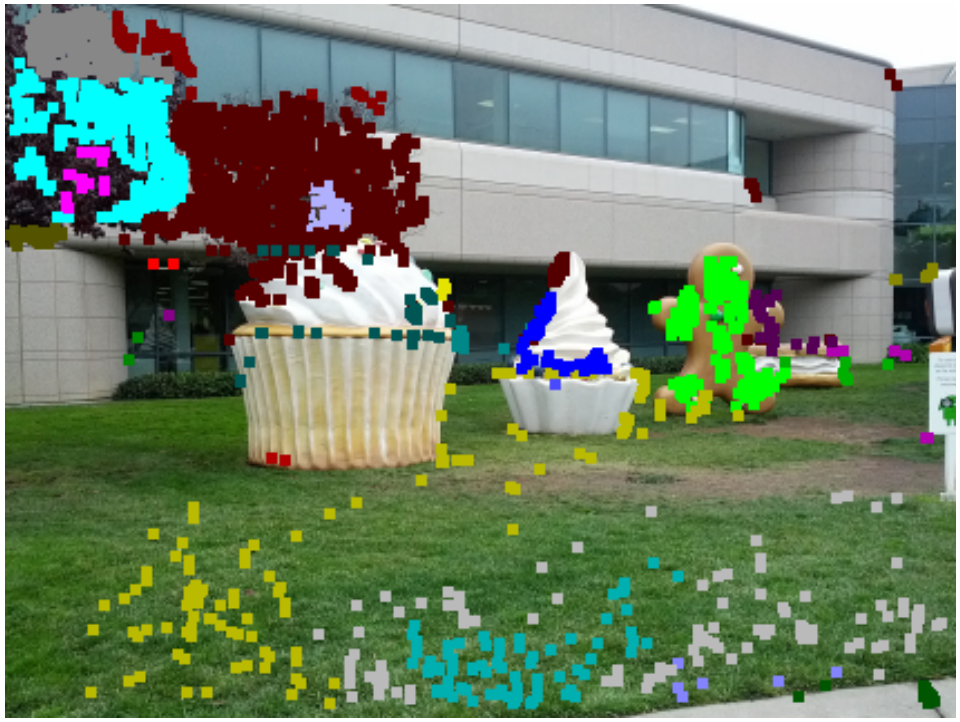
NOTE: consider identifying highly textured and filtering that with atrous



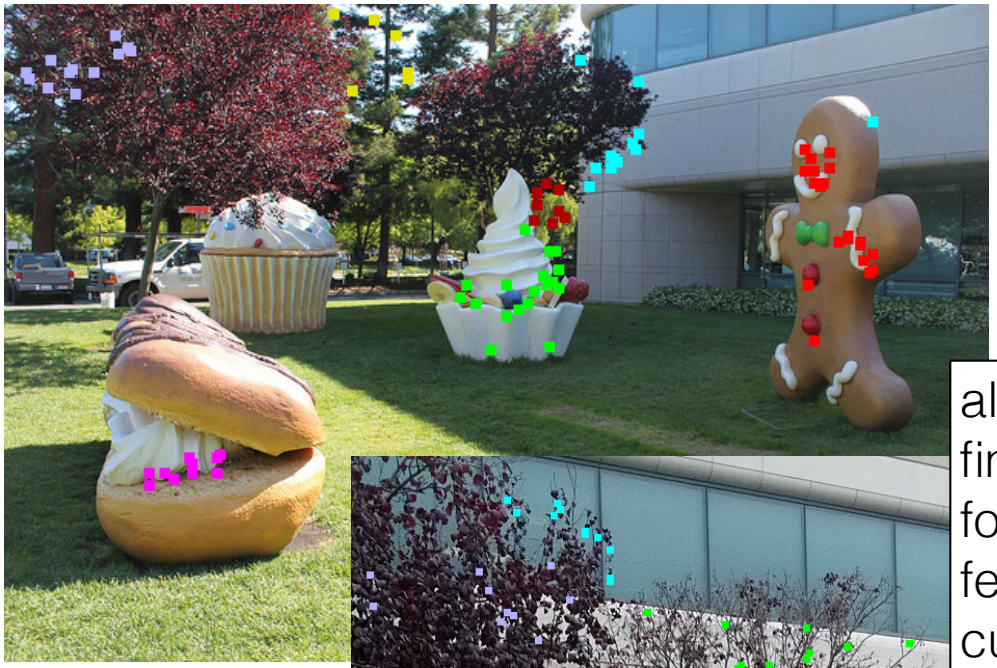
NOTE: these were the points after having ***filtered*** to keep only those matching ***atrous wavelet based*** segmentation *and* then grouping them further into the ***larger bounds*** found by lower resolution segmentation.

The results were not the best. Only a few true matches were found for the gingerbread man, even when matches were group by group.

But the reduction of the highly textured areas is still very useful, so taking the groups from the previous page and if the density is very high and the area large, will filter those groups individually by the atrous based segmentation to reduce those points by number.

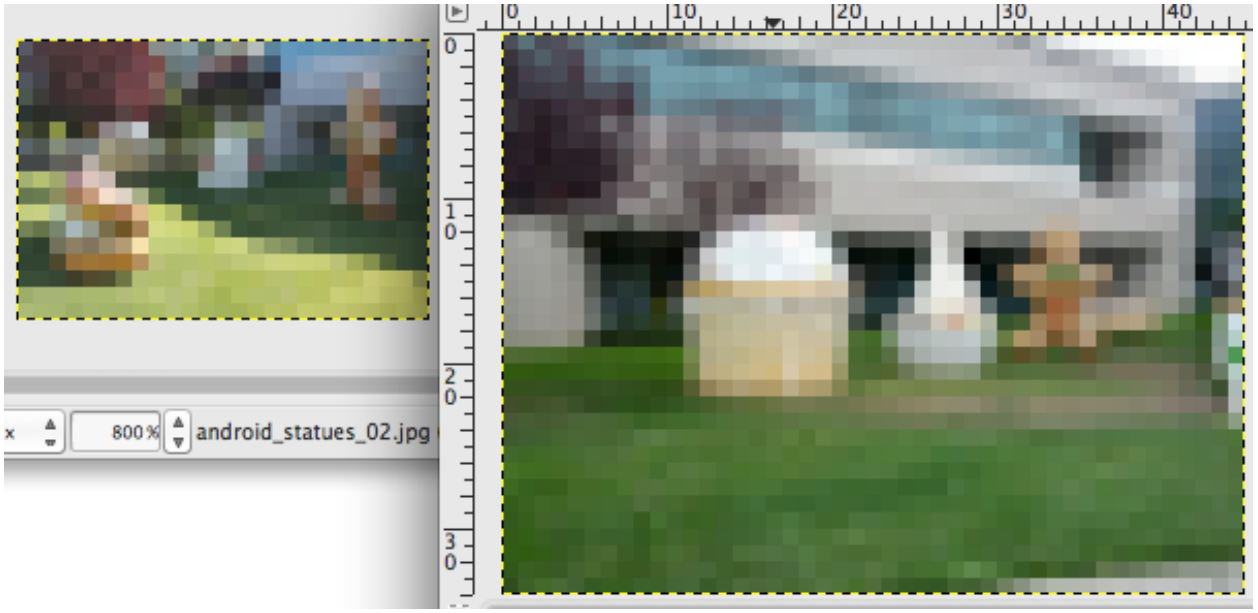


Highly textured areas were filtered by the atrous based segmentation. This may need to be tuned for each dataset.



feature matching by groups of points and an epipolar solver w/ ransac and color descriptors, now finds the gingerbread man.

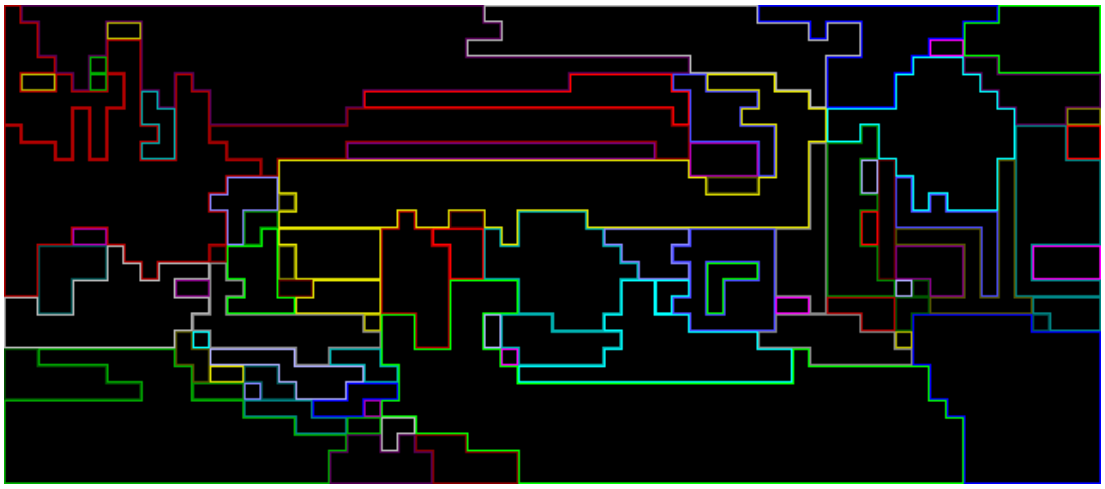
all could be improved by segmentation that finds the object contours better. the ice cream for instance could be matched by shape then features. shape can be matched using the curvature scale space contour matcher.

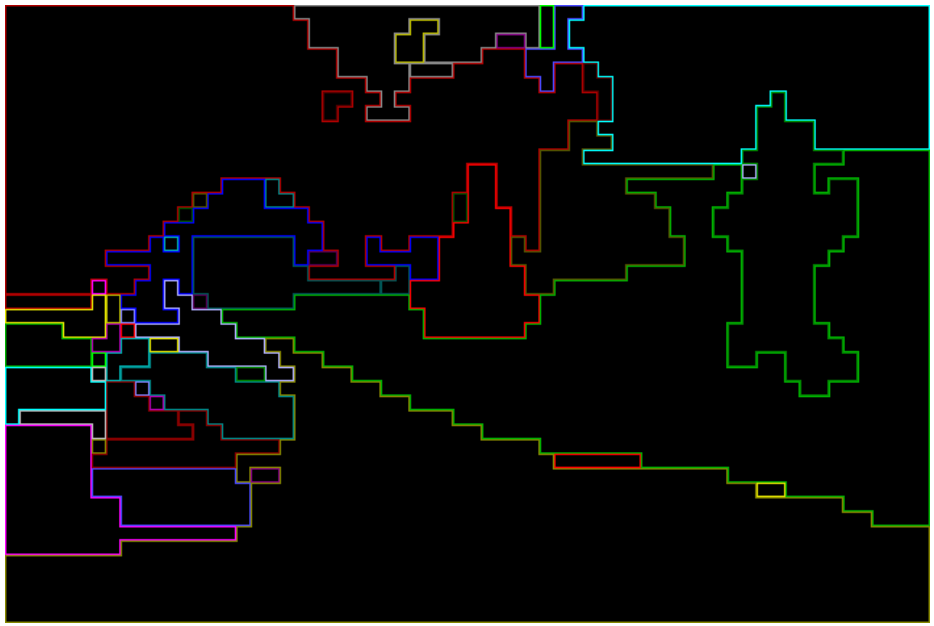


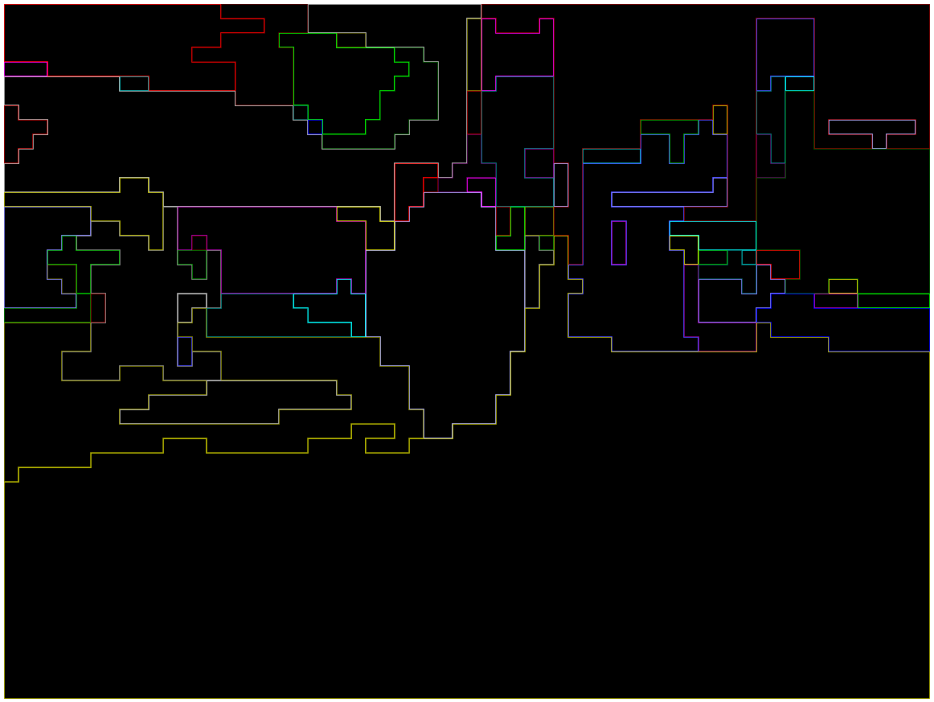
Since a pyramid approach is necessary, can see that the pattern of pixels for color are unique for the gingerbread at matching resolutions of different poses,
so one might prefer object recognition methods instead of sparse point (feature) matching when the images have some objects in common, but the camera perspectives are not panoramic or stereo and the cameras are different.

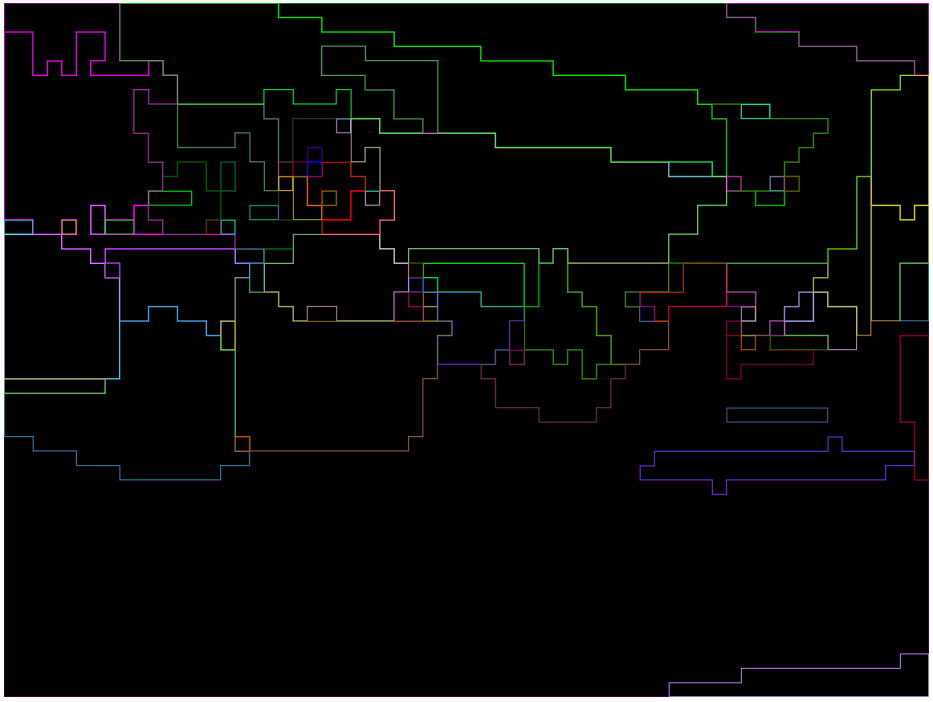
A summary:
goal is to be able to match objects in one image with same objects in another image where details such as location or lighting have changed.
(The images are not necessarily stereo or panoramic.)
Using the android status as tests, can see that color is necessary.

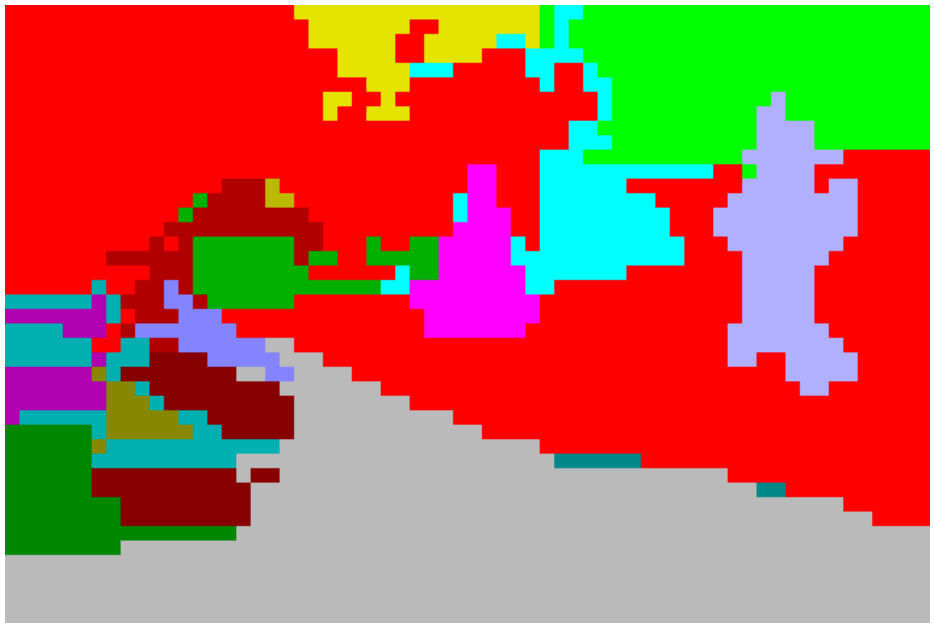
(1) To find the rough location of objects in the images:
decimated image to sizes near 64 pixels per dimension
and then 32 pixels per dimension.
Then used a super-[ixel algorithm to make starter cells for the normalized cuts algorithm which then labels the small images.
Then the 2 image segmentations are combined to bring out details better
and then the combined relabeling is used
to find contiguous groups and calculate their CIE LAB colors
before expanding the labels into full reference frame coordinates.

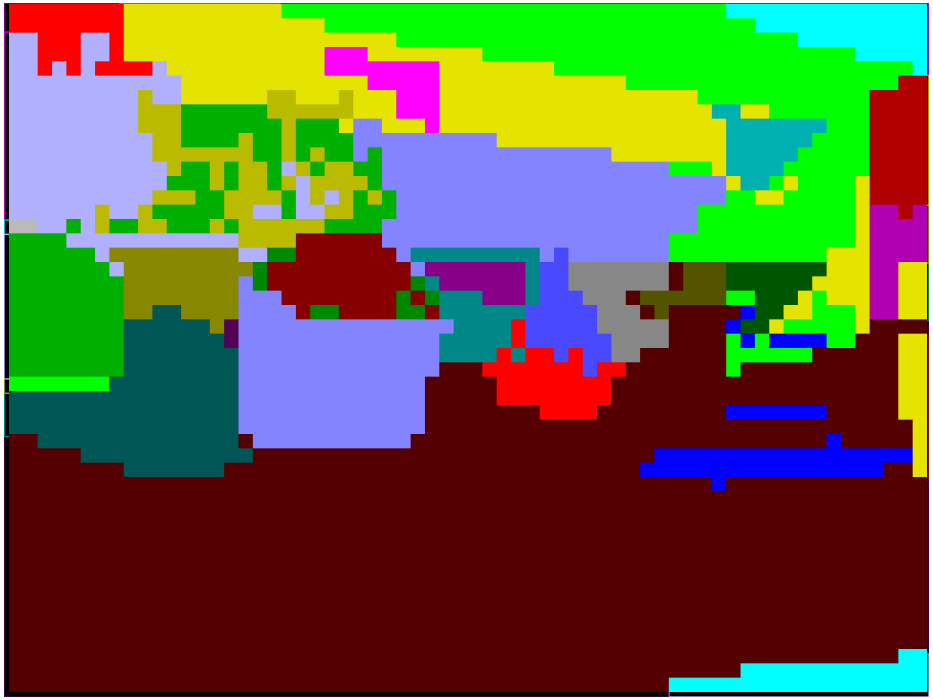




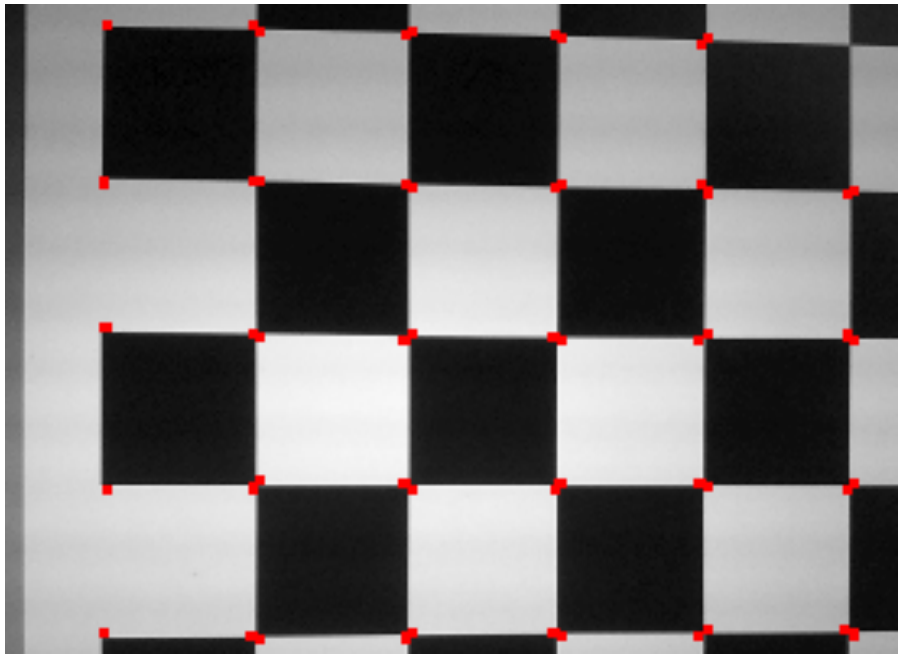


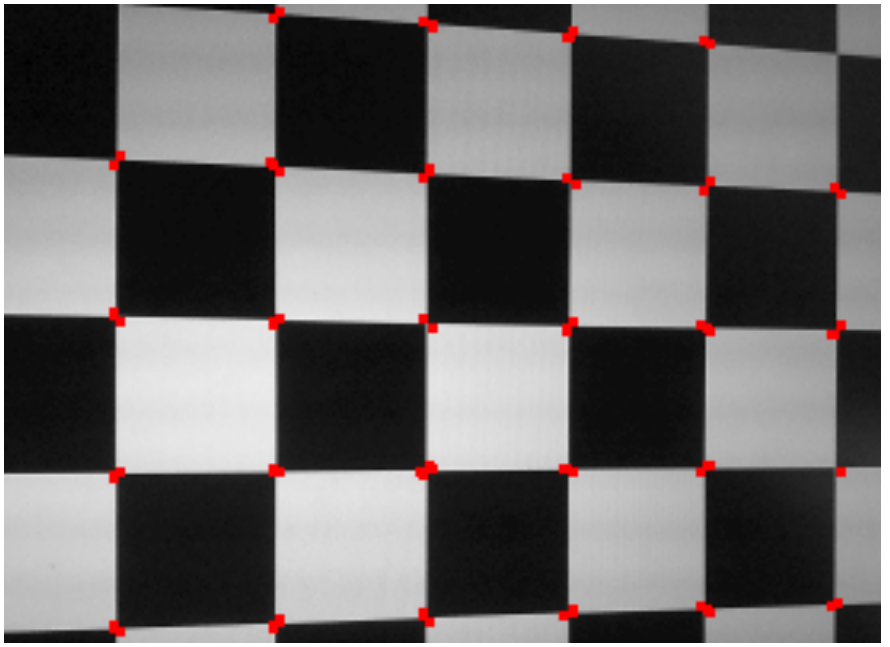




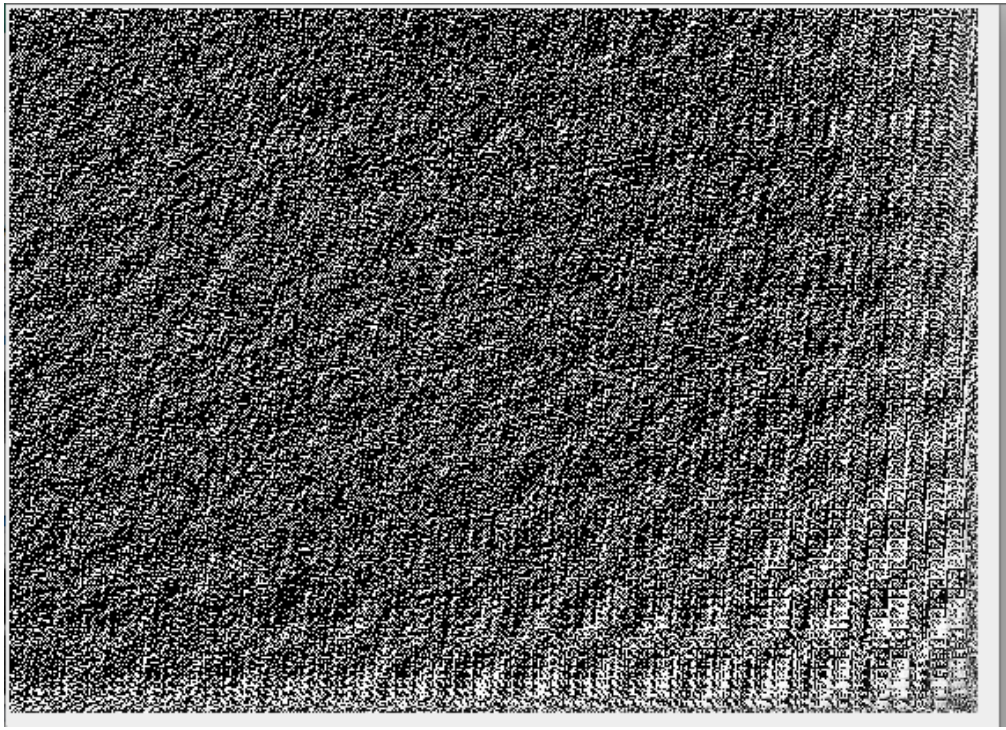


paused here while improving segmentation
and adding a phase congruence
edge and corner detector

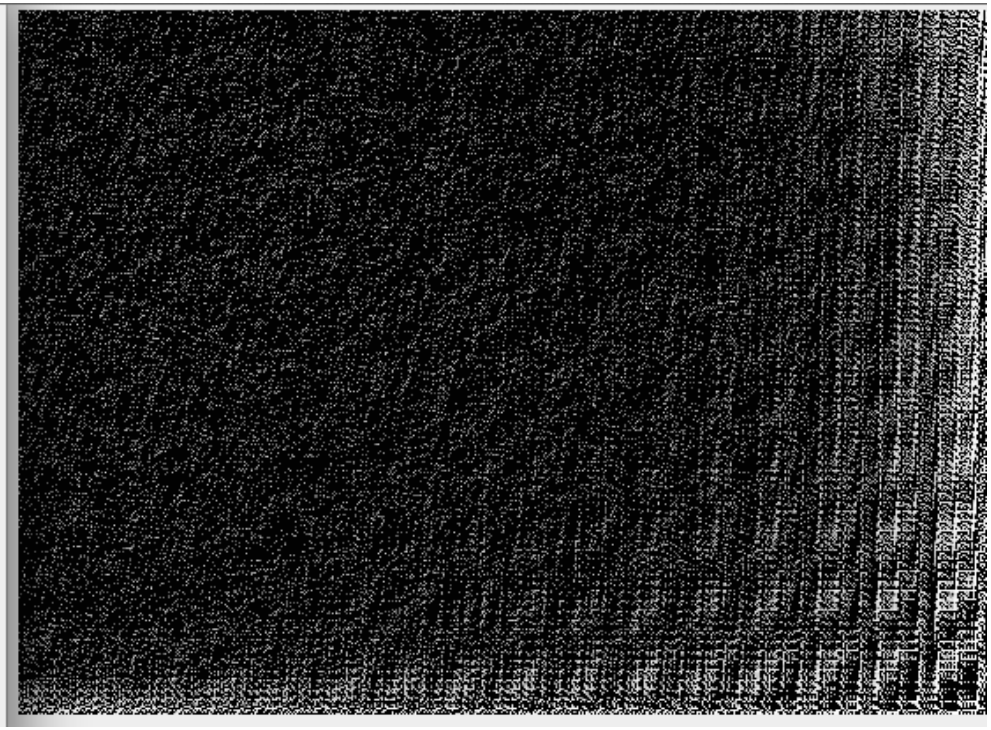




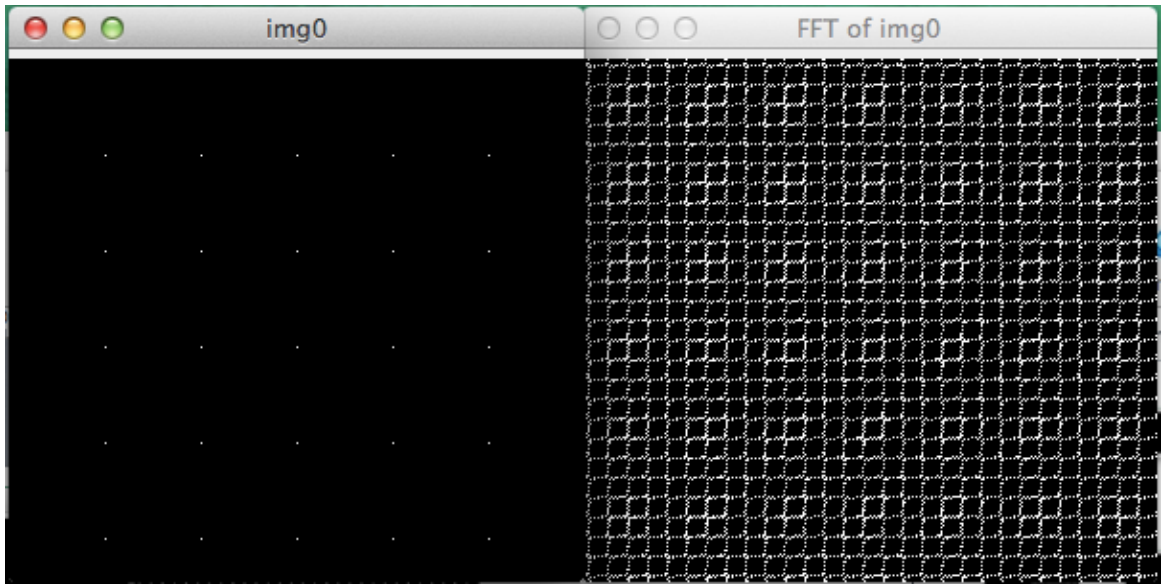
FFT of first checkerboard image.



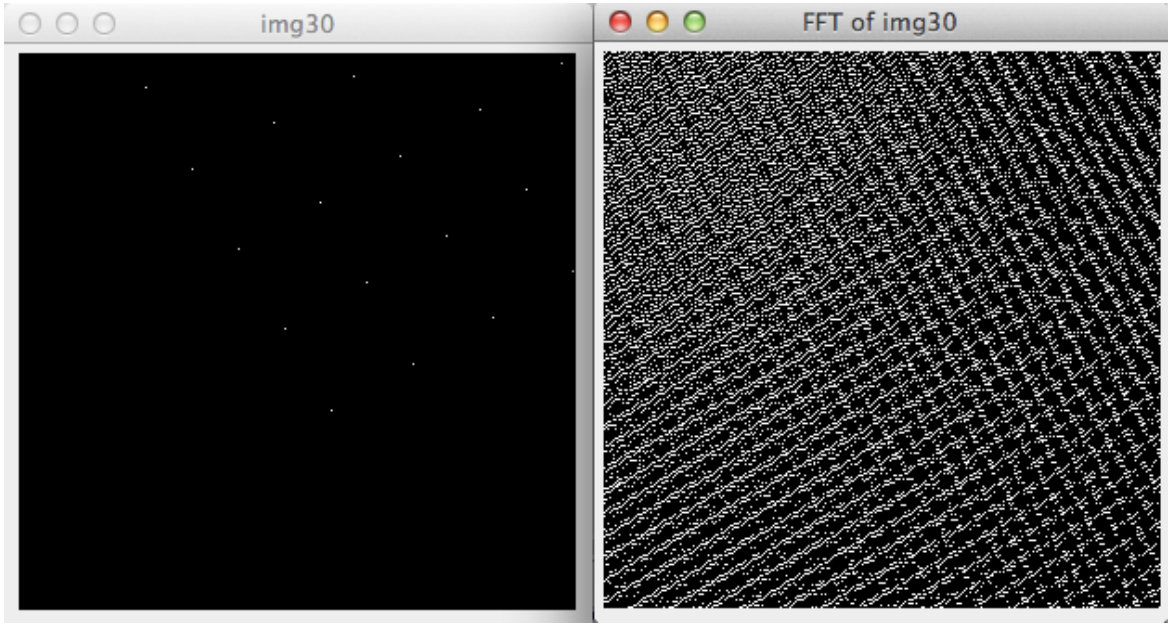
Periodic FFT of first checkerboard image.



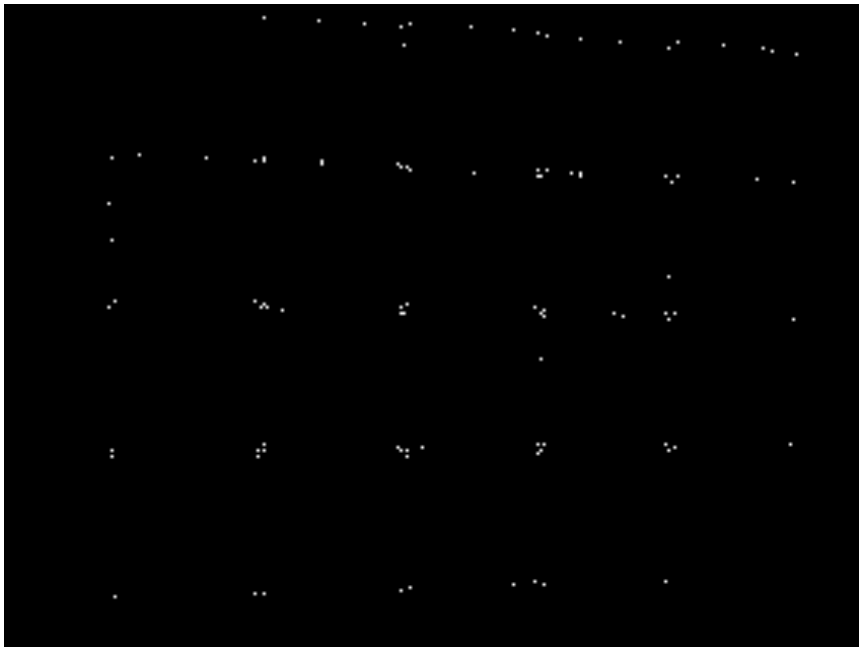
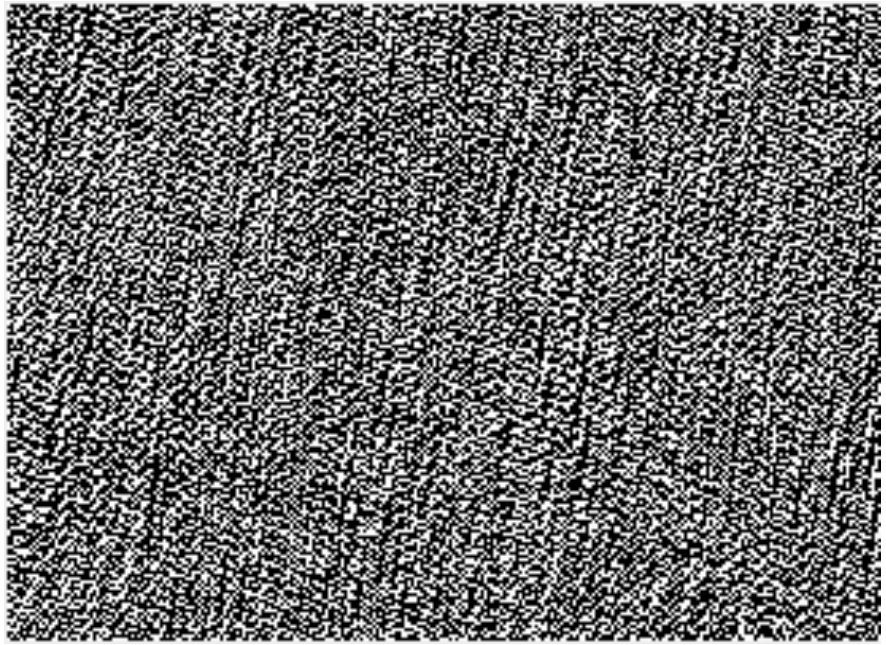
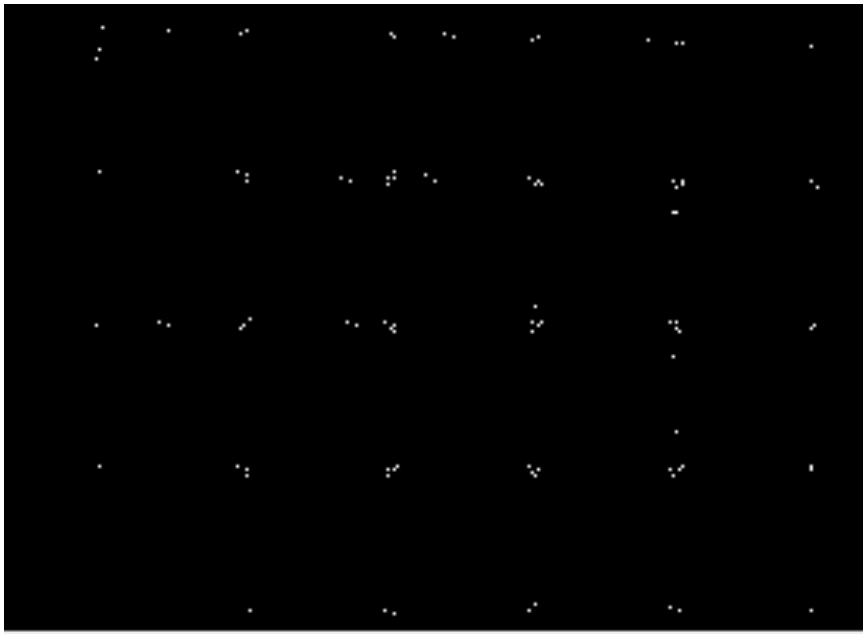
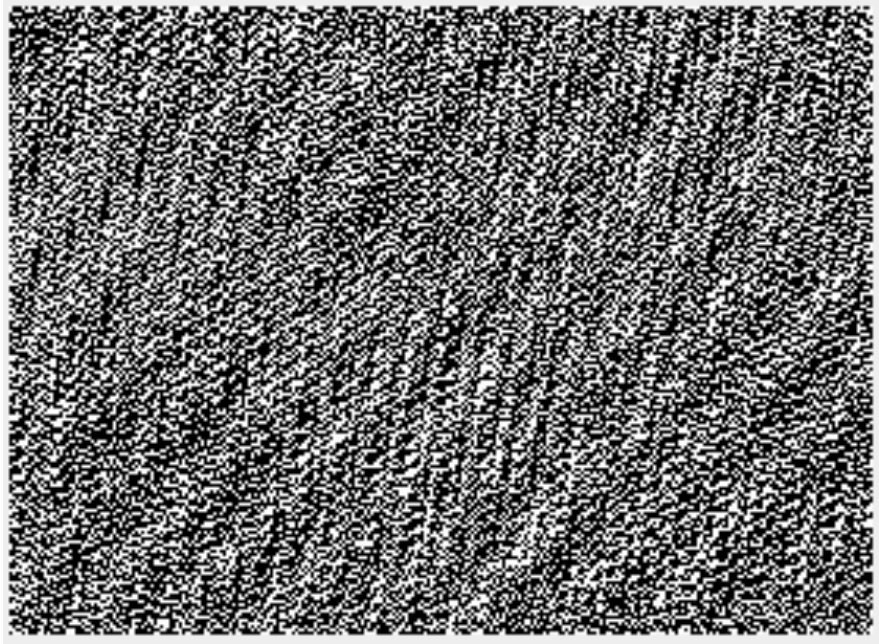
quick look at complexity of using FFT to find textures. needs images to be aligned along axes of interest for simplest extraction of spatial components.



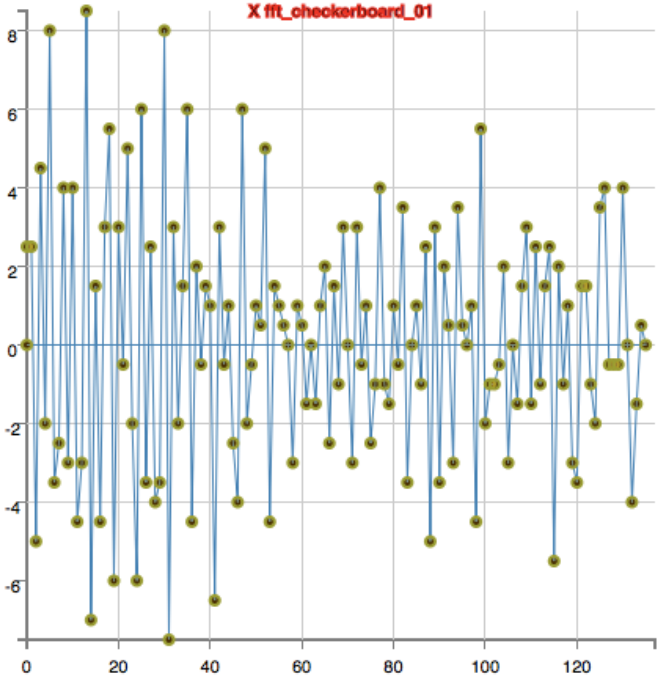
2D FFT of a grid of points

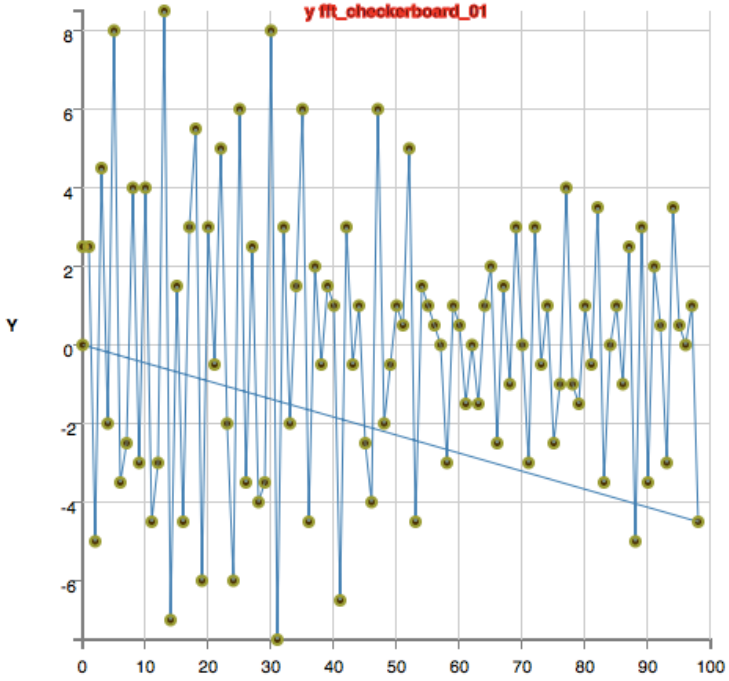


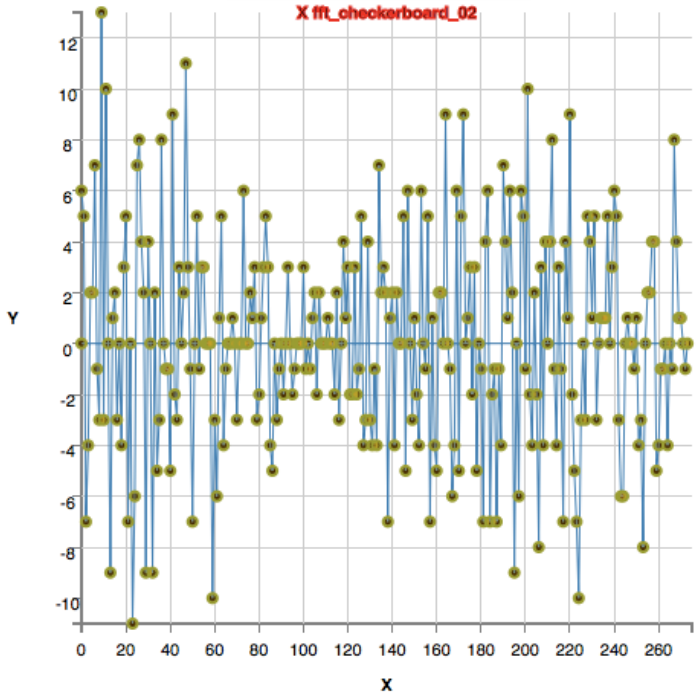
FFT of a same grid of points rotated by 30 degrees

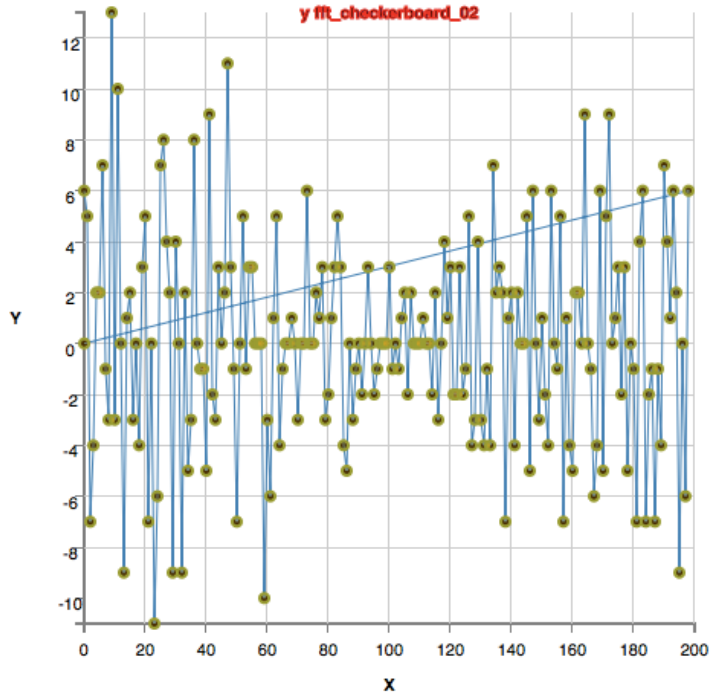





FFT of the corner regions

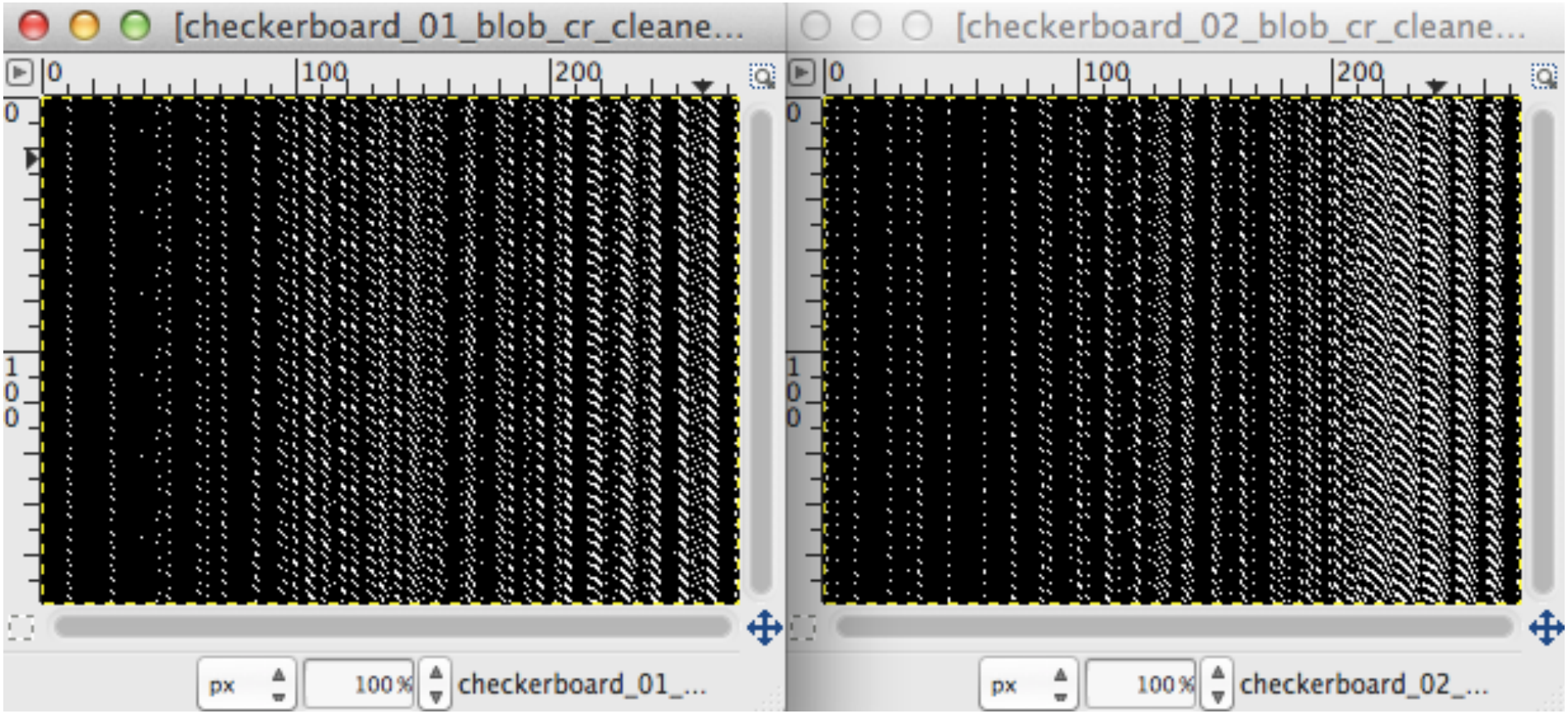




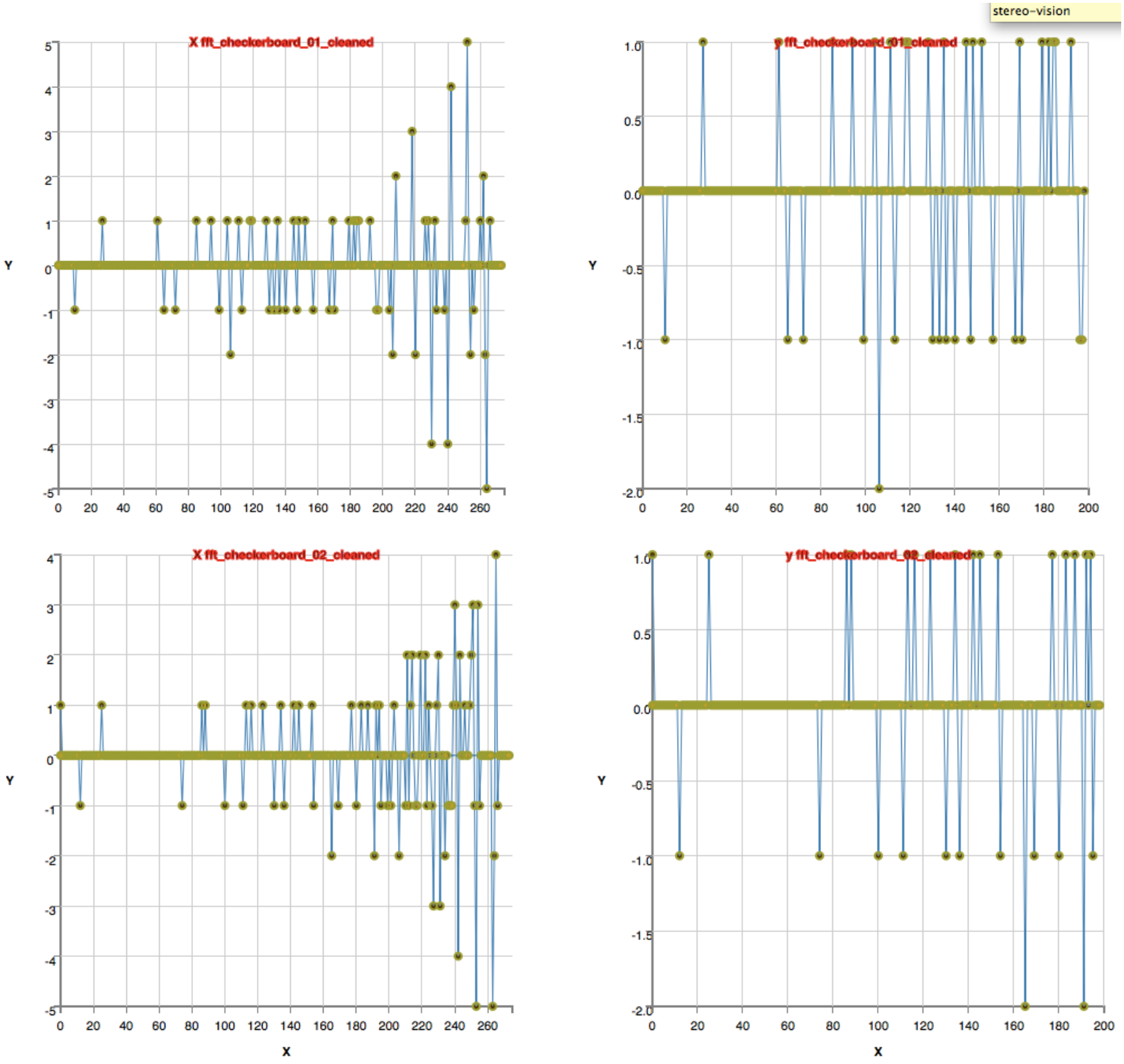




FFT of the corner regions

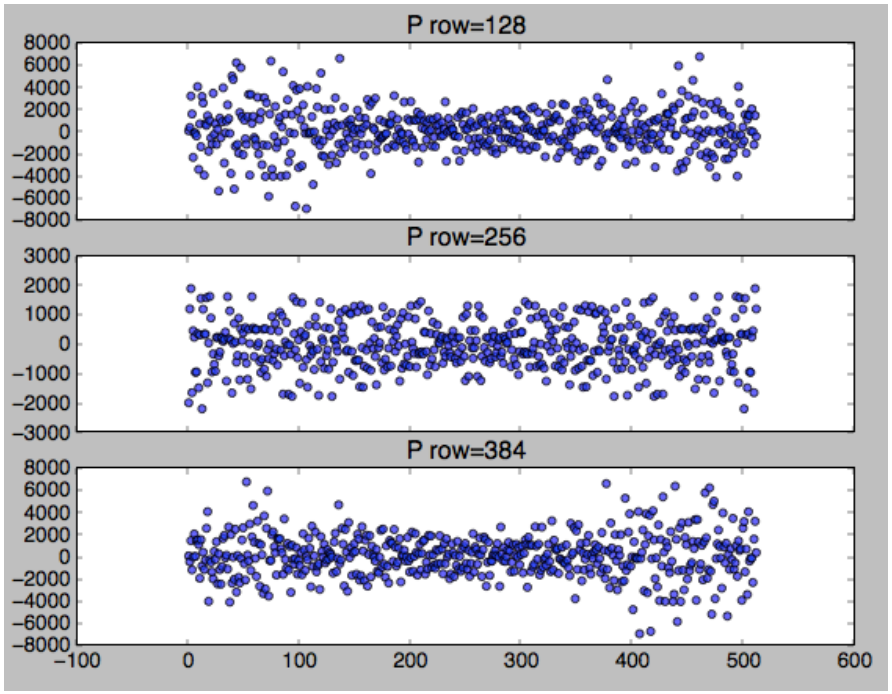


FFT of corner regions (artifact corners removed)

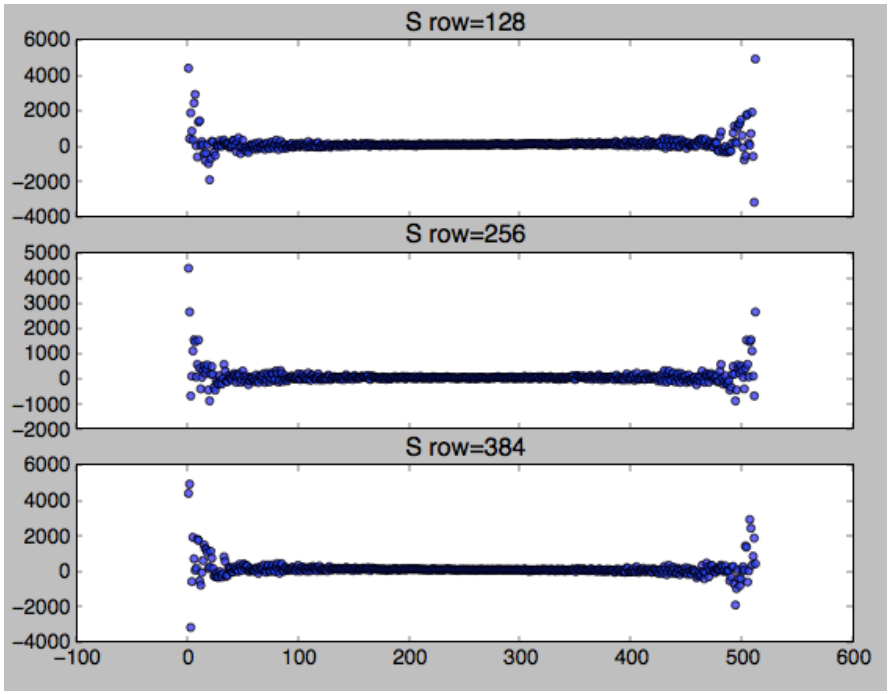


FFT of corner regions (artifacts removed)

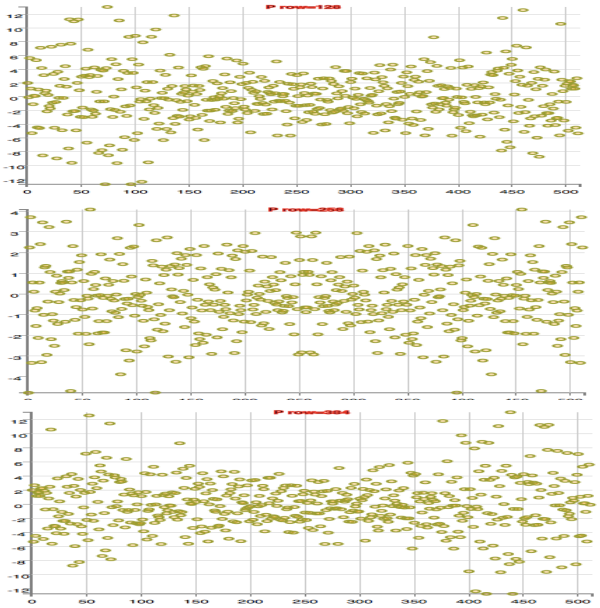
phasepack, lena.jjpg, PeriodicFFT **P**



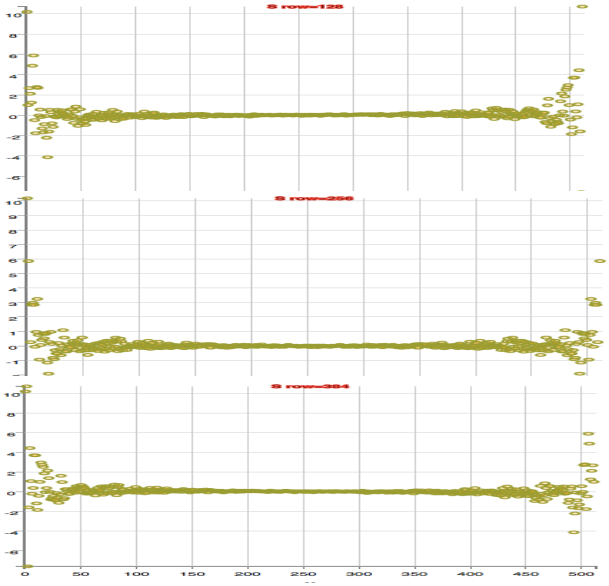
phasepack, lena.jjpg, PeriodicFFT **S**



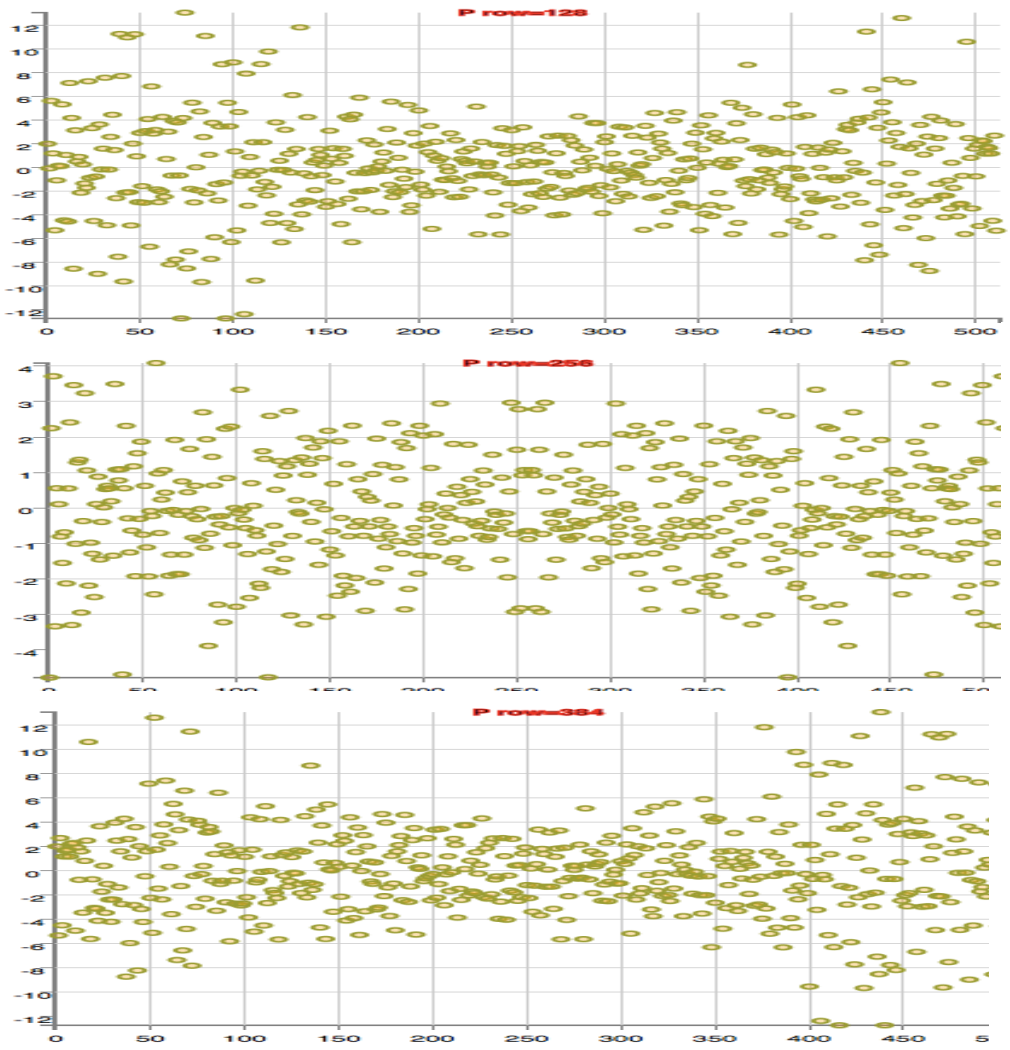
my code, lena.jjpg, PeriodicFFT **P**



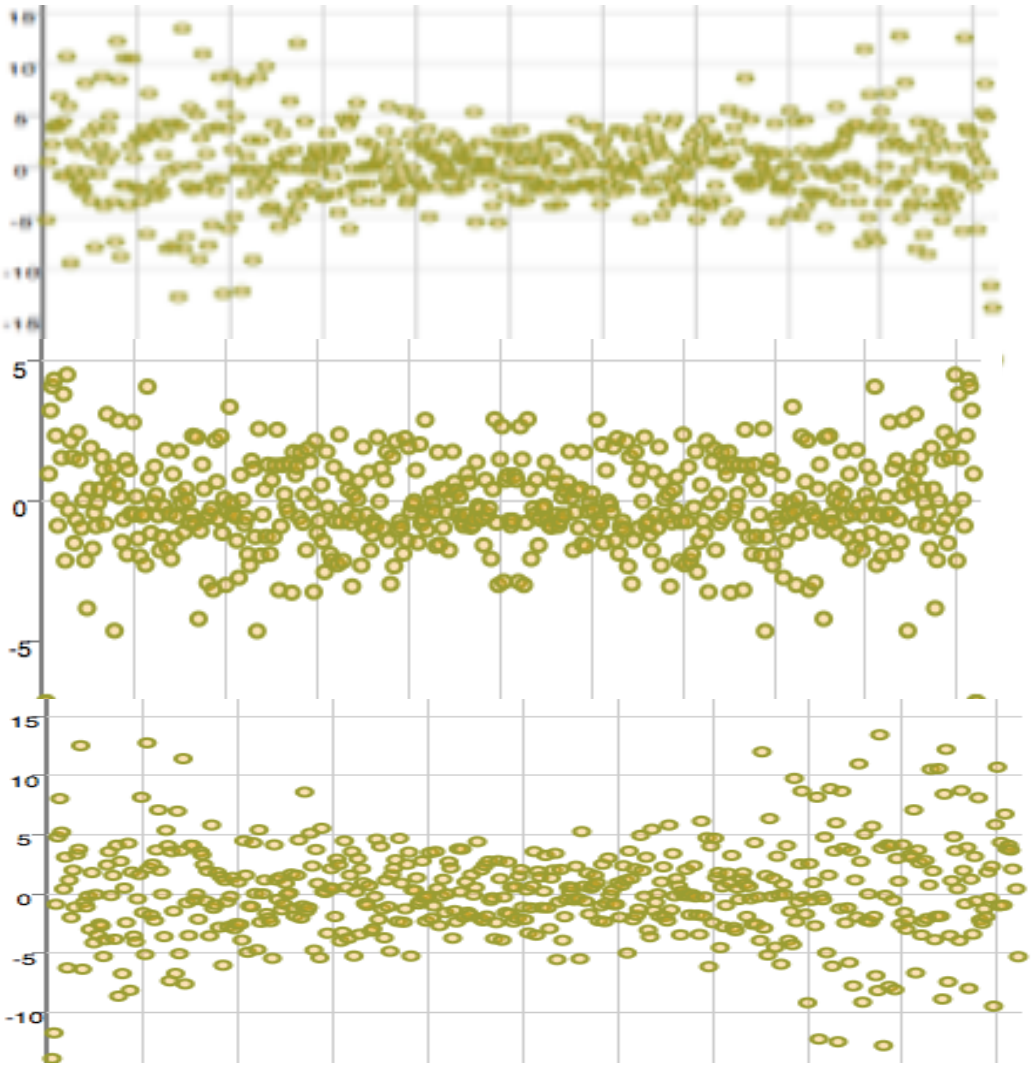
my code, lena.jjpg, PeriodicFFT **S**



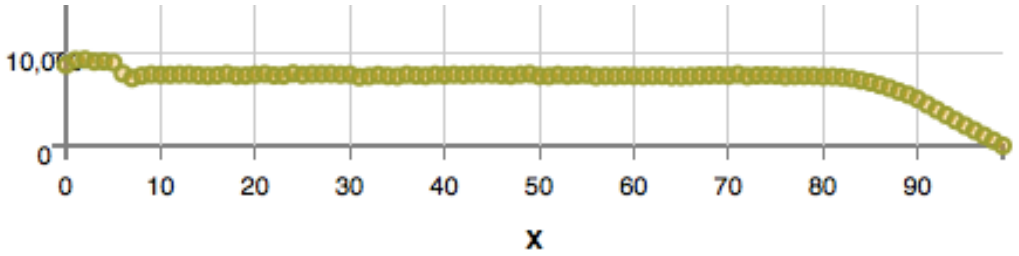
Periodic FFT w/ JFFTPack
P (lena)

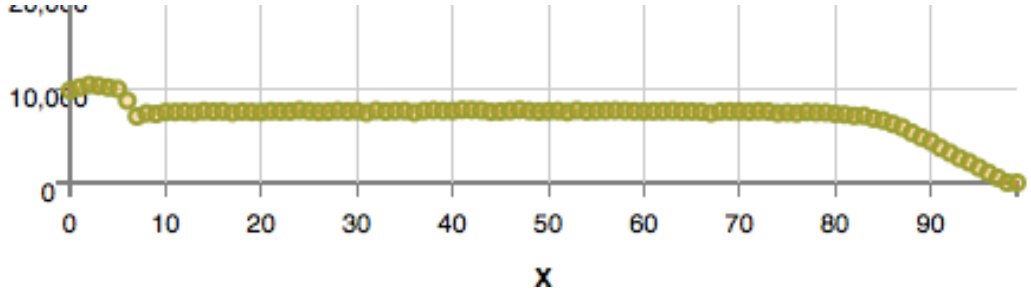


Periodic FFT w/ efficient
power of 2 FFT
P (lena)

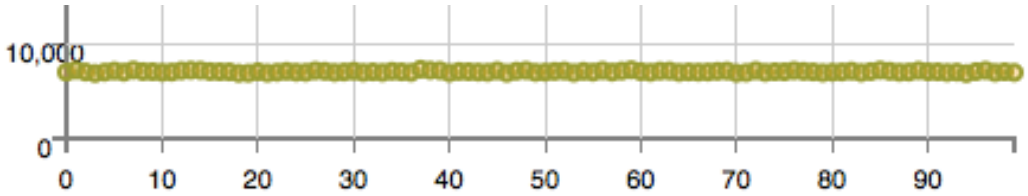


On a different topic, looking at improving the random sampling for the RANSACSolver.
Created a k bits random number selector for a distribution of indexes and
created the same for a distribution of indexes altered to replicate the number by its value.
Plotted is a look at the numbers drawn from 100,000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





The new methods have a few biases due to a bug in BigInteger's random. (filed a bug report with 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.