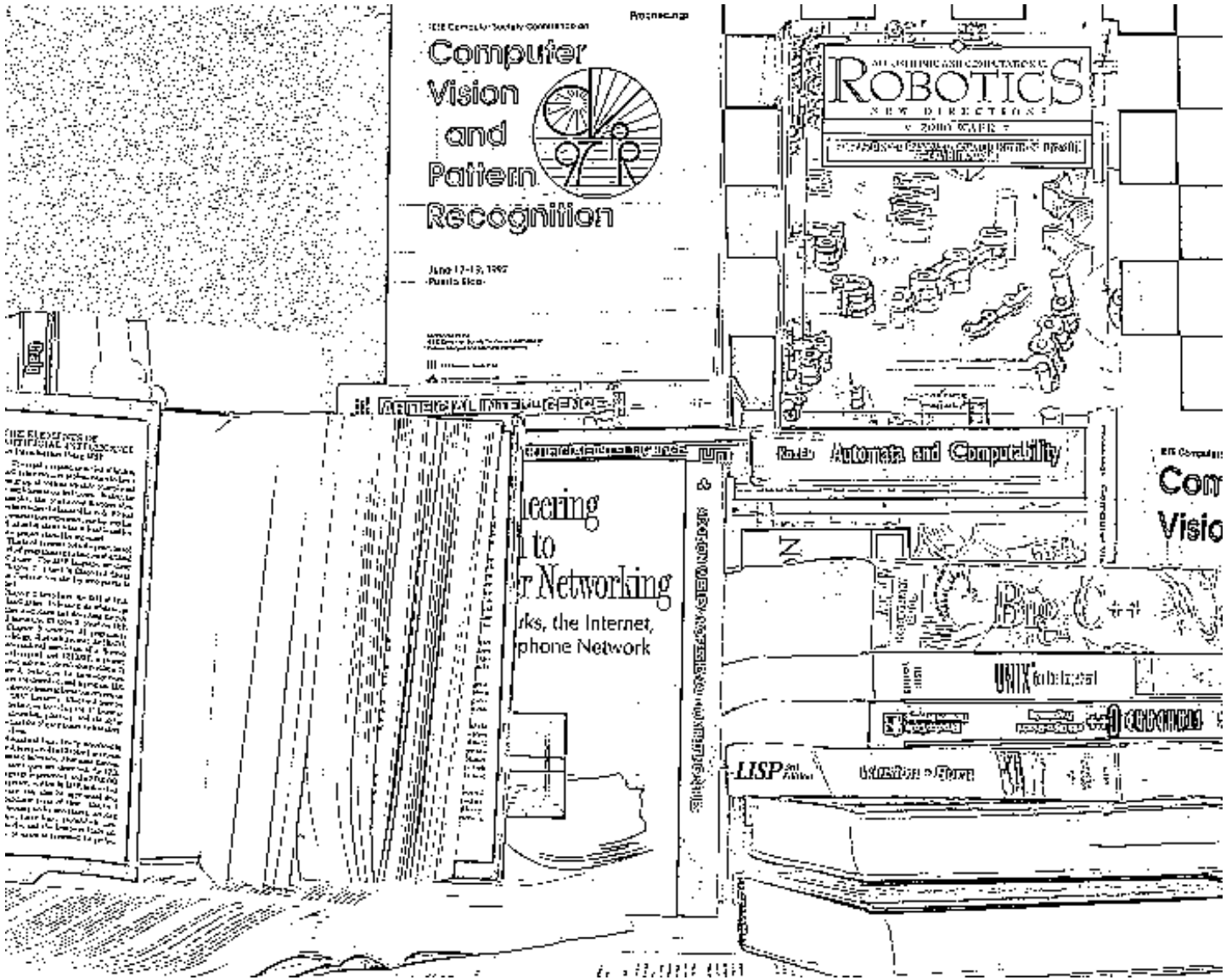
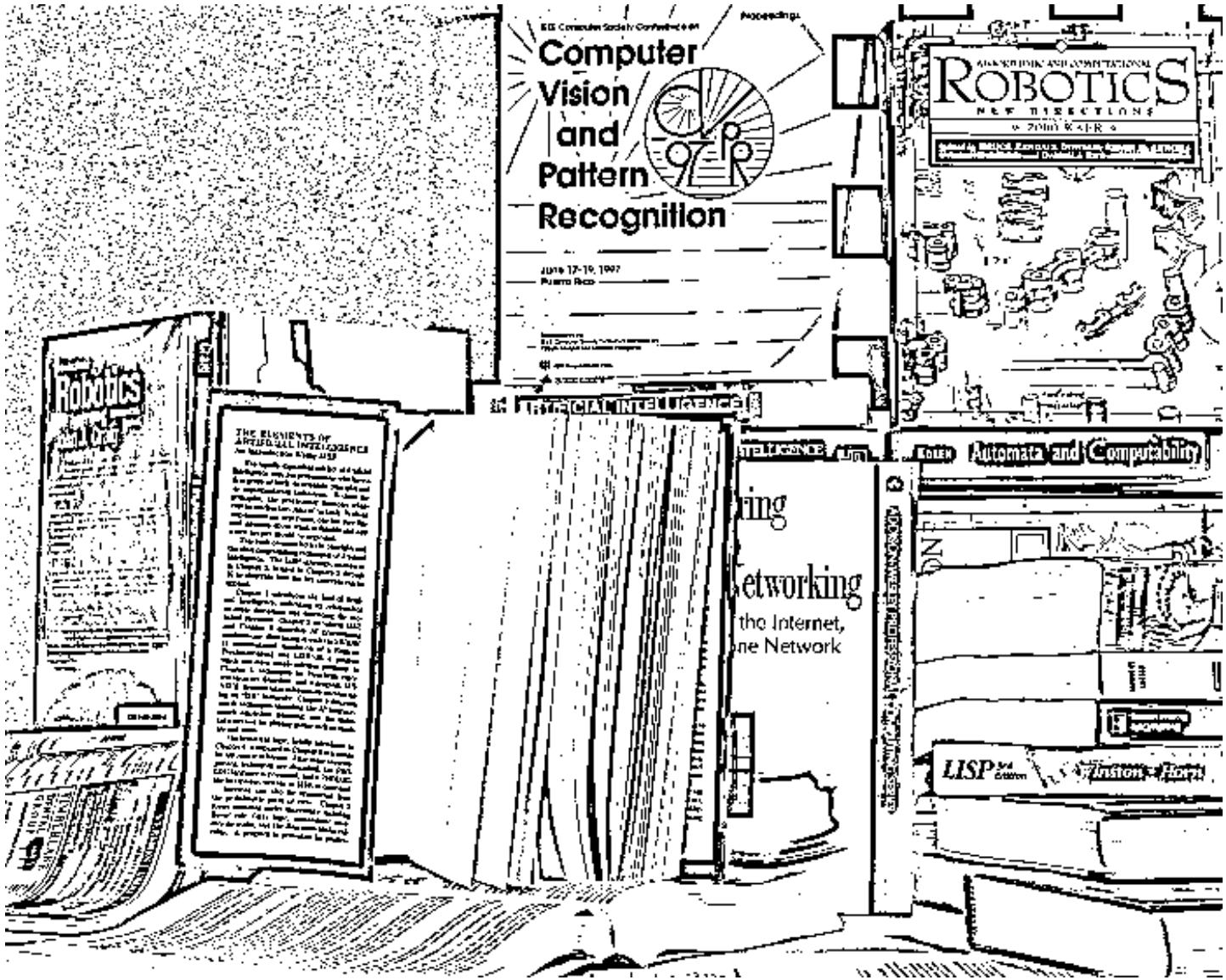
various snapshots while building classes to match an object in another images follow.

Note that the document should be/will be moved to name segmentation as shape and motion are used in addition to color and illumination.

adaptive means, h=1

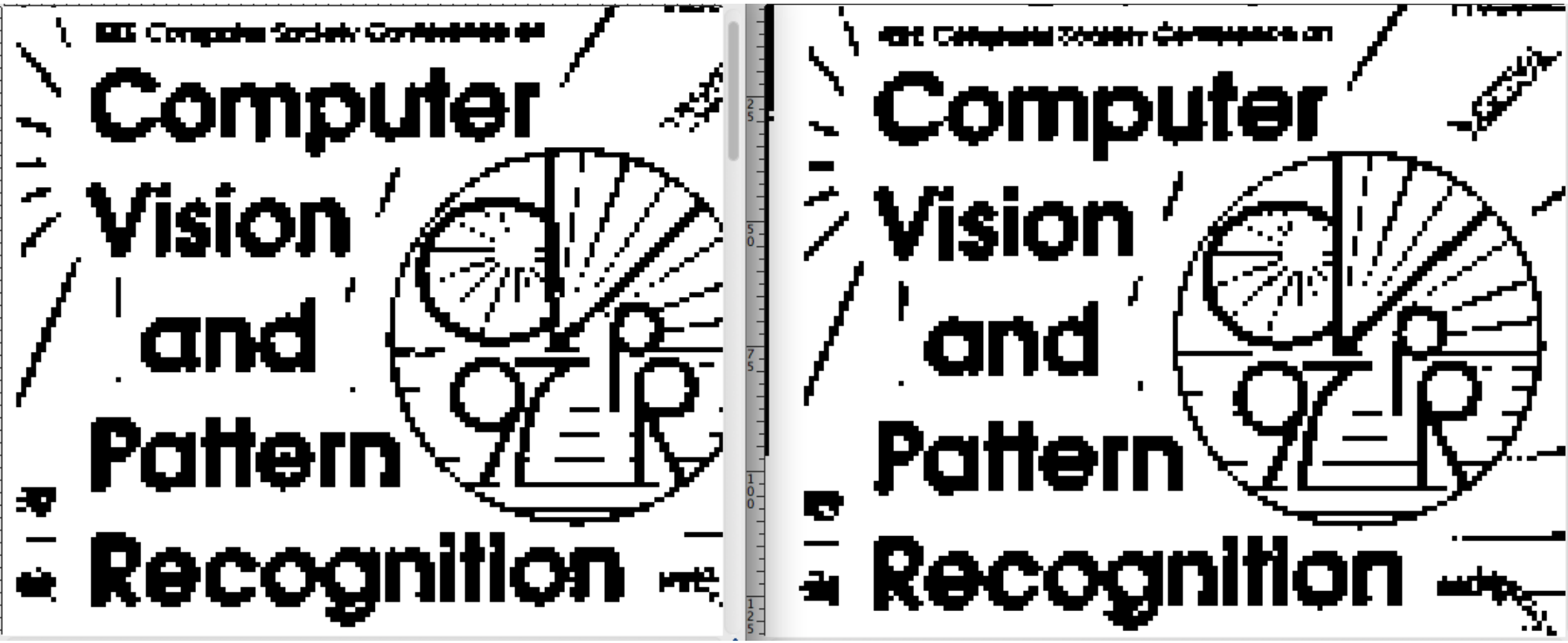


adaptive means, h=3

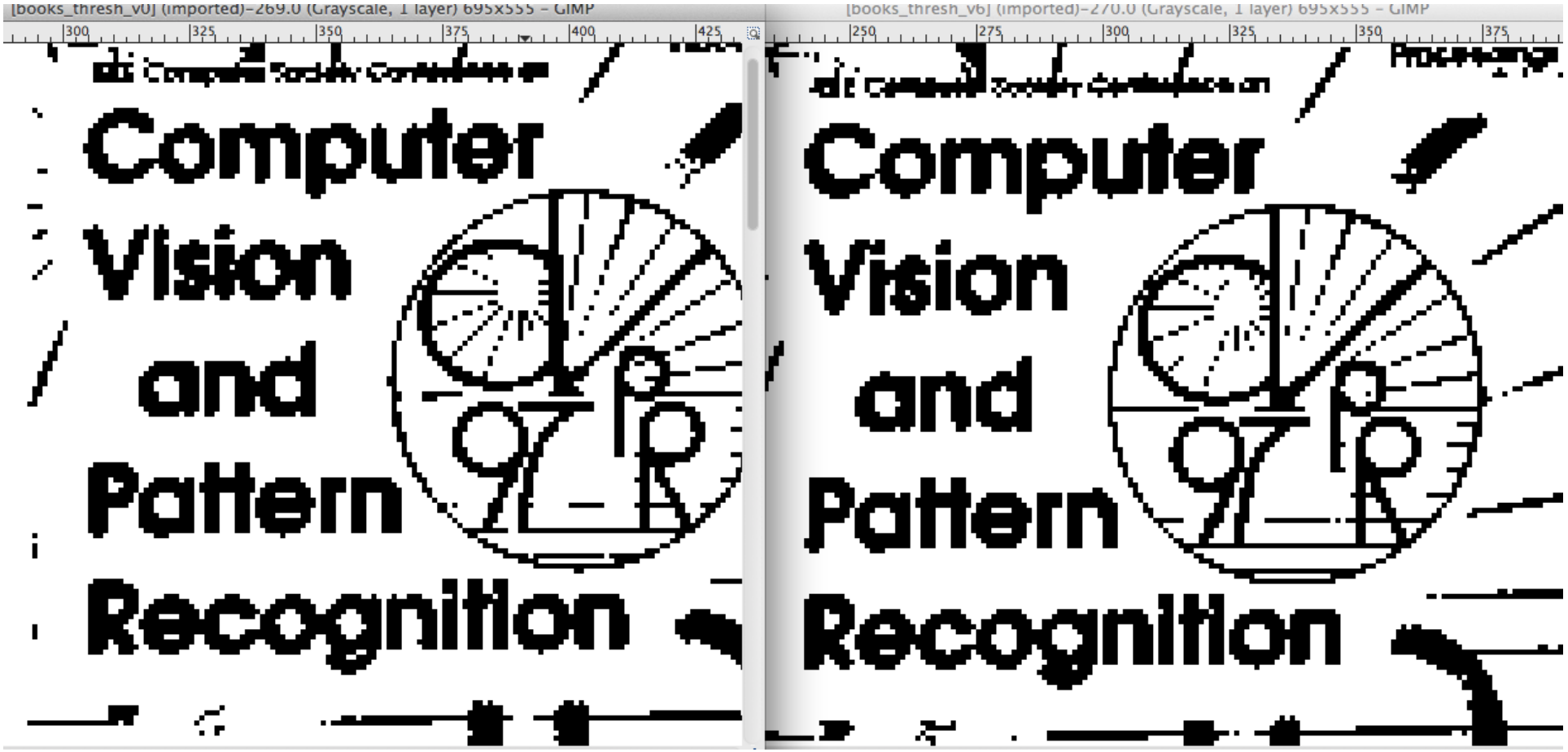


adaptive means, h=5

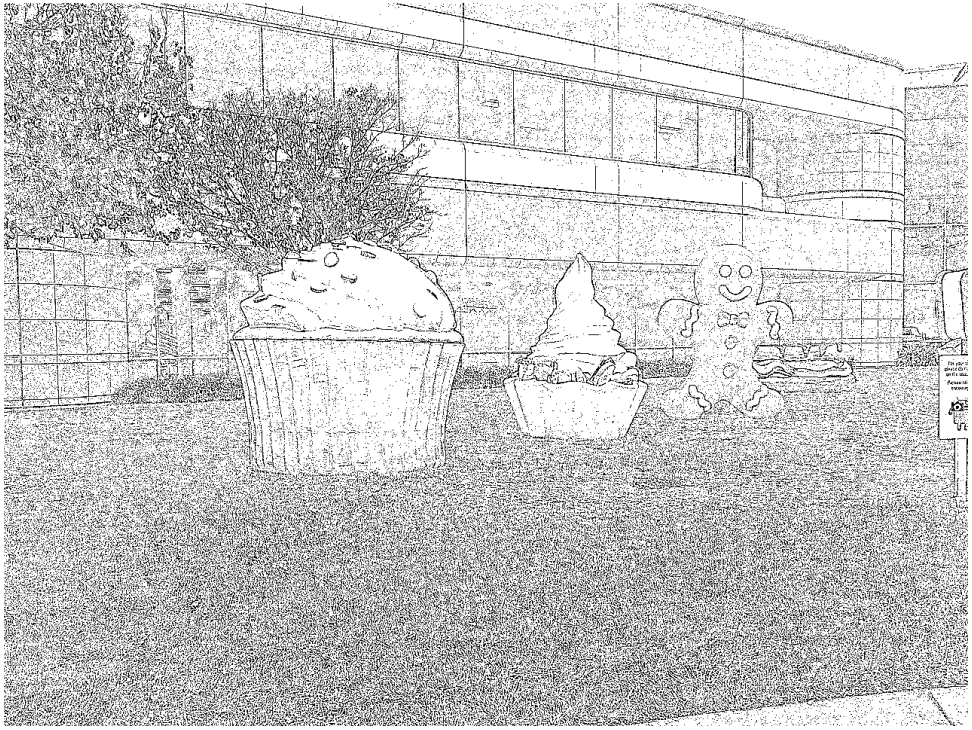
adaptive means, h=7



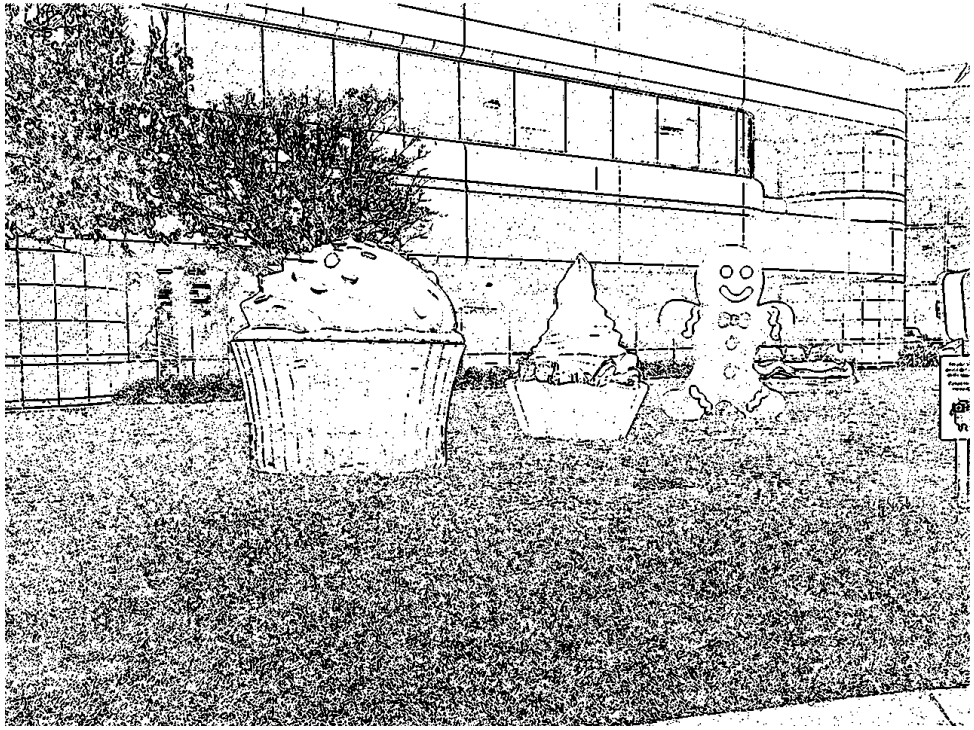
adaptive means, h=11 for adaptive to join close letters



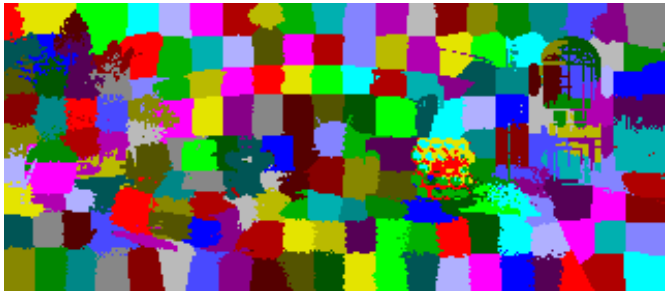
adaptive means, h=1

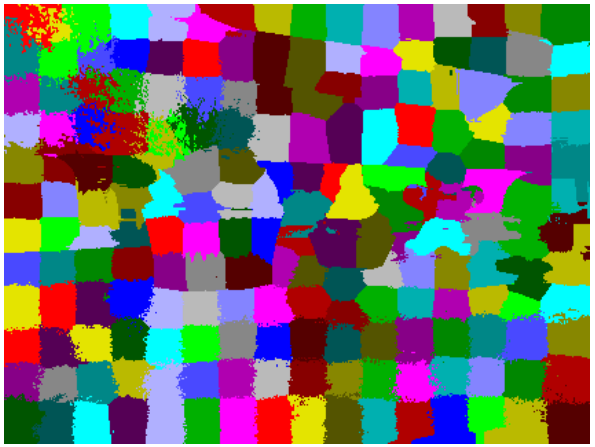
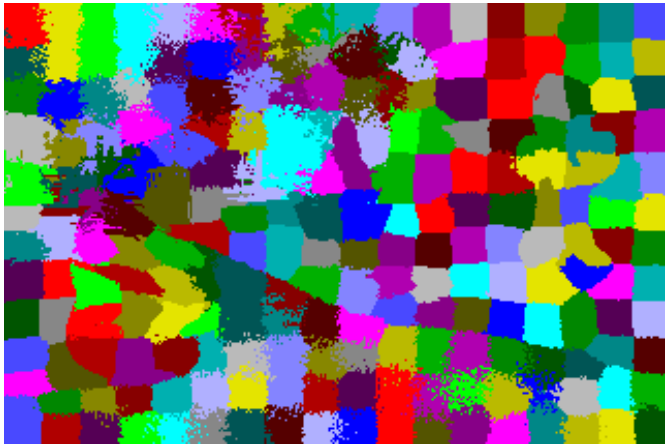


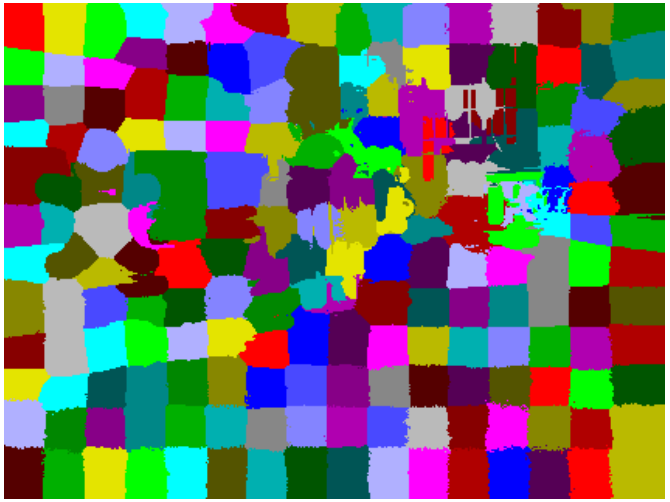
two median smooth with h=2



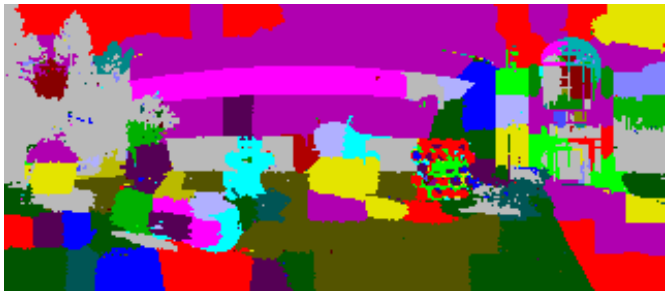
SLIC super pixels algorithm

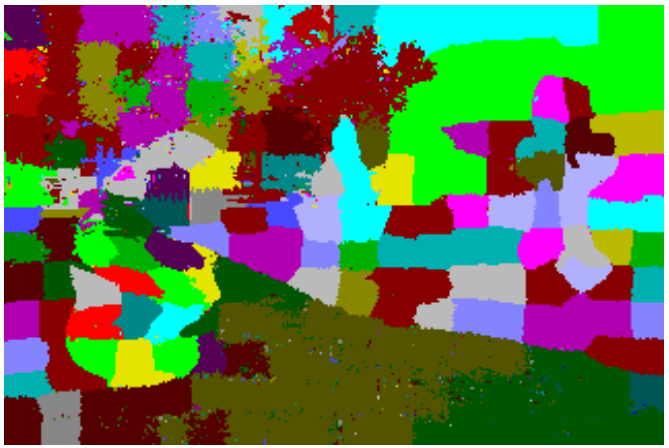


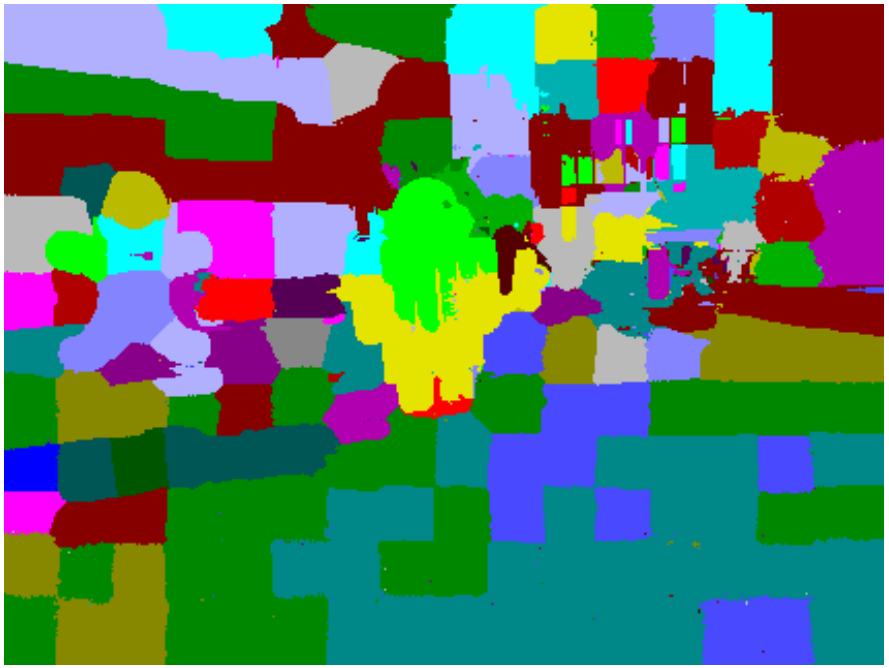



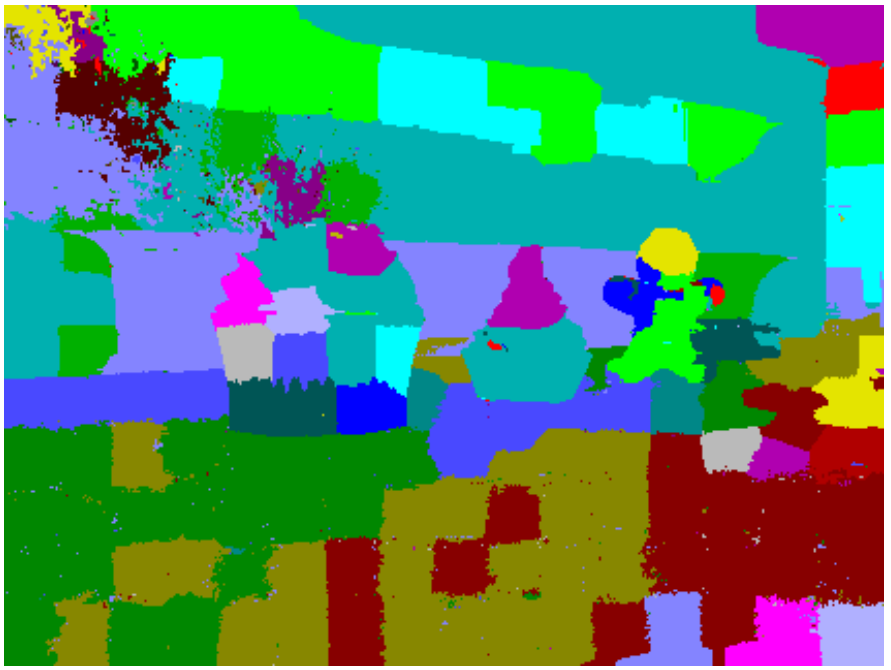




CIE XY Theta Clustering of the super pixels

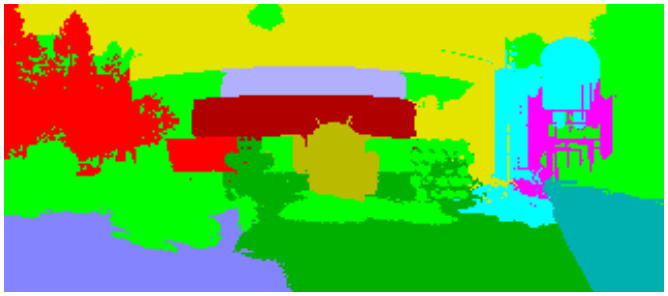


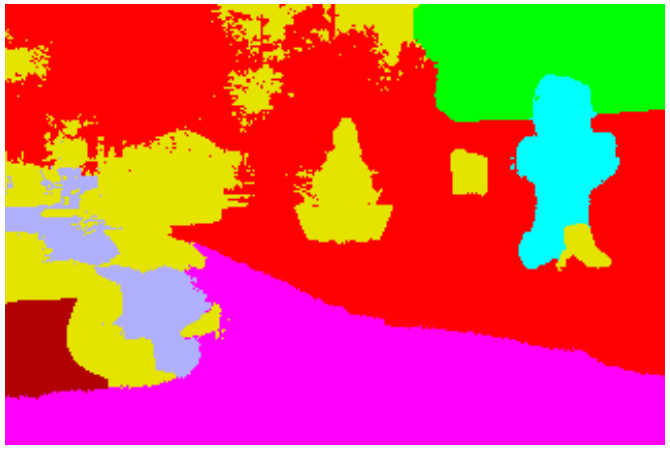


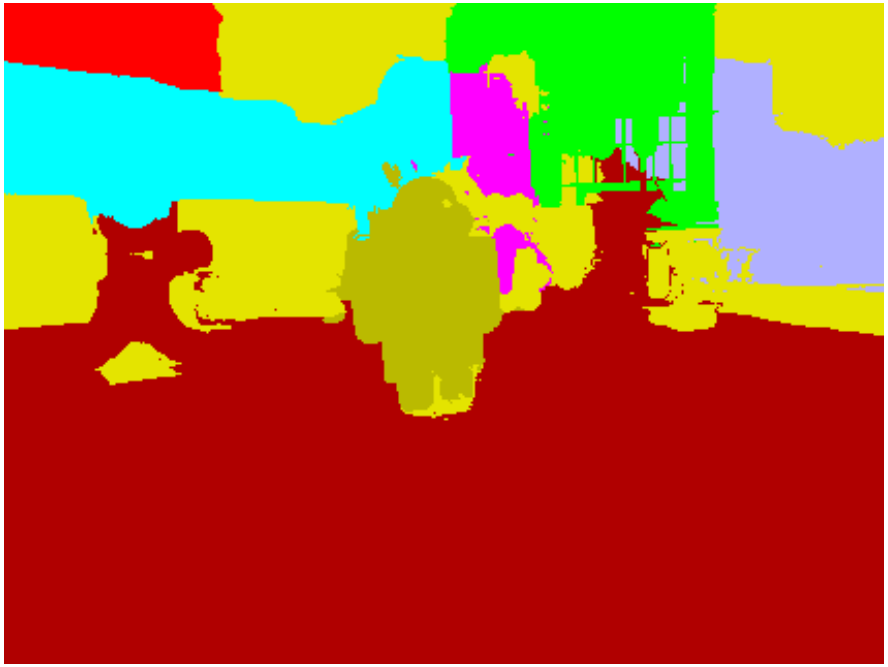


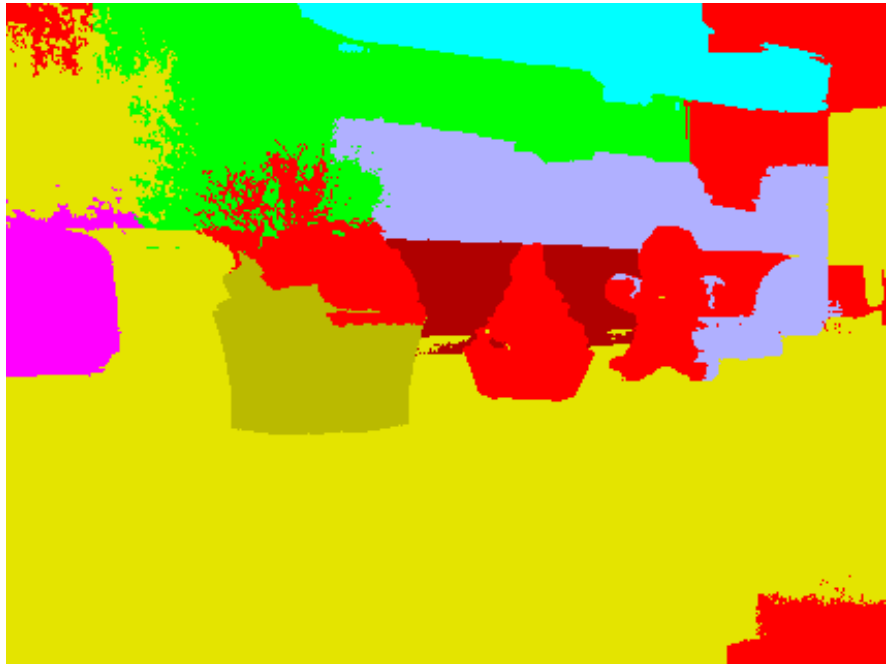


Normalized Cuts w/ HSV cs on the super pixels

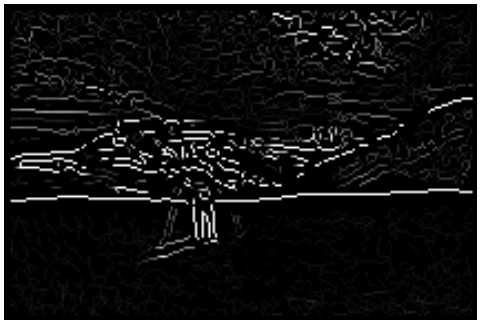




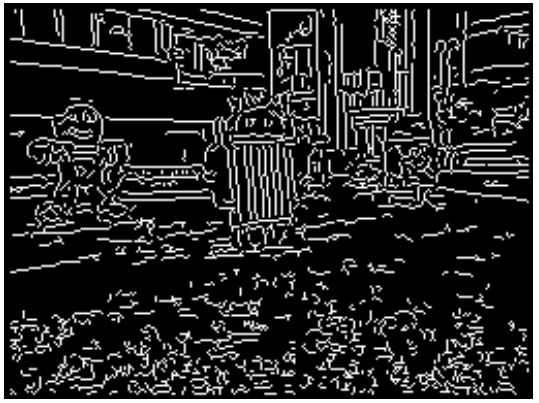


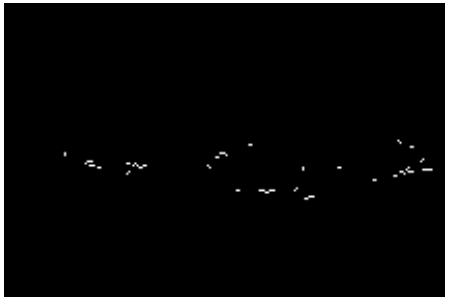


canny greyscale

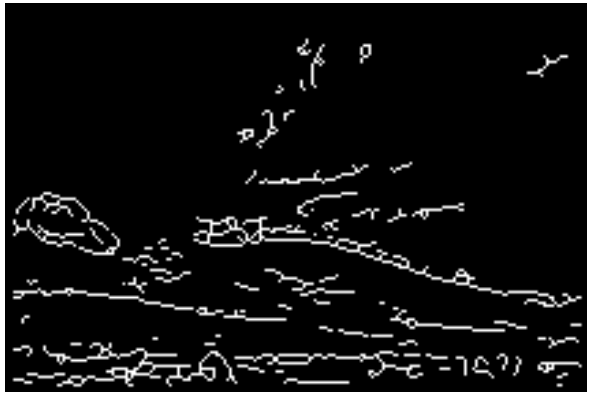


canny deltaE 2000





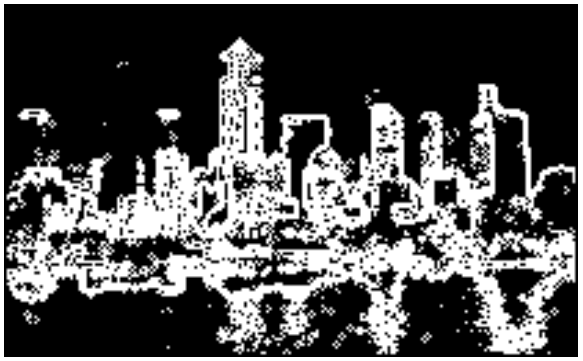
phase congruence

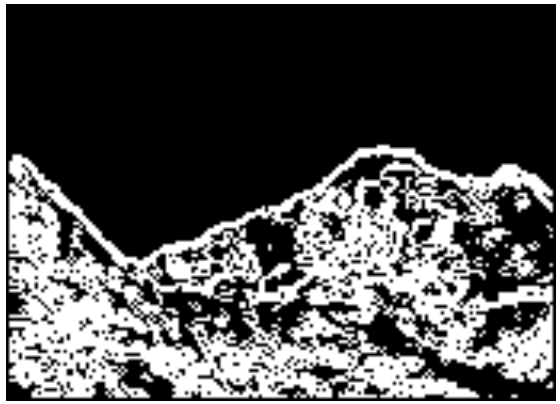


binary gradient “H” of LCH with 20 degree threshold

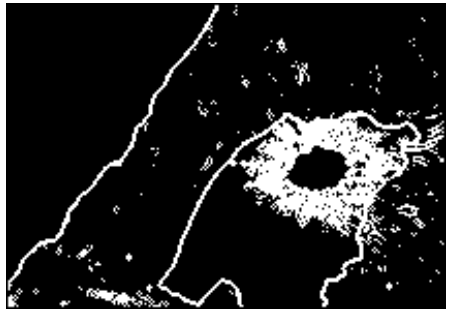


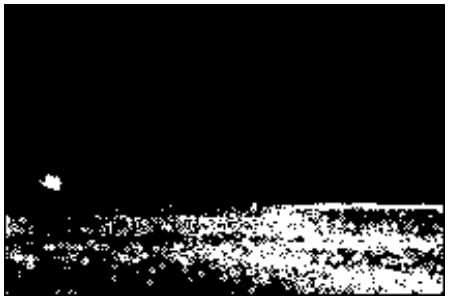




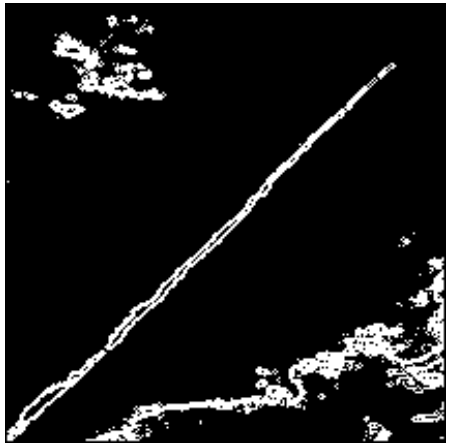
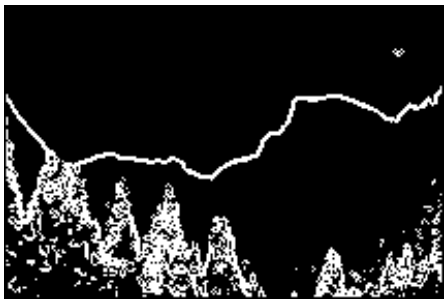



binary gradient “H” of LCH with 20 degree threshold





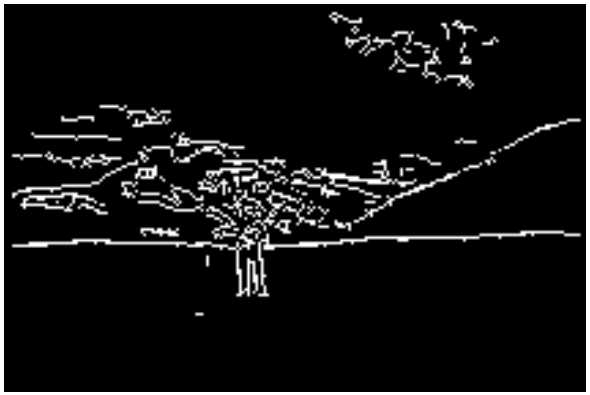



Canny edges of “L” and “C” from LCH colorspace







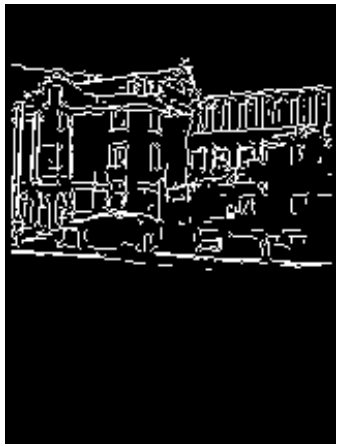


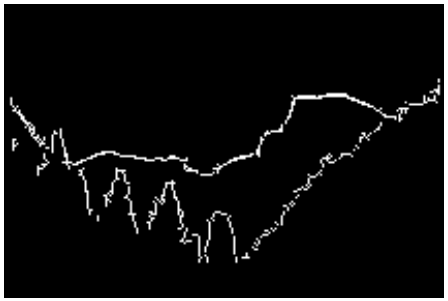

Canny edges of “L” and “C” from LCH colorspace

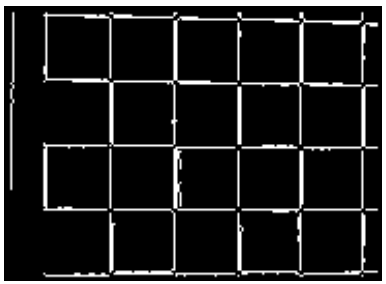




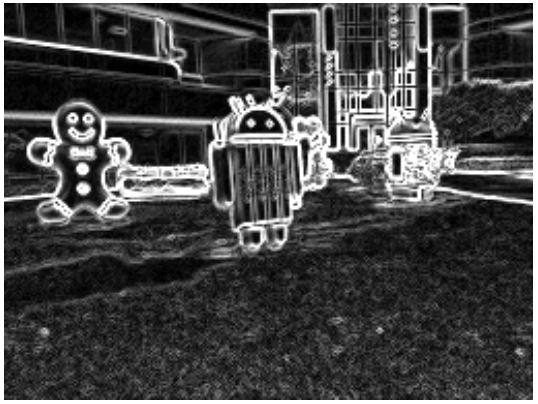


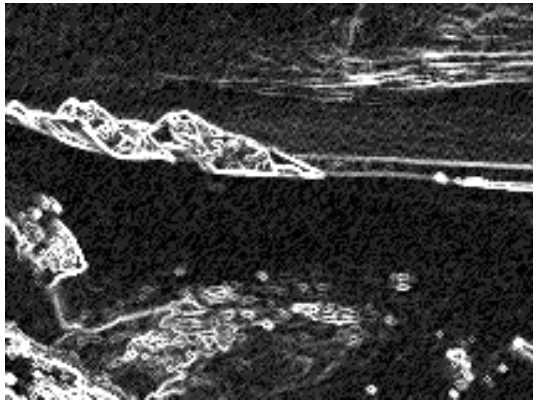


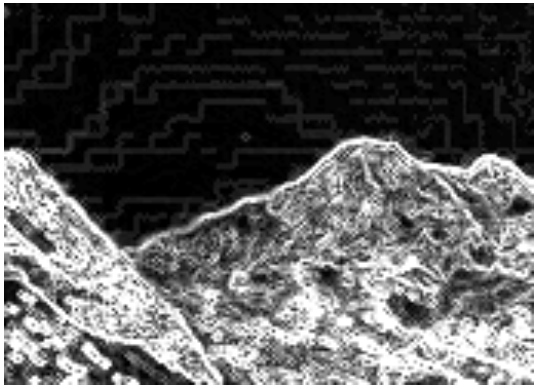





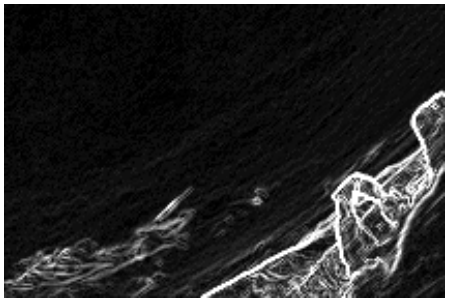
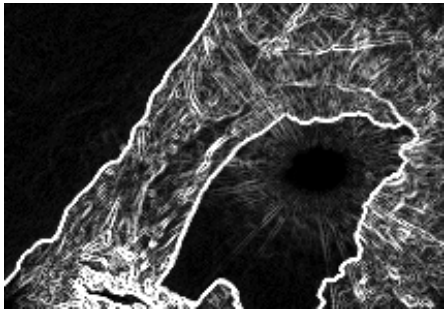
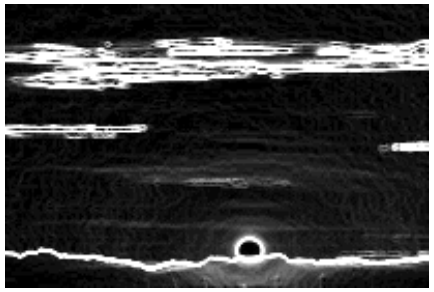
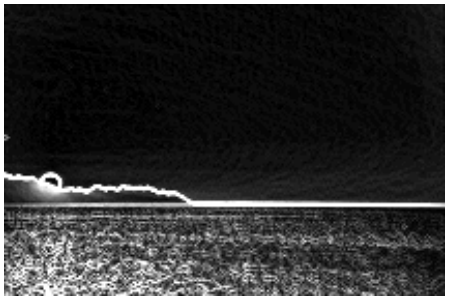
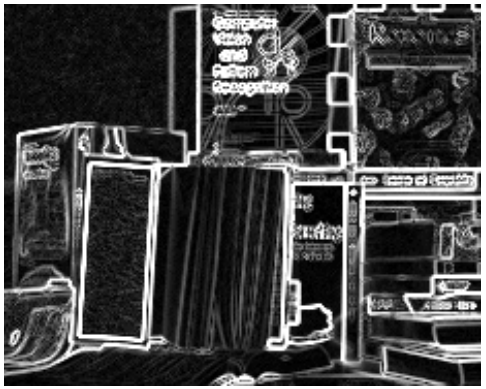
Sobel edges of “L” + “C” from LCH colorspace (stretched by x 4)

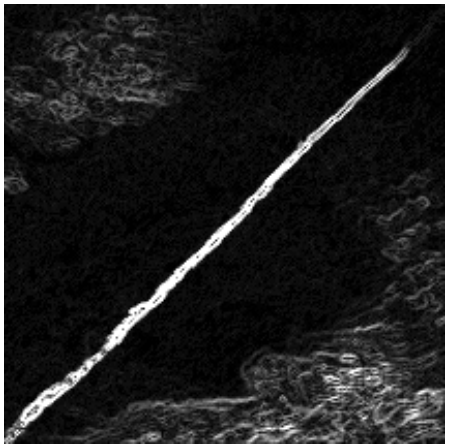
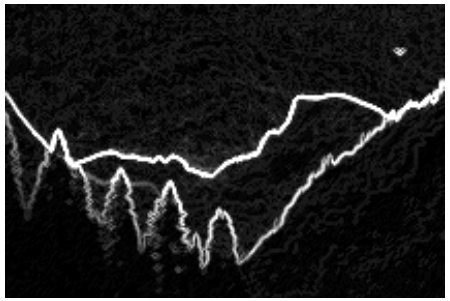
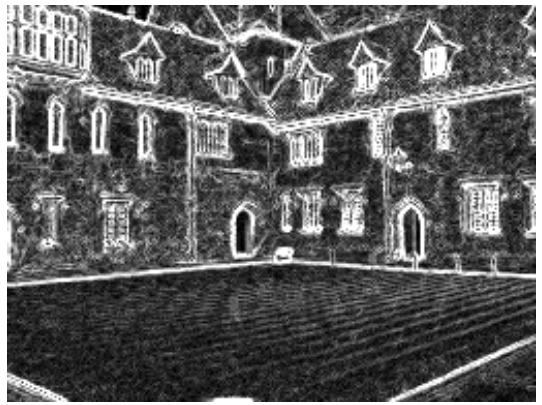
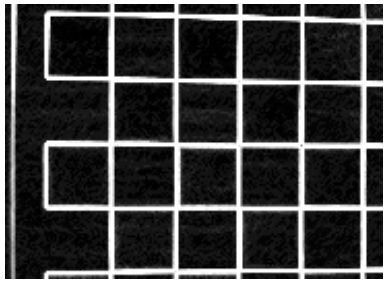




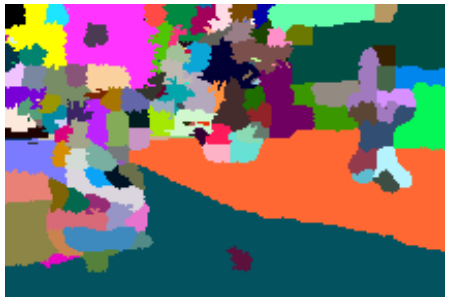


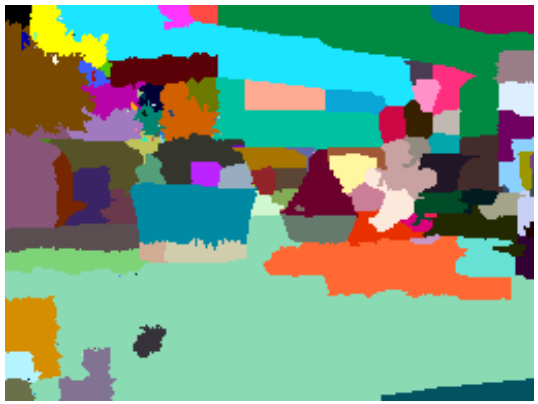
Sobel edges of “L” + “C” from LCH colorspace (stretched by x 4)

super pixels and moderate merging by HSV color

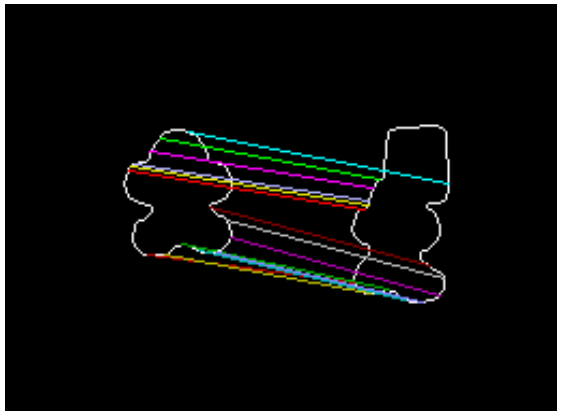



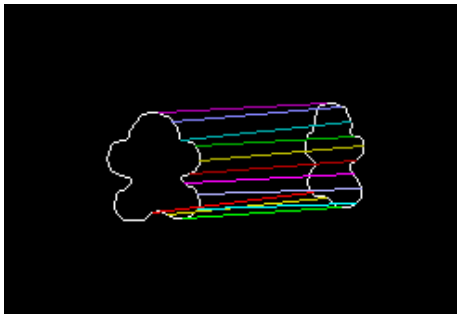


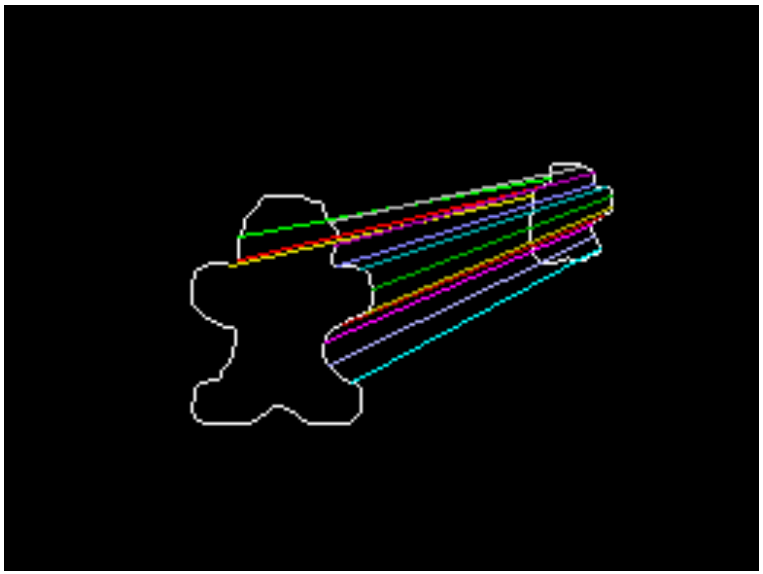


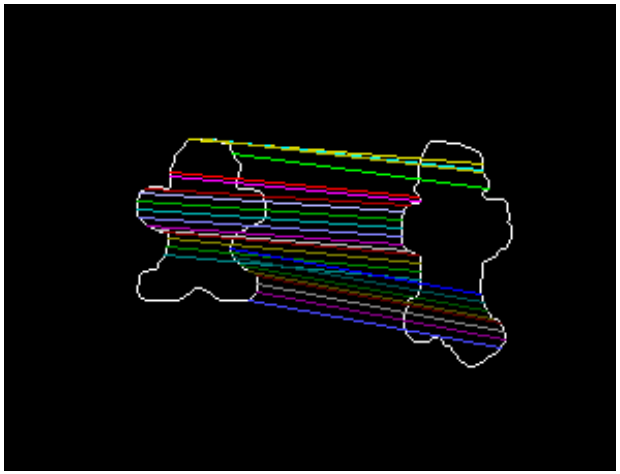
Partial Shape Matching of the gingerbread man

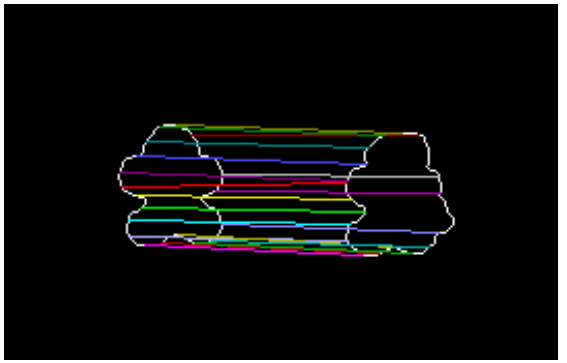
extract ordered borders of shapes, blur them by one or two sigma, then ordered bipartite matching algorithm using difference of chords as a descriptor.











Shape Finding in over-segmented images

Exploring using the partial shape matcher and a local aggregated search pattern within a global search pattern to find an object by shape alone.

The over segmented labeled regions tend to be a large enough number that thorough combinations of labeled regions within a total object size range is often infeasible.

Filtering the labeled regions by information that is thought to be conserved between images can be done to filter the regions.
A couple of the tests use HSV and CIE color filters.

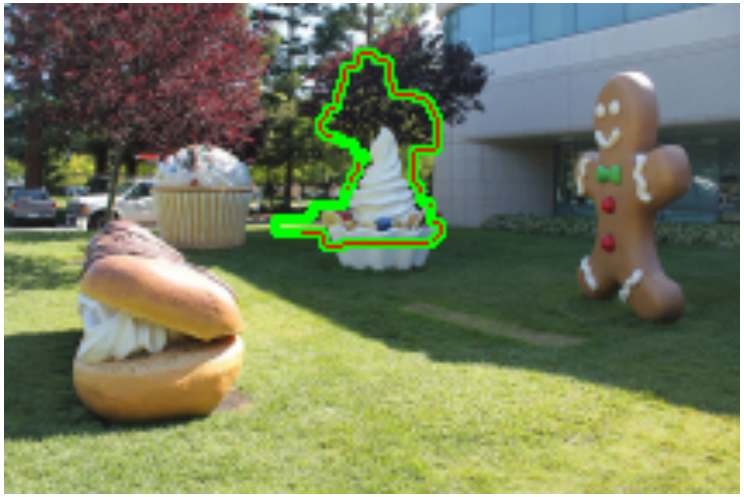
A texture filter can be built from the template object image (for each pyramid image), but this has a slightly large runtime (add complexity here).

The search using just shape on the filtered labeled regions has a long runtime complexity (add details here), and the true match is within the top results, but is not always the top result.

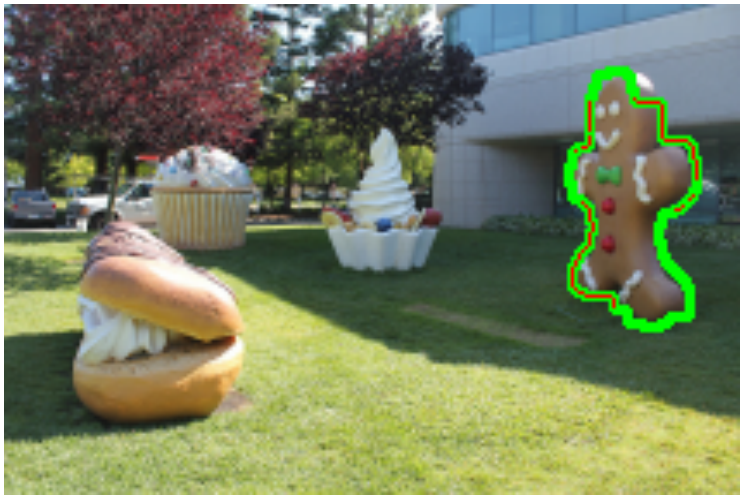
(a shape descriptor applied after top objects is a faster approach…see subsequent results)

top matches to template shape within a test image

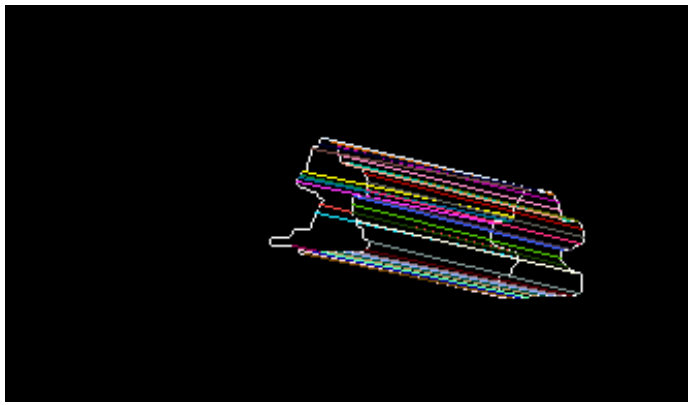
best match and true match

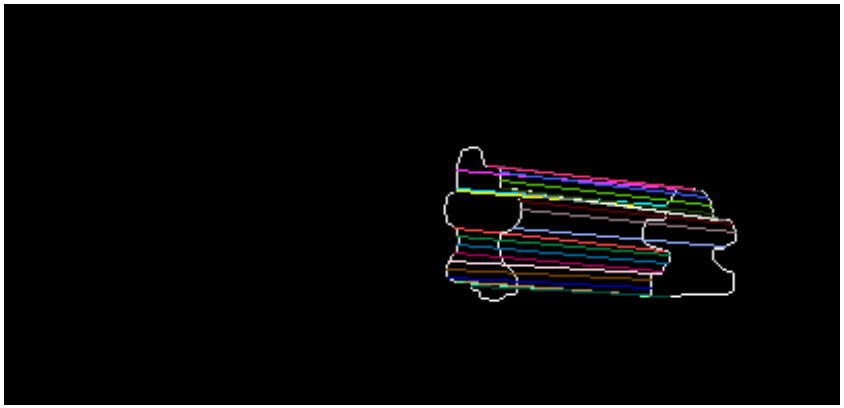


the min cost match to the template shape



the true match has higher cost. (due to scale, and pose)





ORB key points and descriptors

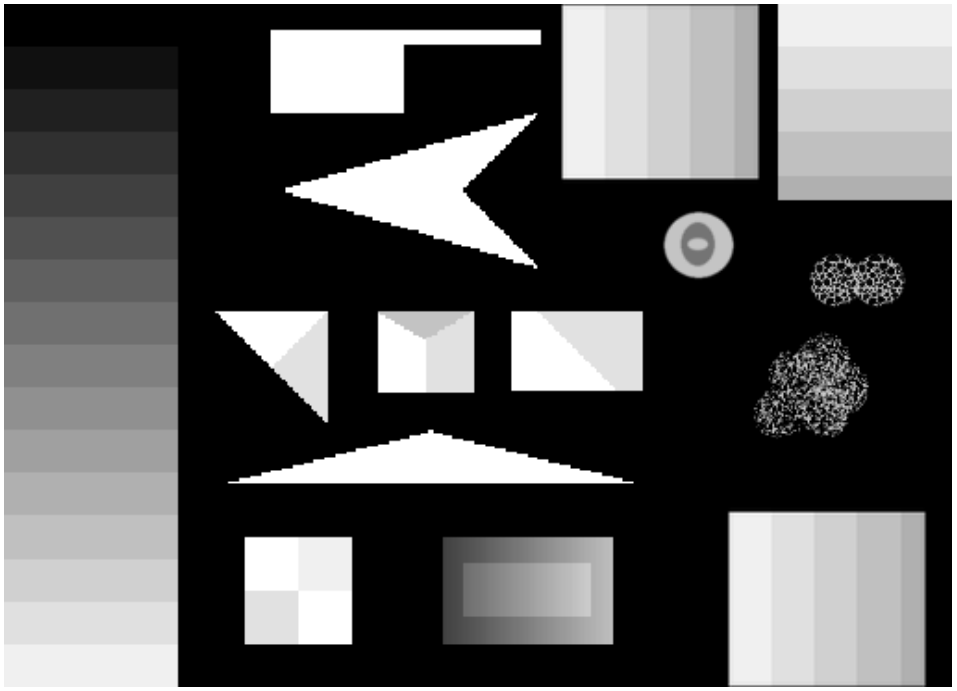
The key points for the template are at the left and on the right are key points filtered to those present in the segmented cells of the image to search, those cells were filtered by color histogram intersections with the template gingerbread man.

texture

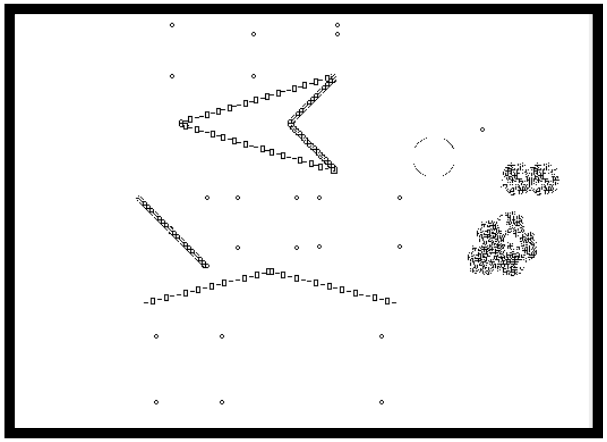
The highly textured regions such as vegetation produce many key points and those need to be filtered out for this case.
The Law's 1980 texture transforms (combinations of gaussian derivative convolutions) show that R5R5 may be a good way to find regions of high spatial and intensity variability, that is high density changes.

E5 = [-1 -2 0 2 1] 1st deriv, sigma=1
R5 ripple = [1 -4 6 -4 1] 3rd deriv

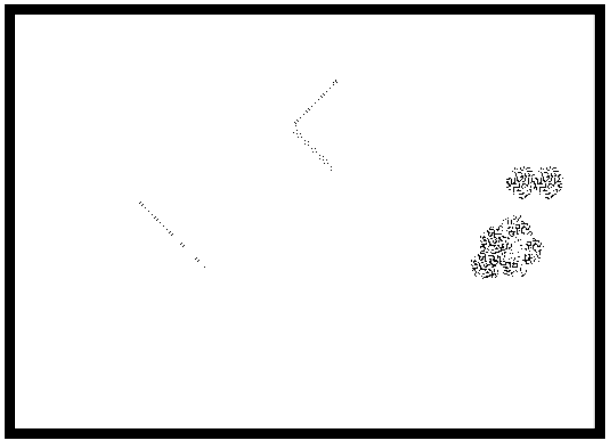
original image



E5E5 then adap med



R5R5 then adap med



Shape Finding in over-segmented images (cont)

Finding the object (gingerbread man) in another image where it has changed location, lighting and pose requires very local and grouped information because the number of possible true key point matches compared to the total number of key points in the image is small.

one approach in progress:

(1) filter key points and segmentation in the image to be searched by filters of difference from the template object's colors (HSV and CIE LAB usually)

(2) filter the segmentation in the image to be searched to remove labeled regions with a higher surface density of key points after the region statistics have been scaled using size arguments. (substitution for a texture filter)

(3) for each labeled region, calculate euclidean transformation using point pair combinations and evaluate the pairs using all key points within a radius of template object size.

(4) using the octave combination and the labeled region's best transformation, include all the bounds of the matched key points' regions in a transformed nearest neighbor match within tolerance

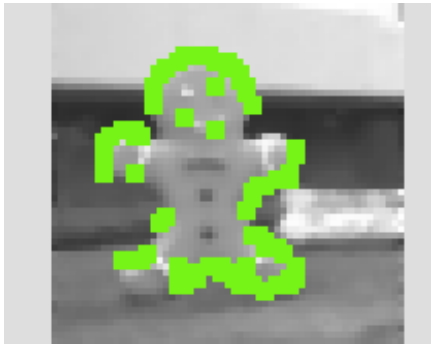
Shape Finding in over-segmented images (cont)

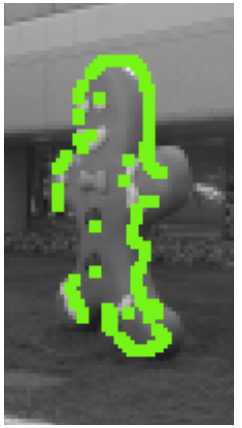
(5) for the octave and labeled region best result, used the matched boundary correspondence to calc chord difference costs within tolerance
(6) determined the cost of those “descriptors”, that is key point hsv costs, euclidean transformation distance costs, and chord difference costs to find the best results, then those within a percentage of best if any.

The best result at this point for one test is not yet the true match, but the 2nd best result is the true match.

Further information is needed to determine that the true match is better.

best match **and** **true match**

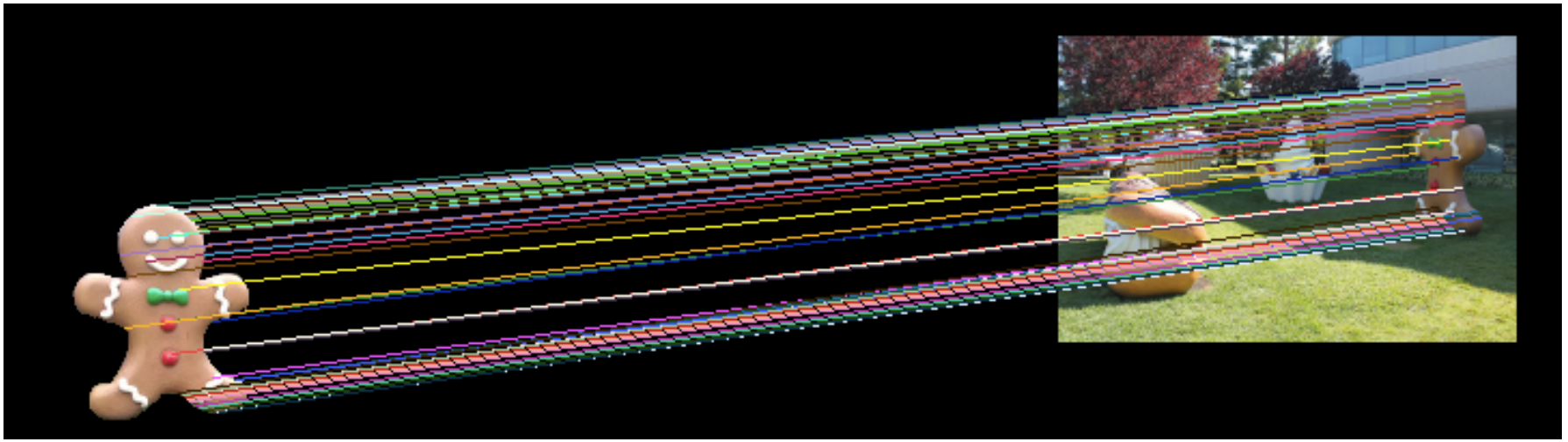




Shape Finding in over-segmented images (cont)

(7) using the combined labeled region as contiguous small hsv descriptors to calculate the pixel by pixel similarity between transformed frame 2 and frame`1 pixels.

true match is found, at least for the test w/ largest projection effects



but alot of time is spent in pairwise euclidean transformations and evaluations

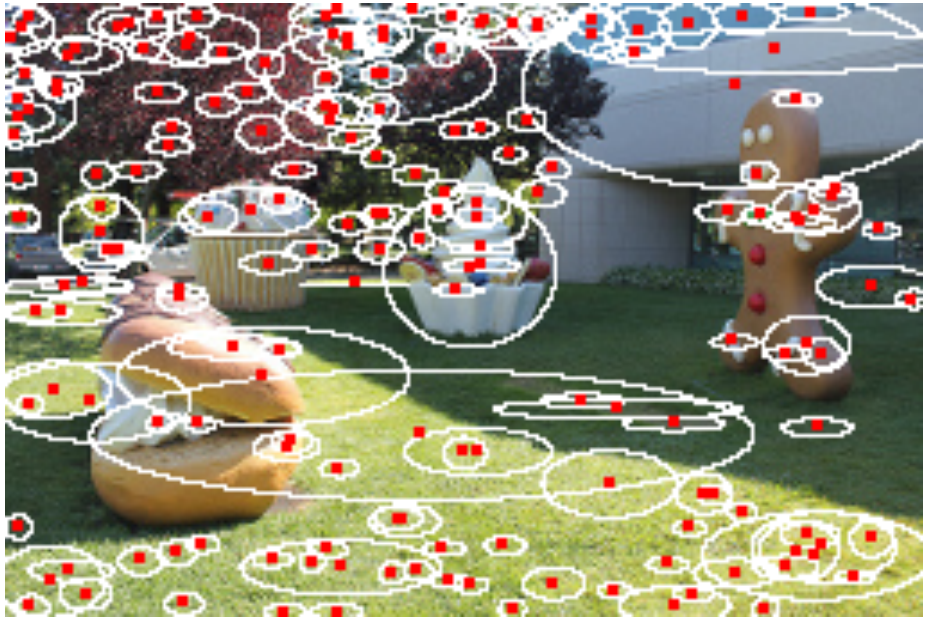
Maximally Stable Extremal Regions (MSER)

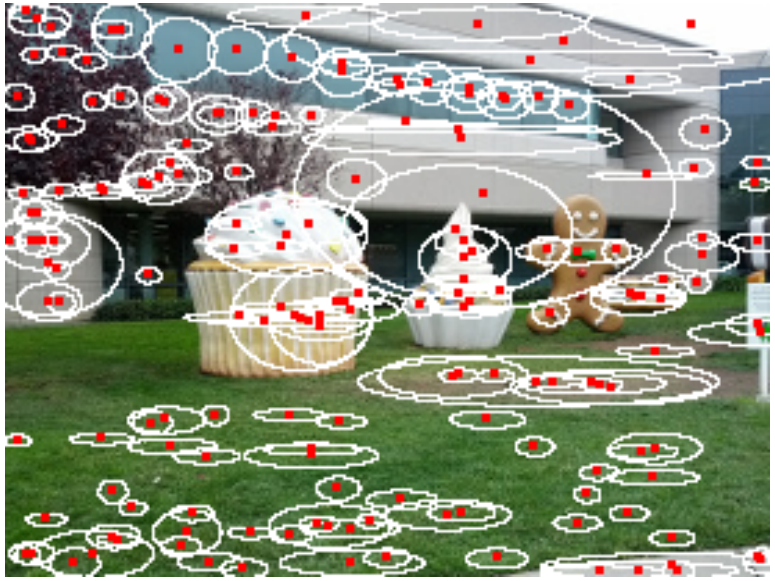
blob detection with MSER followed by canonicalization and descriptor matching is in progress.

plotted below and on the next page are the MSER regions found on the inverted images. The other sets of regions derived from the images are not plotted here.

especially useful for corner-less objects

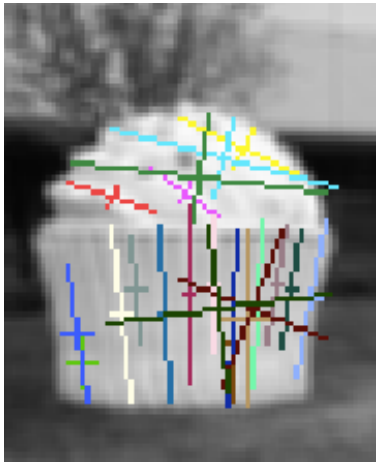
MSER (cont.)

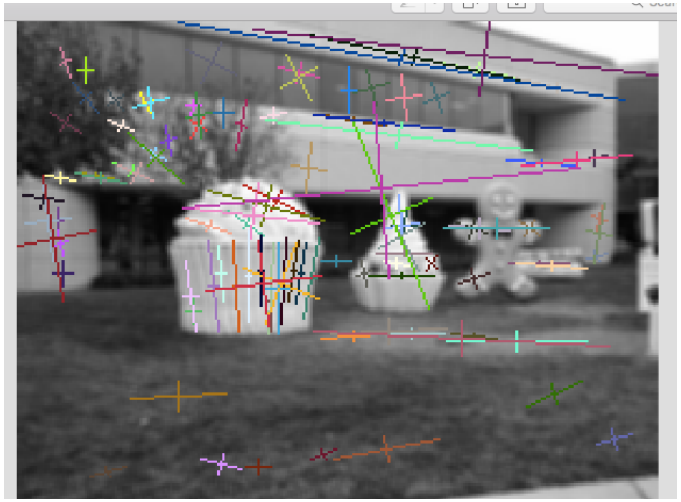


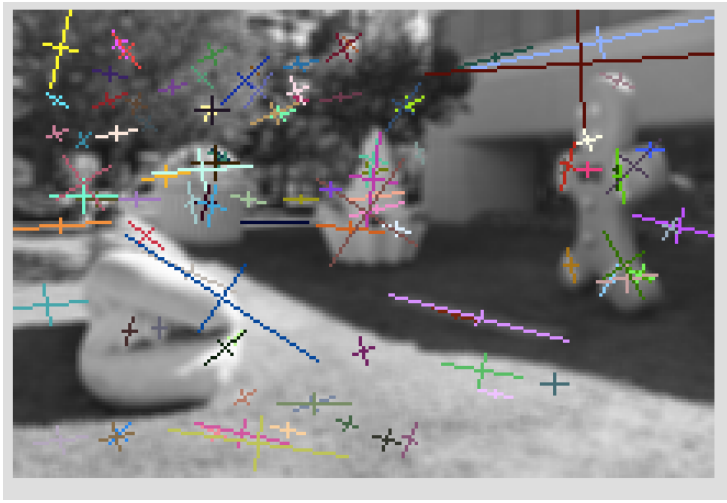


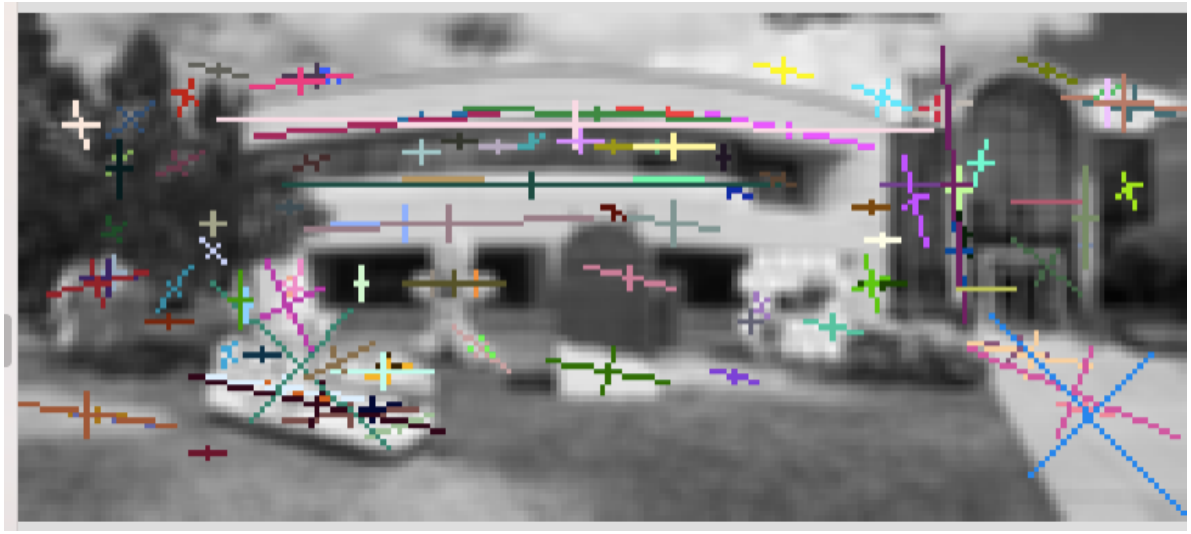
MSER (cont.)

for the cupcake, there is extreme repetition of a pattern that if present in the searchable image, would help to further filter matchable mser regions.

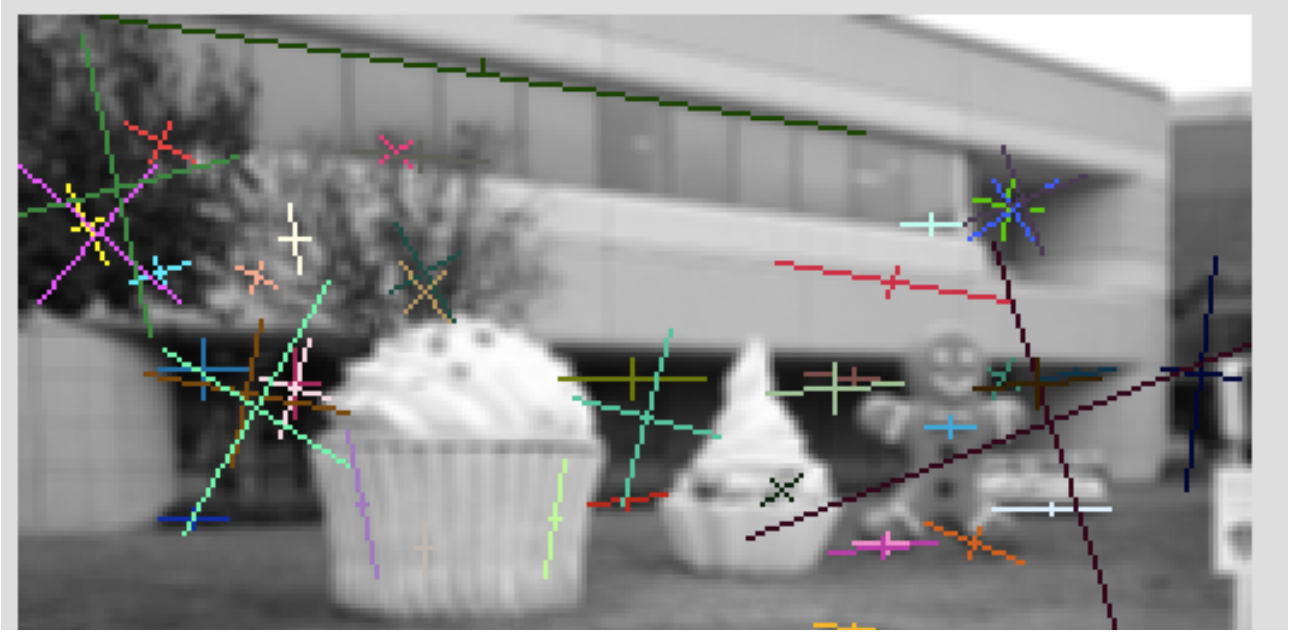


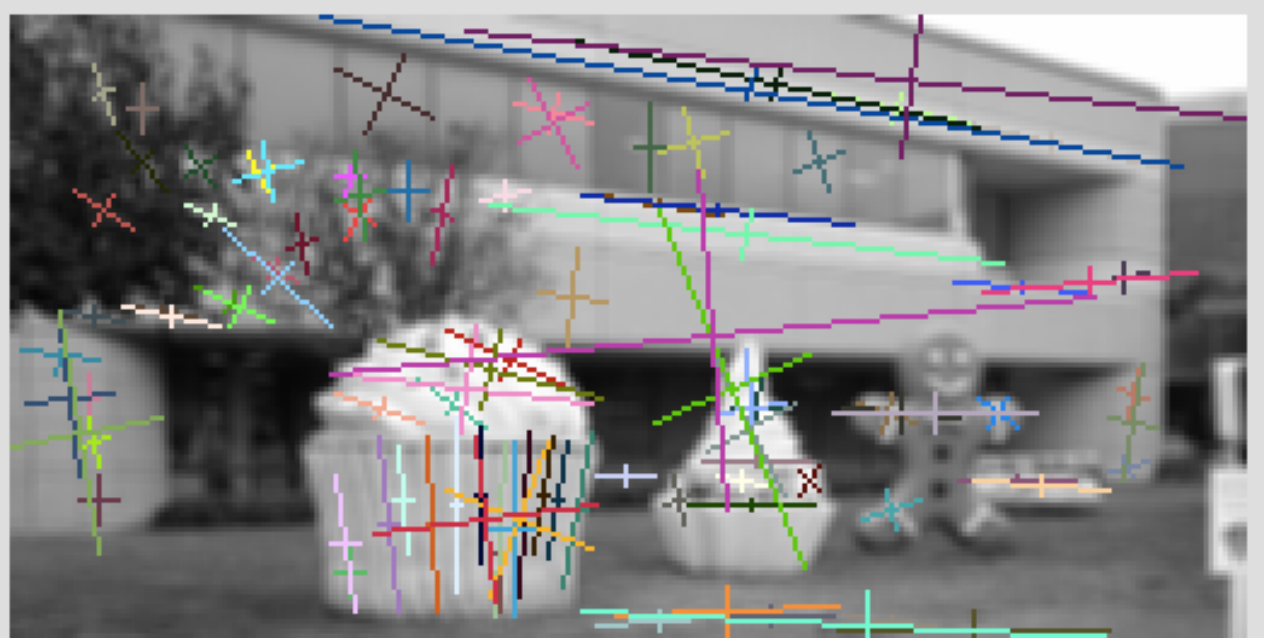




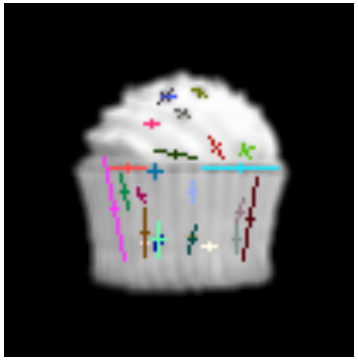


MSER (cont.)



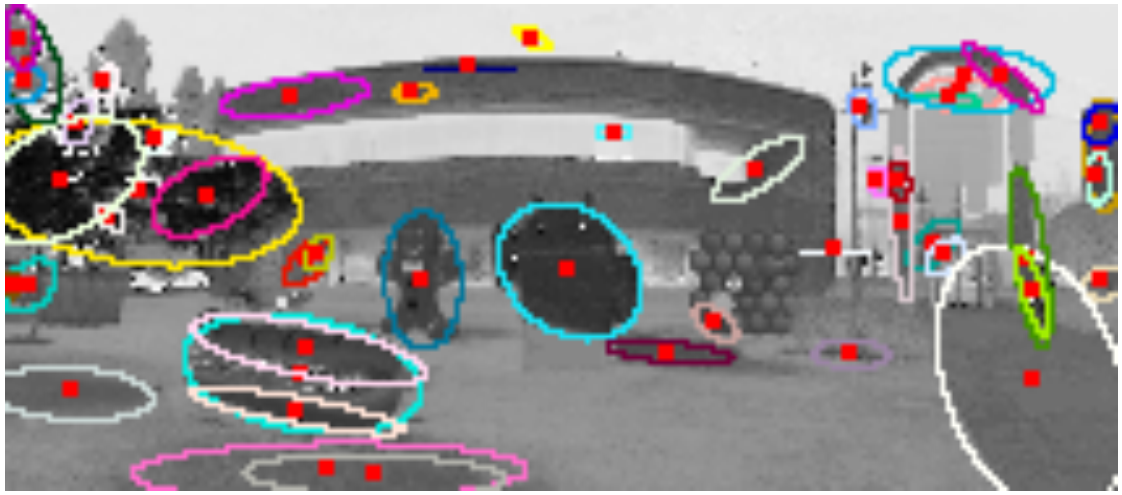


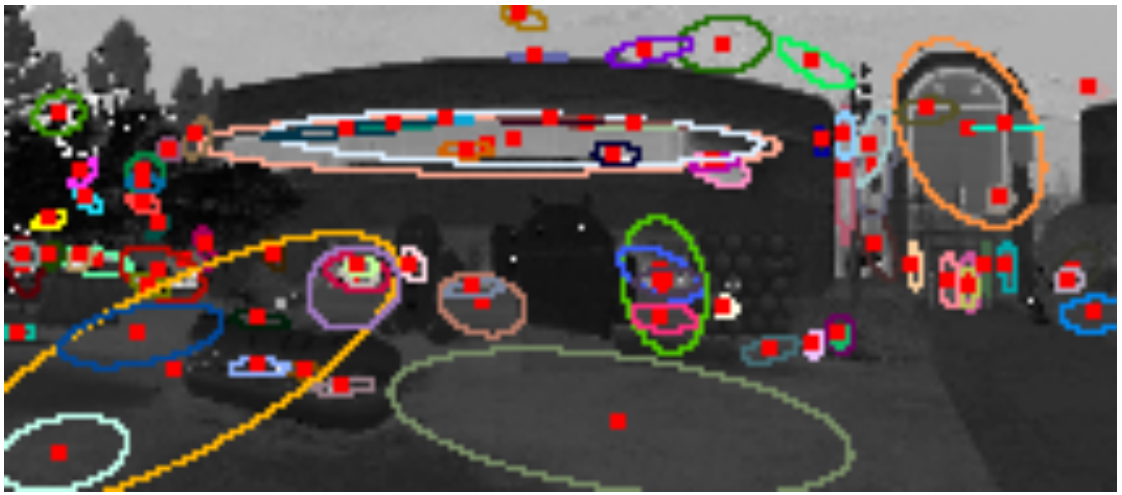
By the way, the MSER semi-major axes possibly are very fast ways to find lines in an image. The semi-major axes of these blobs are the directions of slowest change in 2nd moment of intensity. **The straightness of lines are properties preserved in projective transformations.**



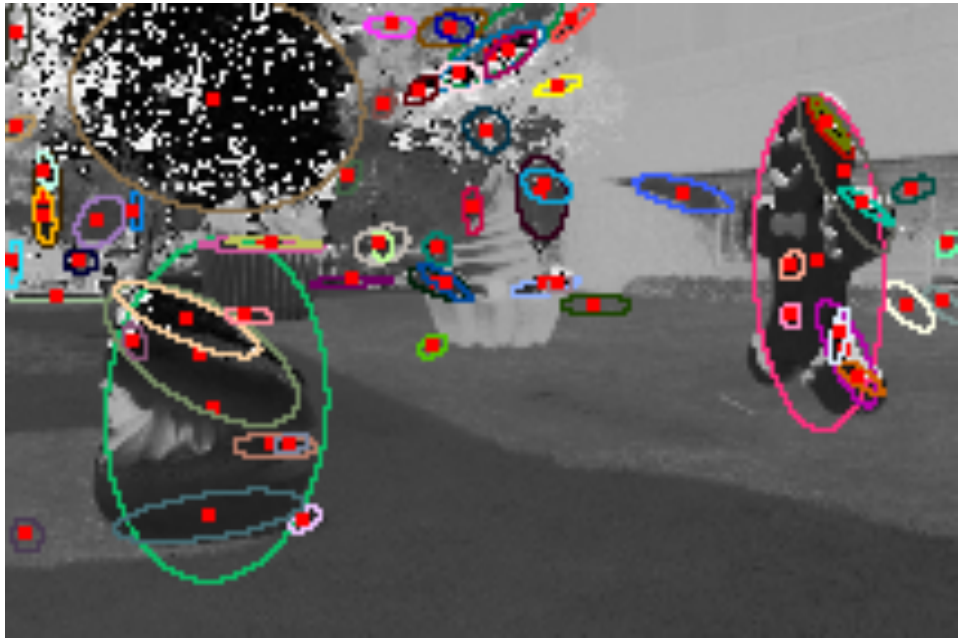
MSER (cont.)

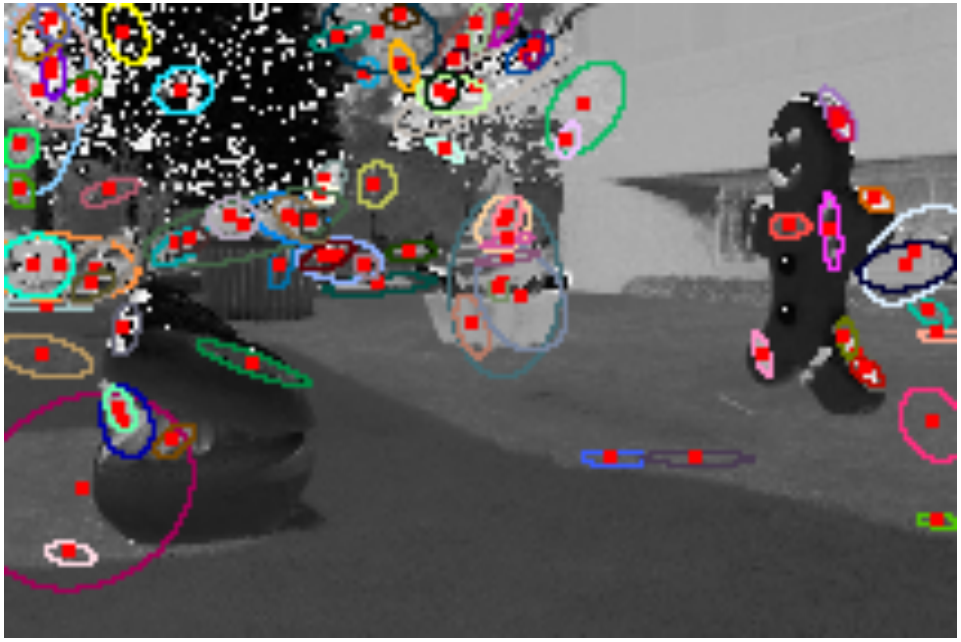
further looking at MSER applied to CIELUV polar theta images.
The MSER blob detector is finding objects and detailed regions. *(blobs from both the positive and inverted images are useful).*



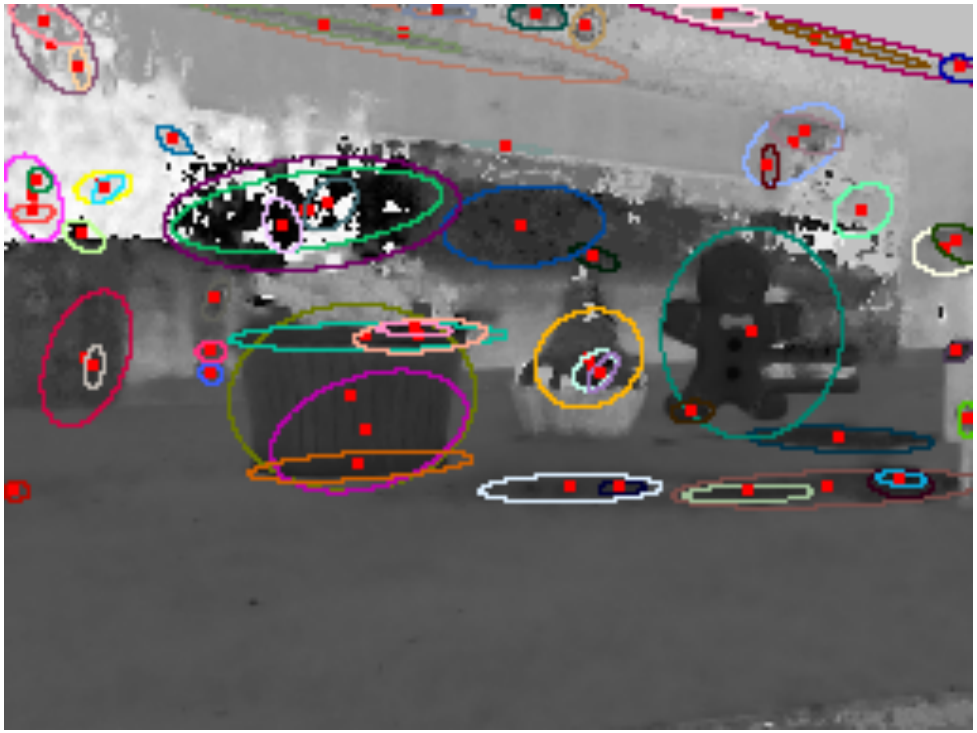


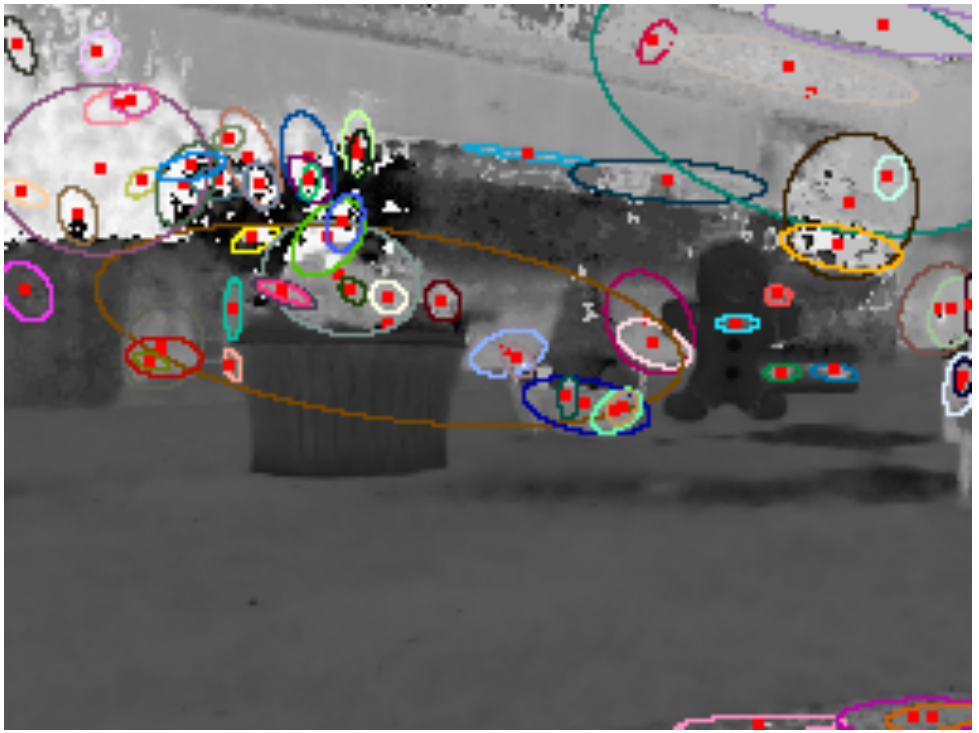
MSER (cont.)



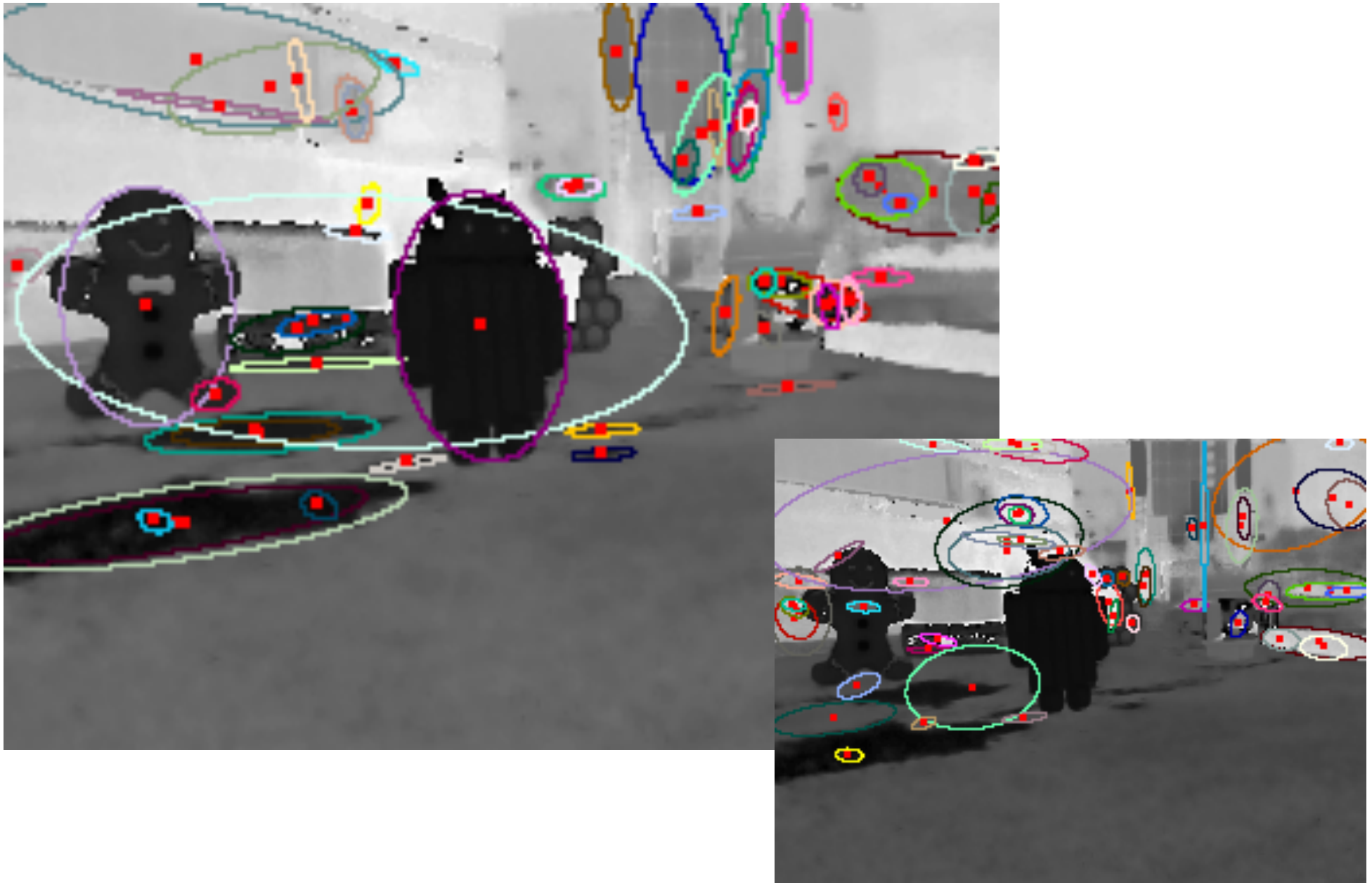


MSER (cont.)





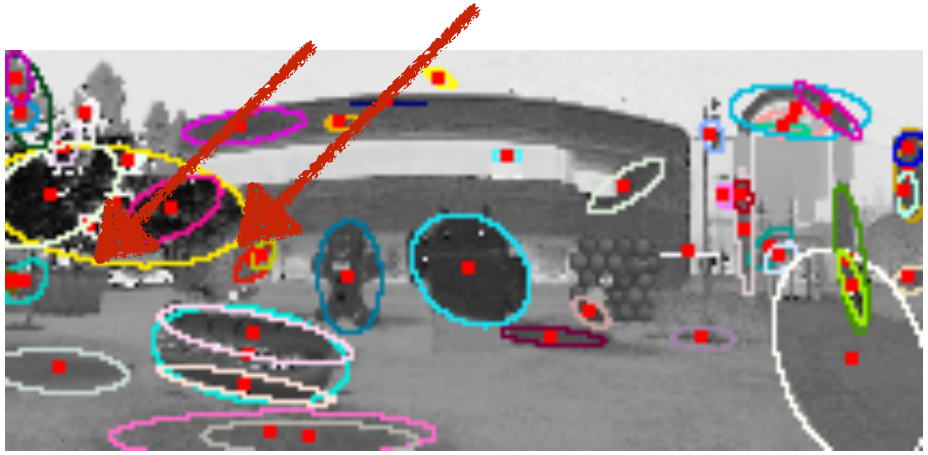
MSER (cont.)

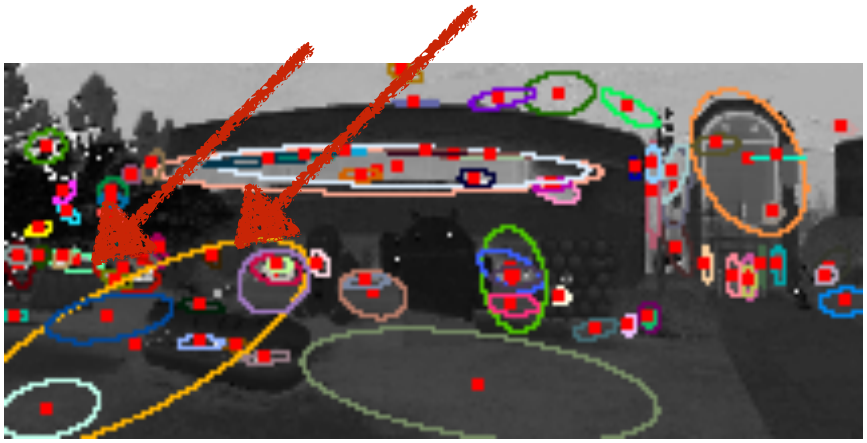


MSER (cont.)

Using MSER on CIELUV polar theta images succeeds for most of the objects, but *fails for the lower resolution* android_statues_01.jpg in finding the *ice-cream and the cupcake*.

For those 2 objects, the intensity contrast is needed (see the previous MSER results).





MSER (cont.)

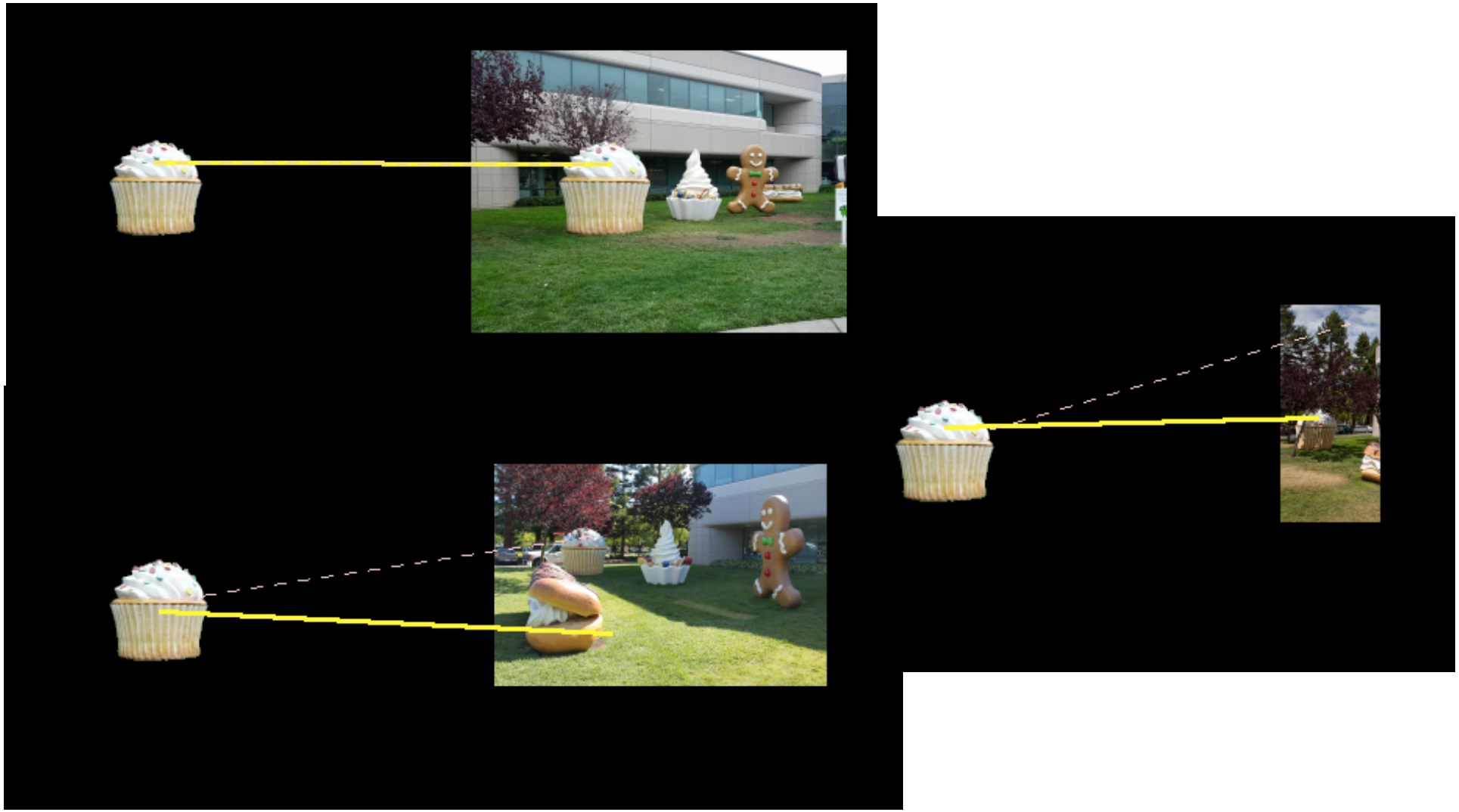
Finding the object (gingerbread man) in another image where it has changed location, lighting and pose, requires very local and grouped information because the number of possible true key point matches compared to the total number of key points in the image is small.

the general approach:

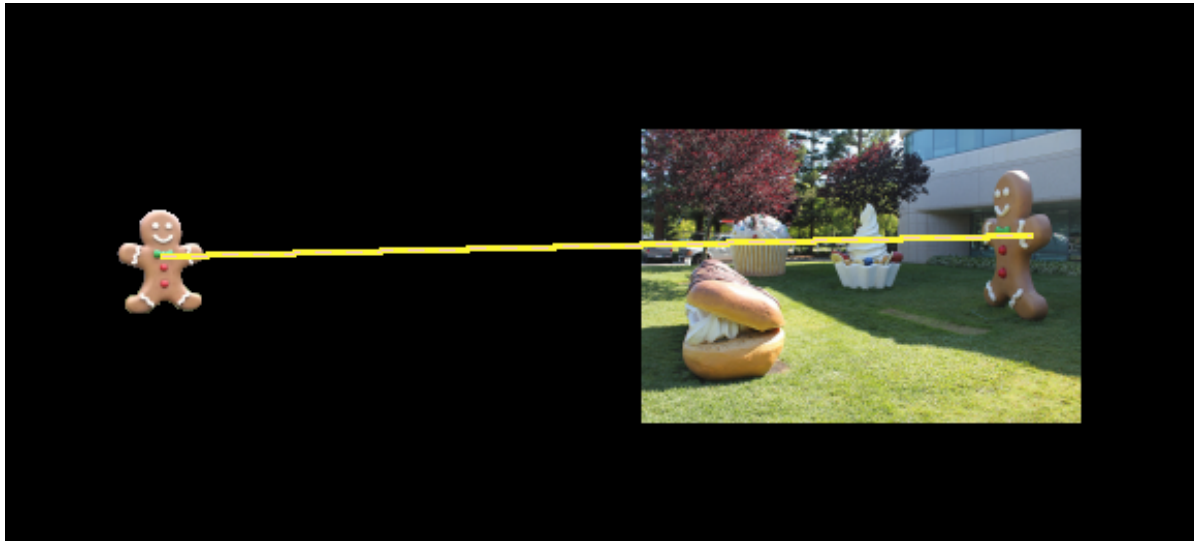
(1) Build an RGB pyramid for both images and calculate MSER regions for the largest image in both pyramids.
(2) Build a Polar theta CIELUV pyramid for both images and calculate MSER regions for the largest image in both pyramids.
(3) filter the regions by color
(4) for each octave, find the best match of dataset0 objects (== template) to dataset1 objects and collect the best for each octave.
(5) Sort the best per octave by cost.

MSER (cont.)

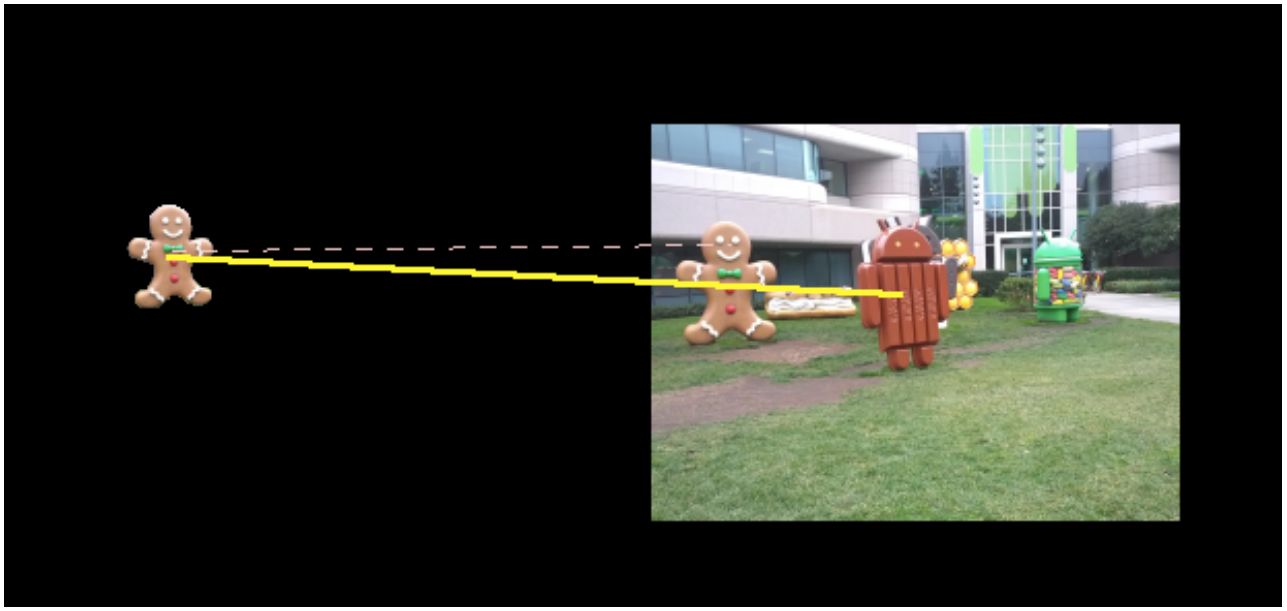
the android statue was found as the 1st or 2nd best match in all 10 tests.
The last one needs shape or other information to further distinguish it.

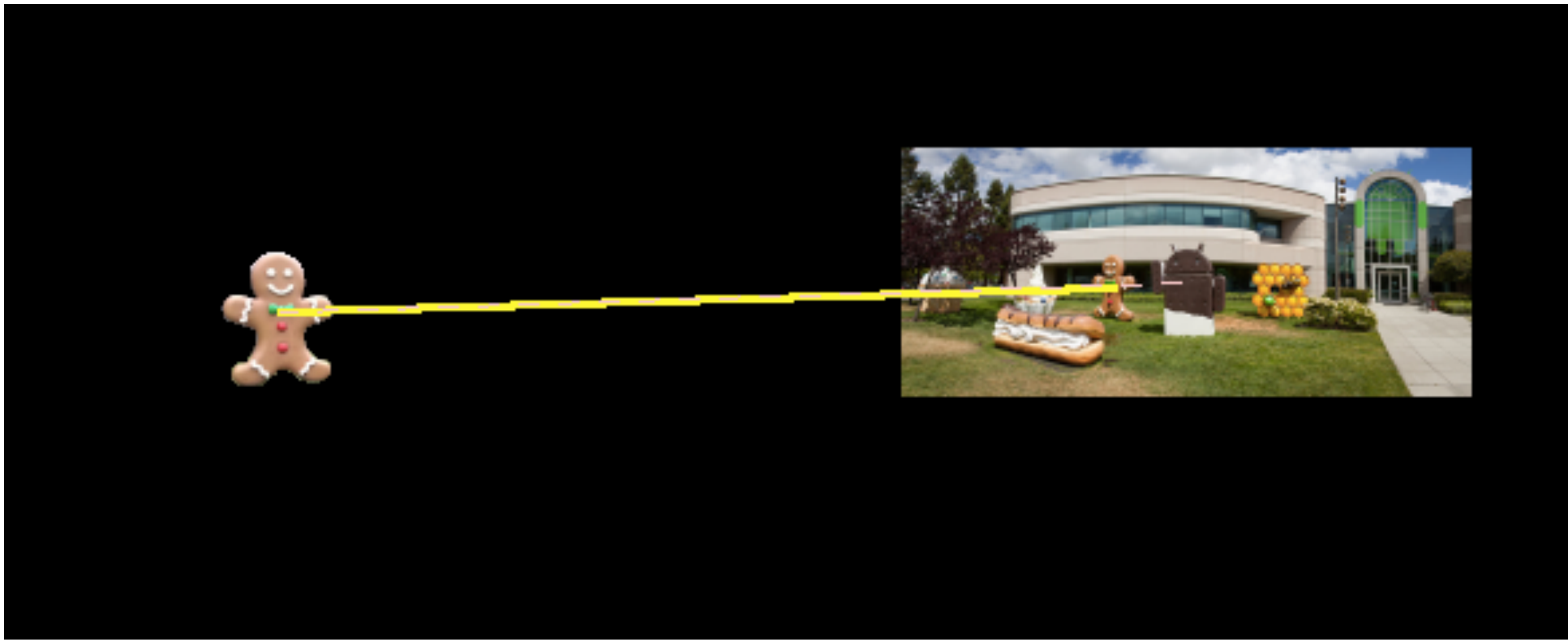


MSER (cont.)

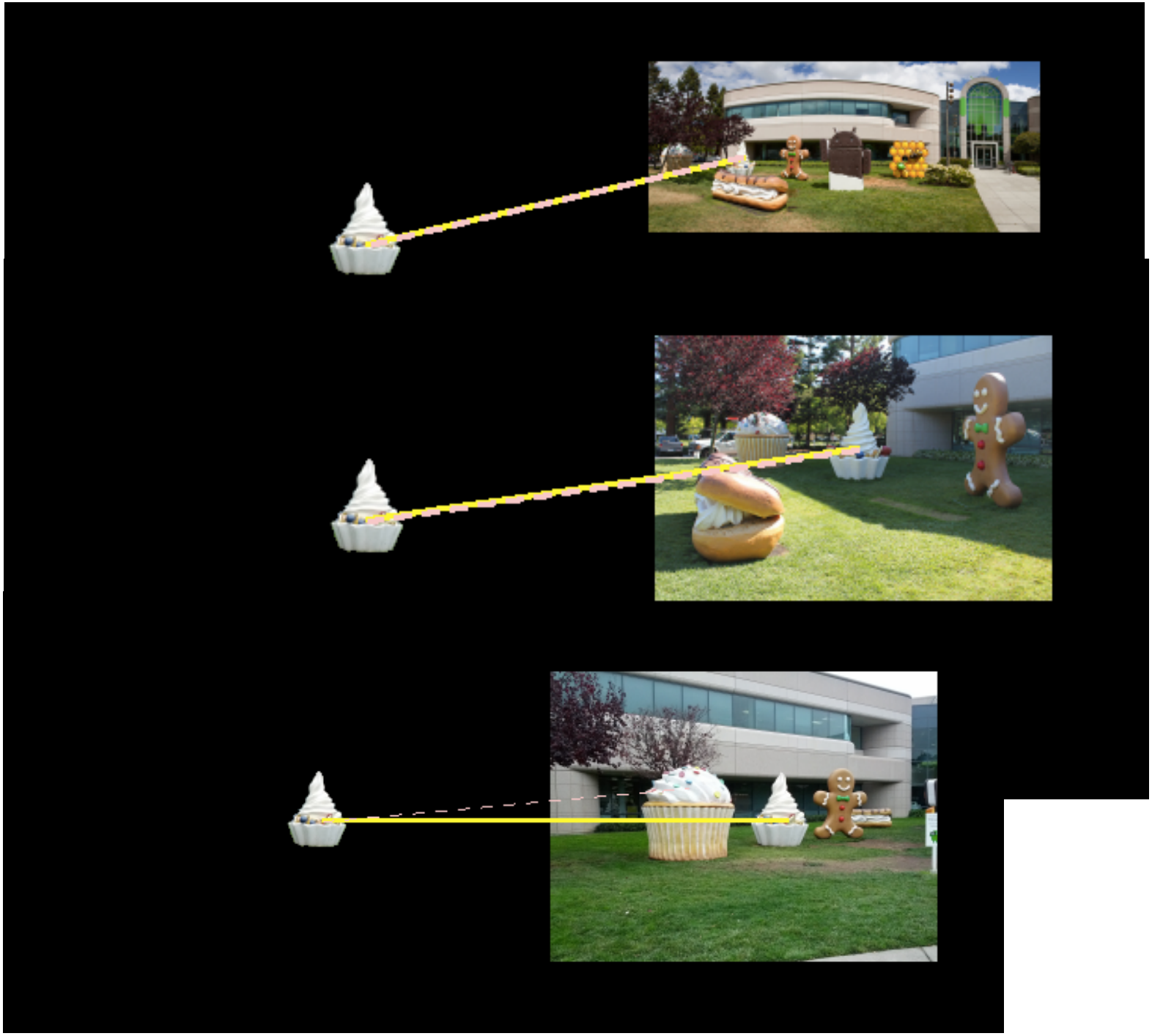


MSER (cont.)

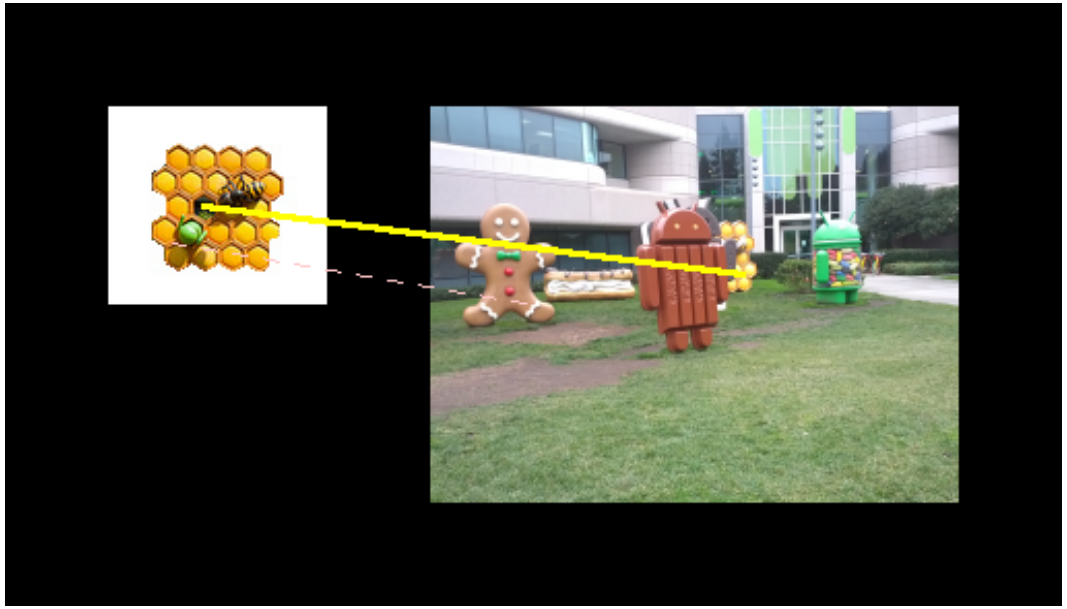




MSER (cont.)



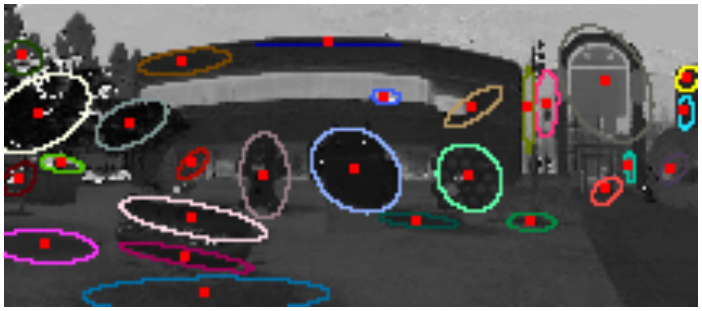
MSER (cont.)



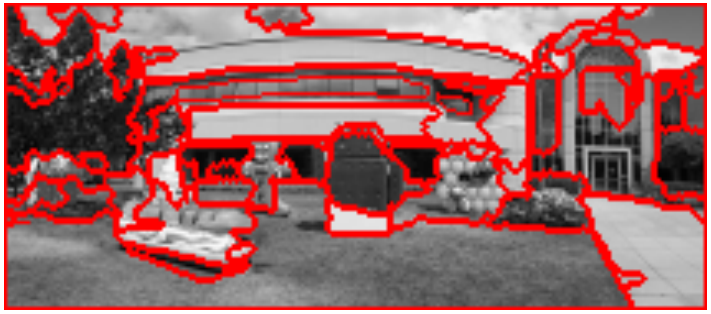
MSEREdges

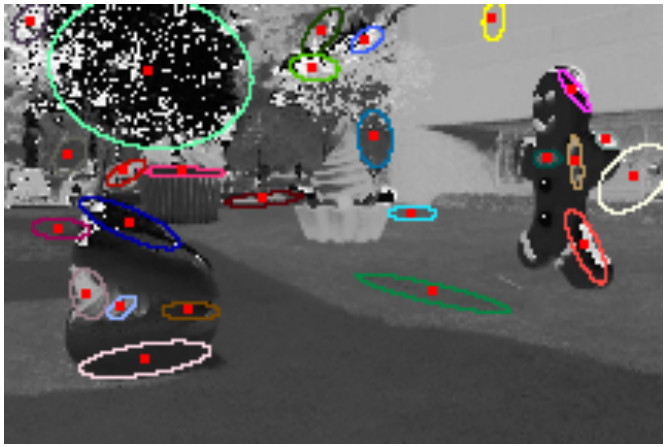
created a class called MSEREdges to make edges with the MSER regions created from greyscale and polar theta CIE LUV images.
then a moderate amount of merging using color limits.
(Note: one goal, not present in canny edges, was to complete the contours enabling better object definition for HOGs and inflection maps, and better horizon locators in landscape images, etc.)

objects found w/ MSER



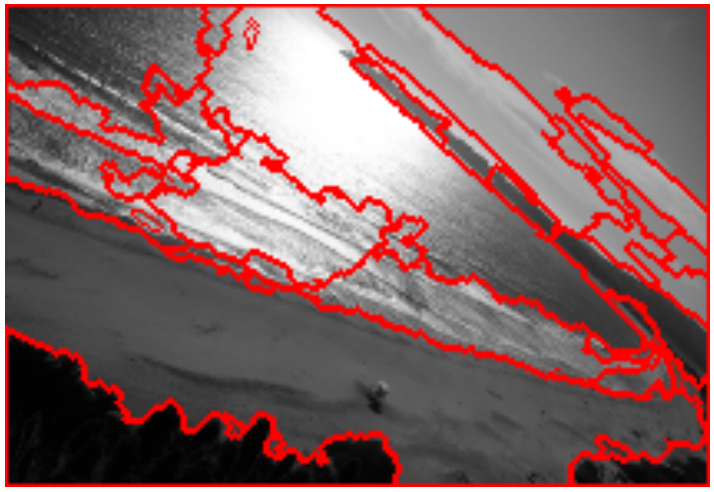
then closed edges (contours)



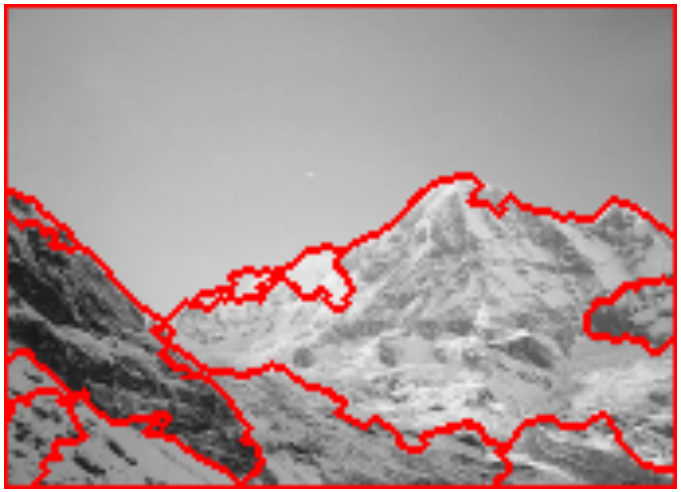


MSEREdges (cont.)

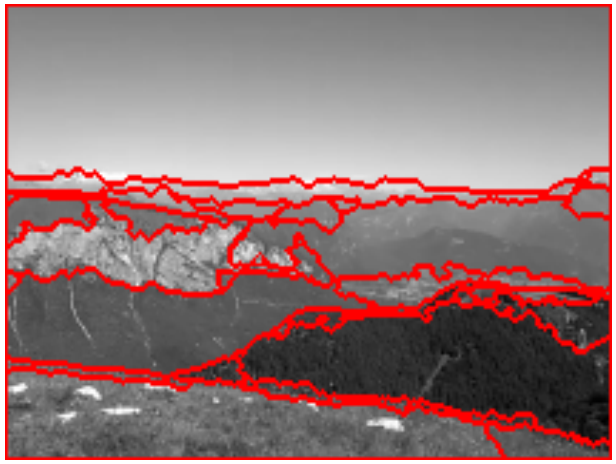
MSEREdges (cont.)

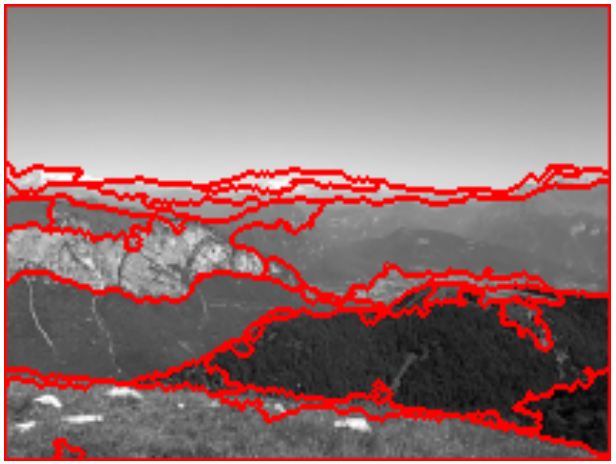


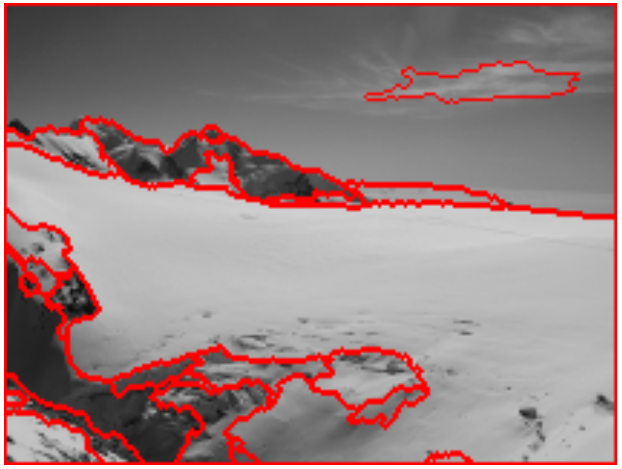
MSEREdges (cont.)

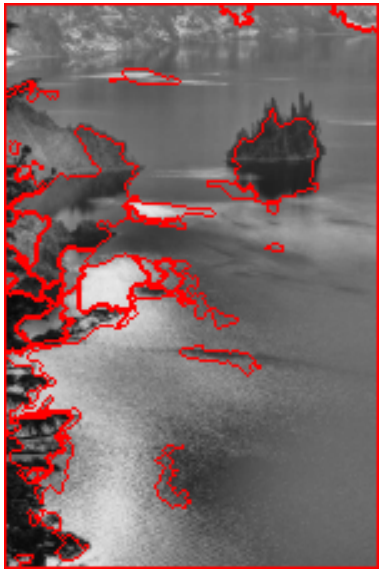


MSEREdges (cont.)

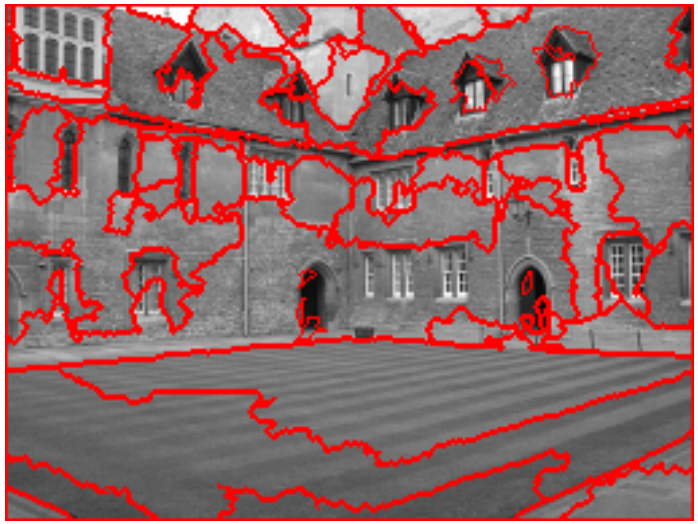




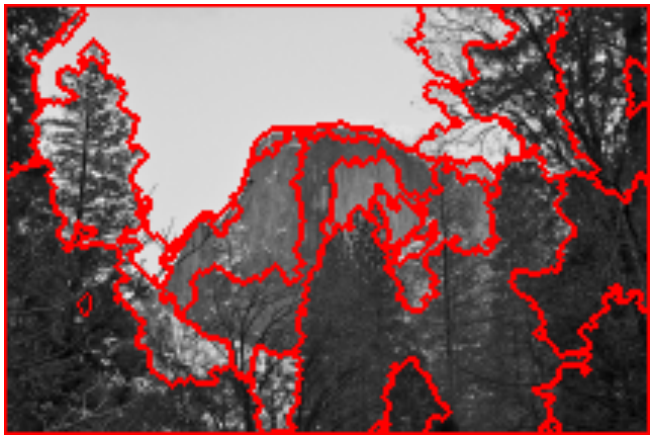




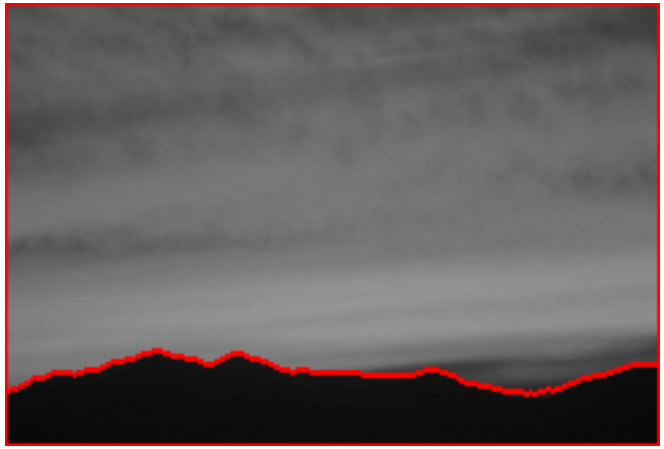
MSEREdges (cont.)

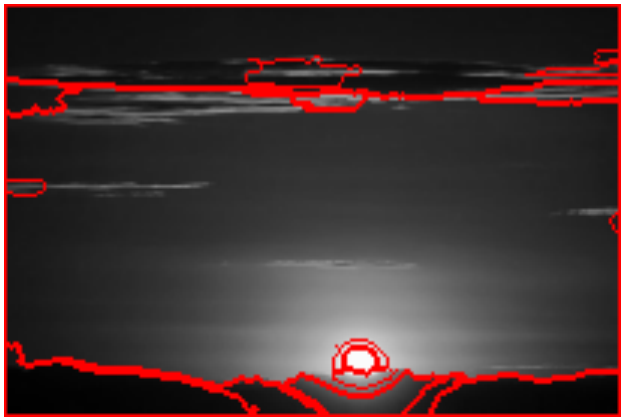


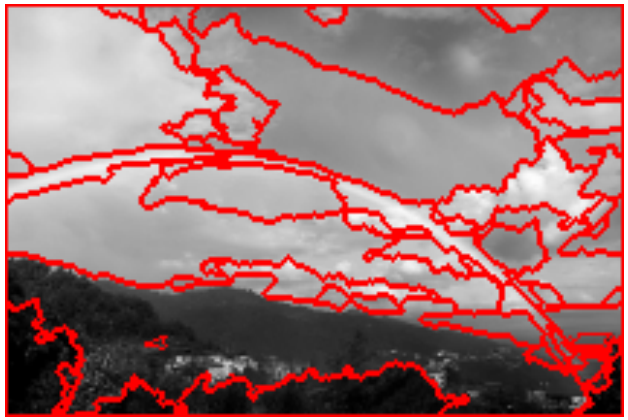
MSEREdges (cont.)

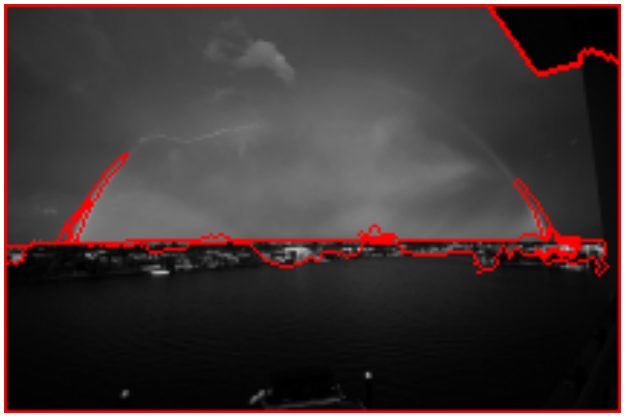


MSEREdges (cont.)

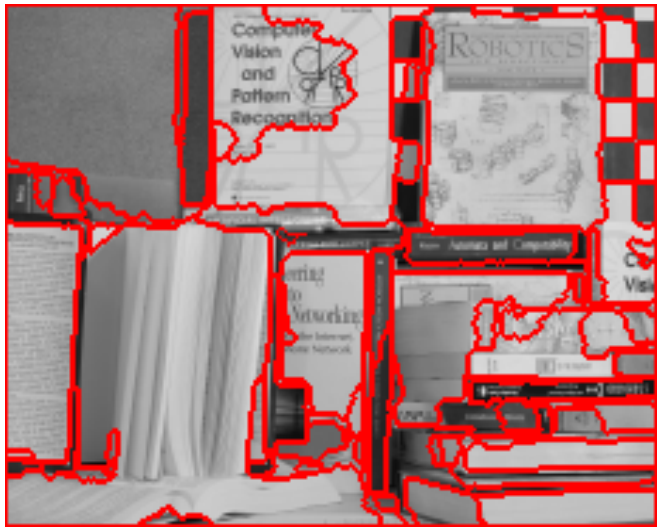


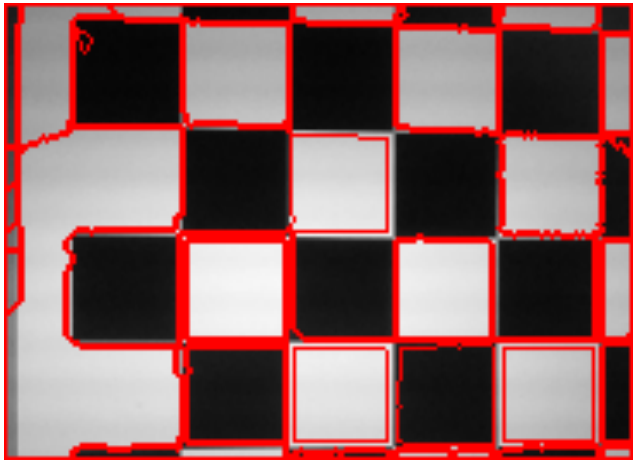


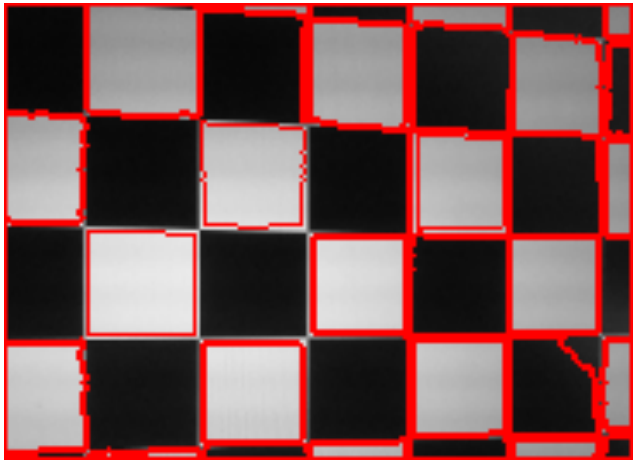




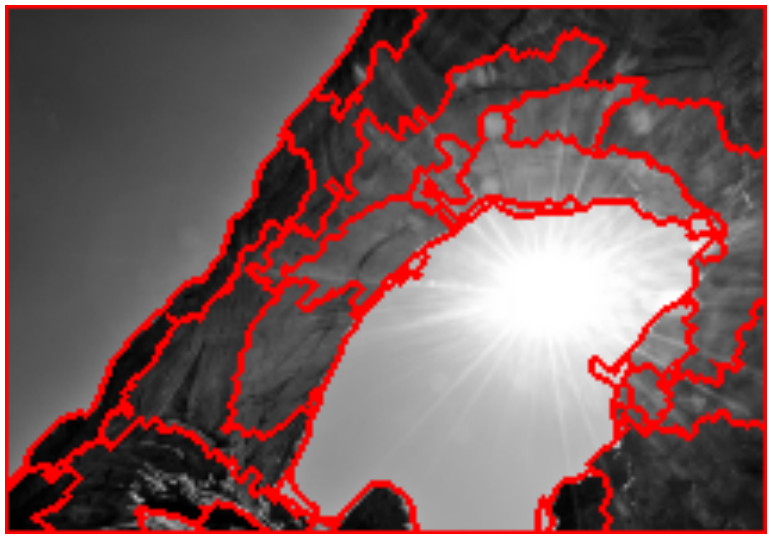
MSEREdges (cont.)

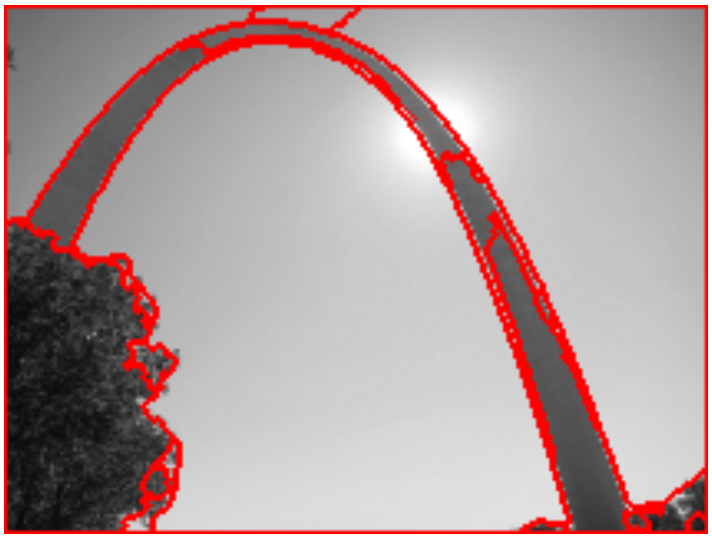


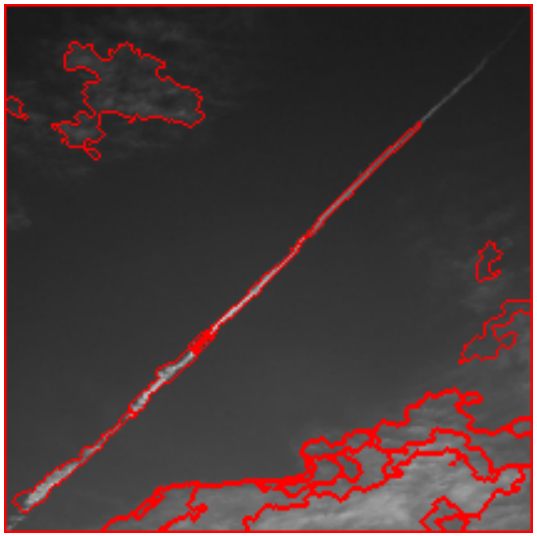




MSEREdges (cont.)





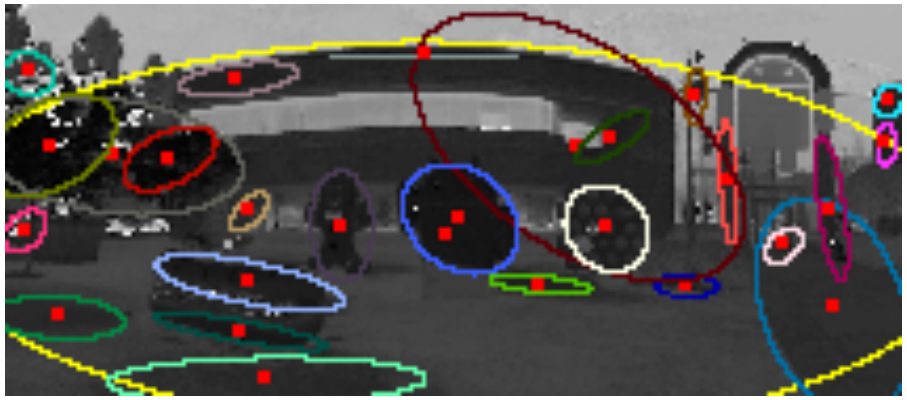


MSEREdges (cont.)

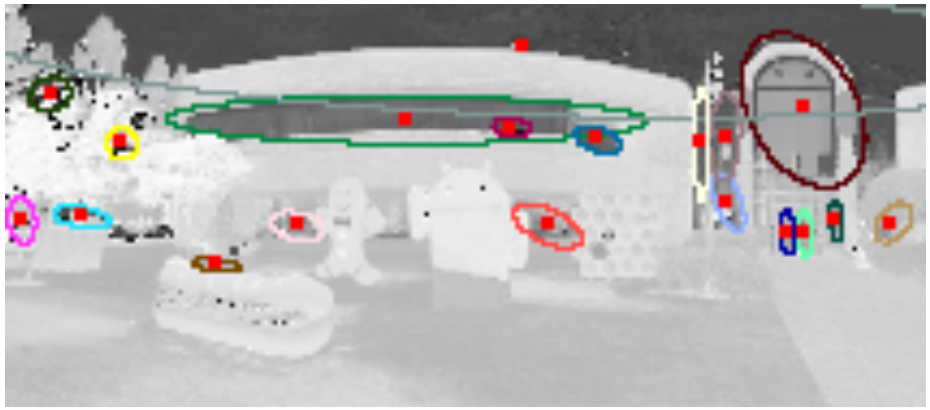
MSEREdges (cont.)

correcting the polar theta image regions for wrap around effects (360^{o} to 0^{o})

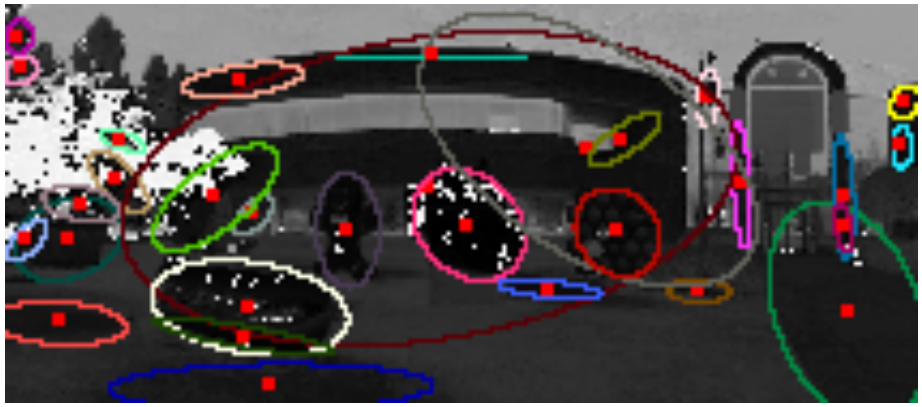
PT0



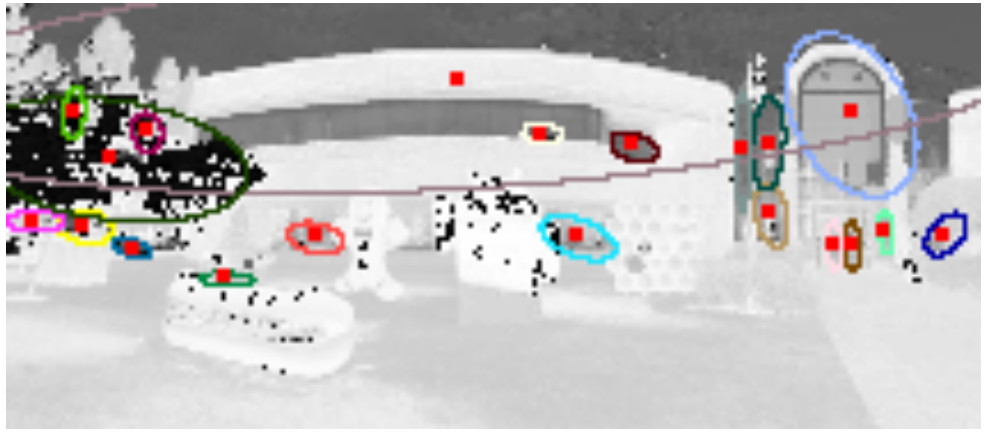
PT1



PT0 shifted by
20 degrees



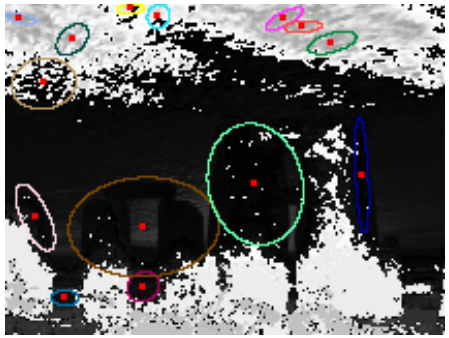
PT1 shifted by
20 degrees



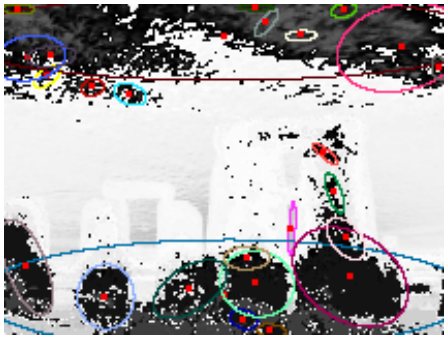
MSEREdges (cont.)

correcting the polar theta image regions for wrap around effects

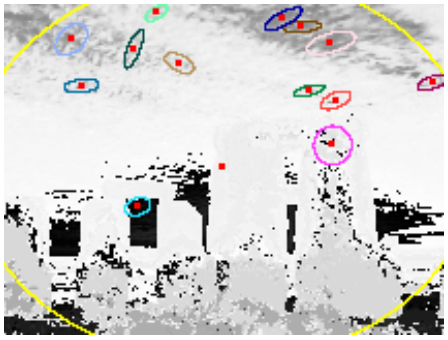
PT0



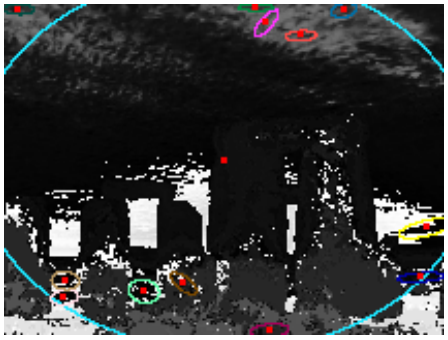
PT1



PT0 shifted by
20 degrees



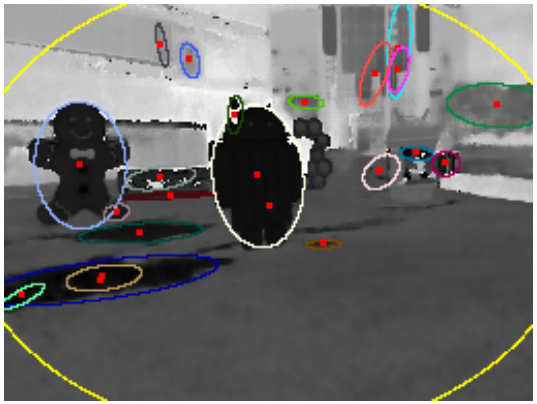
PT1 shifted by
20 degrees



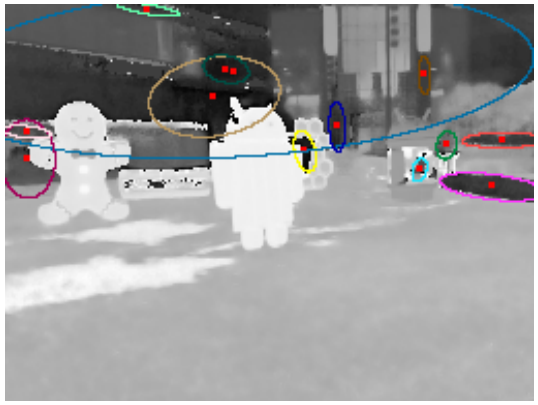
MSEREdges (cont.)

correcting the polar theta image regions for wrap around effects

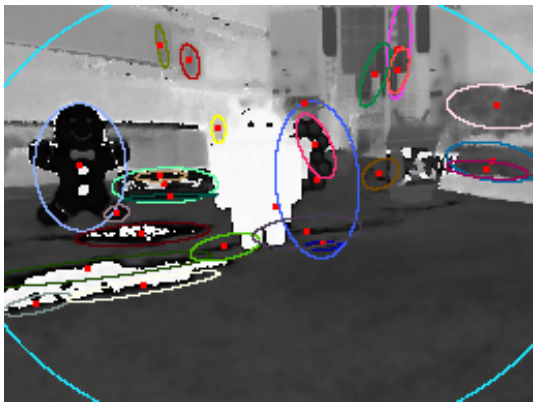
PT0



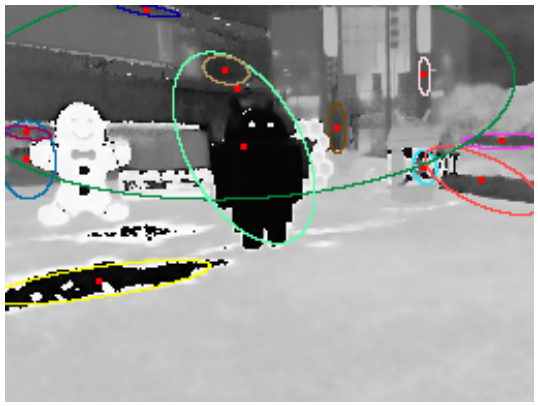
PT1



PT0 shifted by
20 degrees



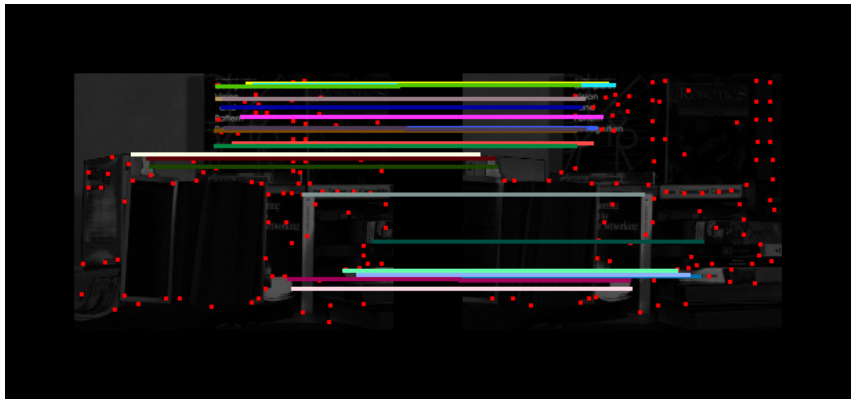
PT1 shifted by
20 degrees



ORB and other key point descriptors

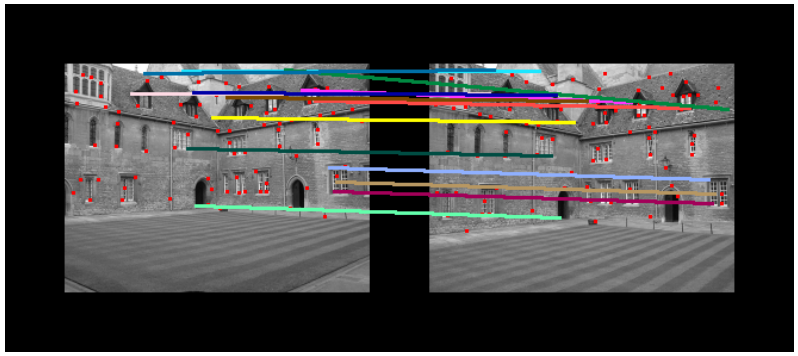
back to the sparse matching approach possible when there are more points that are possible true matches than the number of points that cannot be (most of image is same as other image).

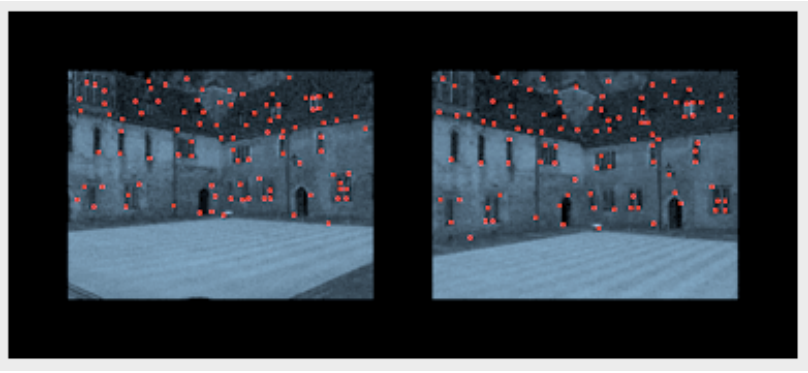
For the test image of books, created descriptors for greyscale (left) and “C” of LCH colorspace (right).
And below that the combined descriptors which might be necessary for some images (e.g. greyscale not distinguishing some color differences).

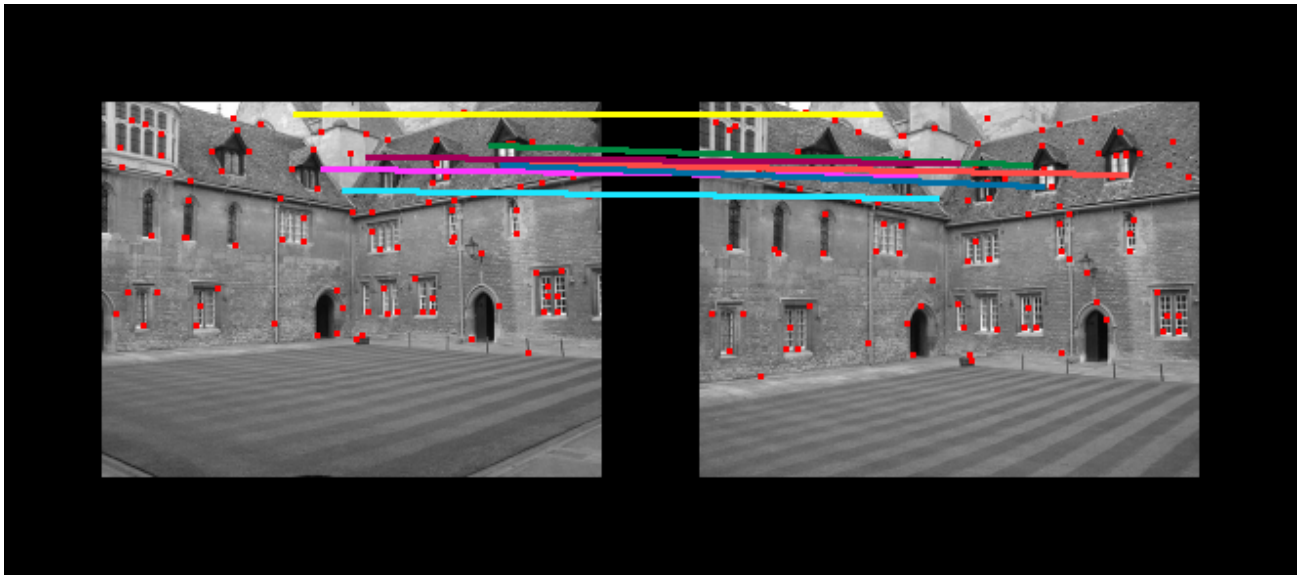


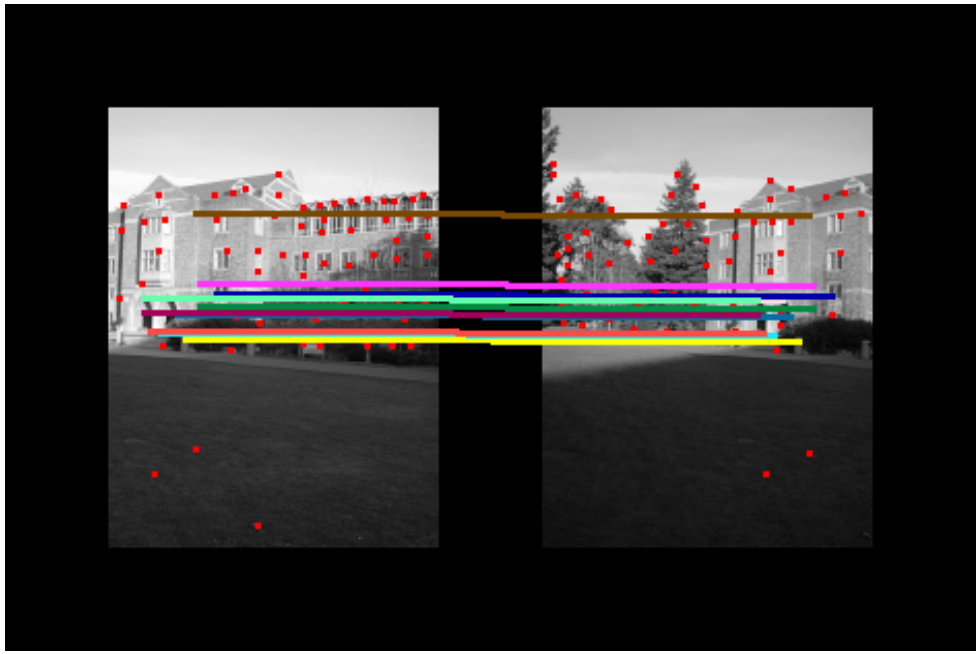
ORB and other key point descriptors (cont.)

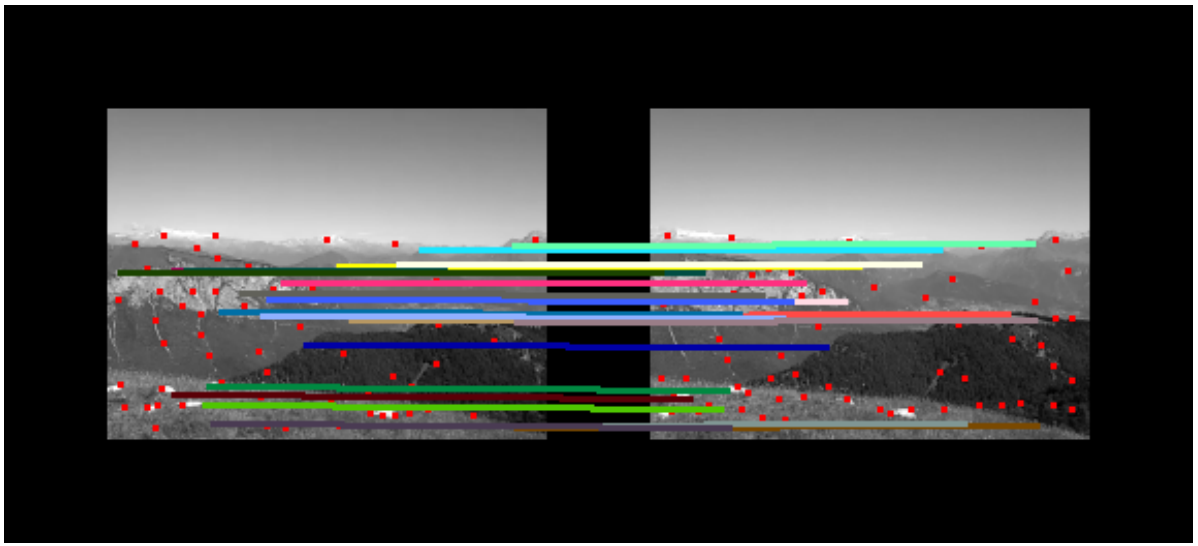
the test on books images used a greedy matcher that excludes points with a close second best match.
Can see that the Merton College test images need a different matching technique.











Sky and objects extracted using MSER and color

Sky and objects extracted using MSER and color










Sky and objects extracted using MSER and color















Optical Flow, Motion Boundary and Segmentation for a motion sequence (not implemented here)

from thesis **"The Early Detection of Motion Boundaries"** by Spoerri, 1990/1991
https://www.researchgate.net/publication/36043121_The_Early_Detection_of_Motion_Boundaries
https://dspace.mit.edu/handle/1721.1/6814

The reader may perceive the outline of a dalmation on its morning walk, but most likely she or he will experience some difficulty because of the absence of distinct intensity edges along the dalmation's outline.

If, however, the reader were to see a motion sequence of the dalmation then she or he would immediately perceive its outline, even though the intensity information in the individual frames is ambiguous.

Regarding aperture problem when determining optical flow:

from "Directional selectivity and its use in early visual processing"
D Marr, S Ullman - Proceedings of the Royal Society of …, 1981

ABSTRACT: ~ 1he construction of directionally selective untt~ind kheir use in the processing of visual motion, are considered. The zero-crossings of ~~~~~~ • I(x,y) are located, as In Mart & Hfldreth (1979]. In addition , the time derivative d/dt (G(x,y) • l(x,y)] Is measured at the zero-crossings, and serves to constraint the local direction of motion to within 180 degrees. The direction of motion is determined in a second stage by combining the local constraints. The second part of the paper suggests a specific model of the information processing carried out by the X and Y cells of the retina and the LON, and certain classes of cortical simple cells. A number of psychophysical and neurophysiological predictions are derived from the theory

The use of zero—crossing segments as primitives for motion raises a substantial difficulty which we shall call the aperture problem (see figure 1). If the motion is to be detected by a unit that is small — compared with the overall contour, the only information one can extract is the component of the motion perpendicular to the local orientation. Motion along the contour will be invisible. Hence local measurements alone fail to give either the direction or speed of movement, and can only restrict the direction to within 180^0. In other words, only the sign of the movement is given directly by the local measurement

Optical Flow, Motion Boundary and Segmentation for a motion sequence (not implemented here)

from **"Detecting Pedestrians Using Patterns of Motion and Appearance"** by Paul Viola, Michael Jones, and Daniel Snow 2005, Proceedings of the International Conference on Computer Vision, Nice, France
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.10.4061&rep=rep1&type=pdf

Motion filters operate on 5 images:

$$\Delta = abs(I_t - I_{t+1})$$
$$U = abs(I_t - I_{t+1}\uparrow)$$
$$L = abs(I_t - I_{t+1}\leftarrow)$$
$$R = abs(I_t - I_{t+1}\rightarrow)$$
$$D = abs(I_t - I_{t+1}\downarrow)$$

where I_t and I_{t+1} are images in time, and $\{\uparrow, \downarrow, \leftarrow, \rightarrow\}$ are image shift operators ($I_t\uparrow$ is I_t shifted up by one pixel). See Figure 2 for an example of these images.

One type of filter compares sums of absolute differences be $f_i = r_i(\Delta) - r_i(S)$ U, L, R, D}

where S is one of {U, L, R, D} and $r_i()$ is a single box rectangular sum within the detection window.

The second type of filter compares sums within the same motion image

$$f_j = \phi_j(S)$$

where ϕ_j is one of the rectangle filters shown in Figure 1.

The third type of filter measures the magnitude of motion in one of the motion images:

$$f_k = r_k(S)$$

where S is one of {U, L, R, D} and $r_k()$ is a single box rectangular sum within the detection window.

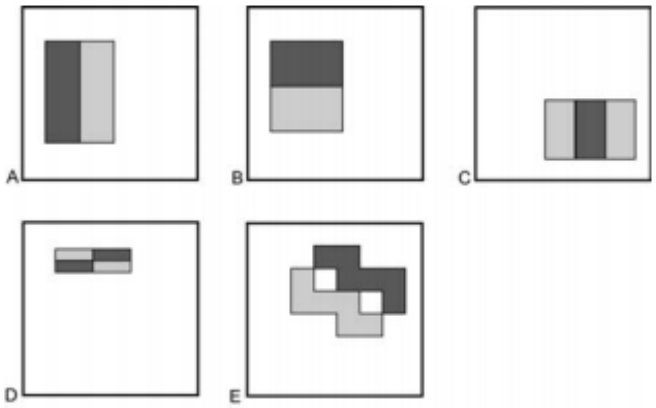


Figure 1: Example rectangle filters shown relative to the enclosing detection window. The sum of the pixels which lie within the lighter rectangles are subtracted from the sum of pixels in the darker rectangles. For three-rectangle filters the sum of pixels in the darker rectangle is multiplied by 2 to account for twice as many lighter pixels.

A classifier, C, is a thresholded sum of features:

$$C(I_t, I_{t+1}) = \begin{cases} 1 & \text{if } \sum_{i=1}^{N} F_i(I_t, \Delta, U, L, R, D) > \theta \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

A feature, F, is simply a thresholded filter that outputs one of two votes.

$$F_i(I_t, I_{t+1}) = \begin{cases} \alpha & \text{if } f_i(I_t, \Delta, U, L, R, D) > t_i \\ \beta & \text{otherwise} \end{cases} \quad (2)$$

where $t_i \in \mathcal{R}$ is a feature threshold and f_i is one of the motion or appearance filters defined above. The real-valued α and β are computed during AdaBoost learning (as is the filter, filter threshold t_i and classifier threshold θ).

In order to support detection at multiple scales, the image shift operators $\{\uparrow, \downarrow, \leftarrow, \rightarrow\}$ must be defined with respect to the detection scale. This ensures that measurements of motion velocity are made in a scale invariant way. Scale invariance is achieved during the training process simply by scaling the training images to a base resolution of 20 by 15 pixels. The scale invariance of the detection is achieved by operating on image pyramids. Initially the pyramids of I_t and I_{t+1} are computed. Pyramid representations of $\{\Delta, U, L, R, D\}$ are computed as follows:

$$\Delta^l = abs(I_t^l - I_{t+1}^l)$$
$$U^l = abs(I_t^l - I_{t+1}^l\uparrow)$$
$$L^l = abs(I_t^l - I_{t+1}^l\leftarrow)$$
$$R^l = abs(I_t^l - I_{t+1}^l\rightarrow)$$
$$D^l = abs(I_t^l - I_{t+1}^l\downarrow)$$

where X^l refers to the the l-th level of the pyramid. Classifiers (and features) which are learned on scaled 20x15 training images, operate on each level of the pyramid in a scale invariant fashion. In our experiments we used a scale factor of 0.8 to generate each successive layer of the pyramid and stopped when the image size was less than 20x15 pixels.

Optical Flow, Motion Boundary and Segmentation for a motion sequence (not implemented here)

from **"Human Detection using Oriented Histograms of Flow and Appearance"** by Dalal, Triggs, and Schmid 2010
https://hal.inria.fr/file/index/docid/548587/filename/DTS06.pdf

introduces descriptors that use differential flow to cancel out most of the effects of camera motion and HOG like oriented histogram voting to obtain robust coding. First note that the image flow induced by camera rotation (pan, tilt, roll) … is locally essentially translational because significant camera roll is rare…The remaining signal is due to either depth-induced motion parallax between the camera, subject and background, or to independent motion in the scene. Differentials of parallax flows are concentrated essentially at 3D depth boundaries, while those of independent motions are largest at motion boundaries. For human subjects, both types of boundaries coincide with limb and body edges, so flow differentials are good cues for the outline of a person. However we also expect internal dynamics such as relative limb motions to be quite discriminant for human motions and differentials taken within the subject's silhouette are needed to capture these. ***Thus, flow-based features can focus either on coding motion (and hence depth) boundaries, or on coding internal dynamics and relative displacements of the limbs.***

Notation: I^x , I^y denote images containing the x (horizontal) and y (vertical) components of optical flow, $I^w = (I^x , I^y)$ denote the 2D flow image (w = (x, y)), and $I^x{}_x$, $I^x{}_y$, $I^y{}_x$, $I^y{}_y$ denote the corresponding x- and y-derivative differential flow images. E.g., $I^x{}_y = d/dy\ I^x$ is the y-derivative of the x component of optical flow.

Motion Boundary Histograms: take local gradients *separately* of the flow components $\boldsymbol{I^x}$ and $\boldsymbol{I^y}$ to calculate gradient magnitudes and orientations, and make those into local orientation histograms similar to the standard gray scale HOG. As with standard gray scale HOG, it is best to take spatial derivatives at the smallest possible scale ([1, 0, –1] mask) without any form of smoothing.

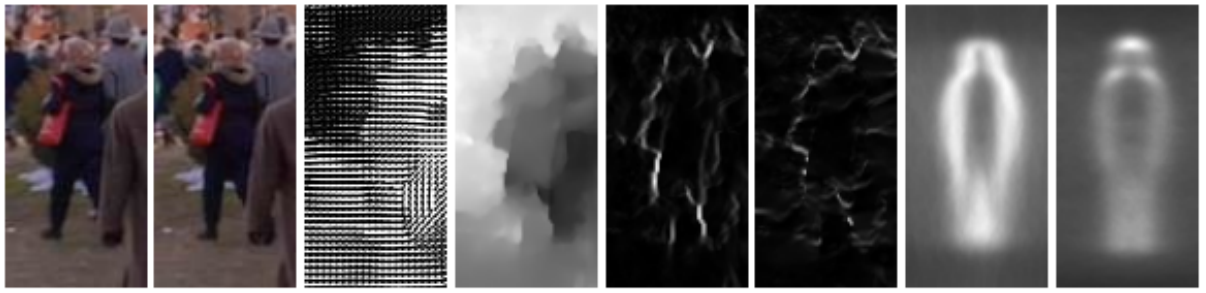


Fig. 3. Illustration of the MBH descriptor. (a,b) Reference images at time t and $t+1$. (c,d) Computed optical flow, and flow magnitude showing motion boundaries. (e,f) Gradient magnitude of flow field $\mathcal{I}^x, \mathcal{I}^y$ for image pair (a,b). (g,h) Average MBH descriptor over all training images for flow field $\mathcal{I}^x, \mathcal{I}^y$.

Optical Flow, Motion Boundary and Segmentation for a motion sequence (continued)

Internal/Relative Dynamics Based Coding: uses the pairs (I^x_x, I^y_x) and (I^x_y, I^y_y) to make ***Internal Motion Histograms (IMH)***

IMHdiff uses (I^x_x, I^y_x) and (I^x_y, I^y_y) to create two relative-flow-direction based oriented histograms. (NOTE: IMHdiff uses central differences and in the boundary cells it accesses pixels that lie outside the block.)

IMHcd uses the blocks-of-cells structure of the HOG descriptors differently. It uses 3×3 blocks of cells, in each of the 8 outer cells computing flow differences for each pixel relative to the corresponding pixel in the central cell and histogramming to give an orientation histogram1 . Figure 4(a) illustrates. The resulting 8 histograms are normalized as a block.

The motivation is that if the person's limb width is approximately the same as the cell size, IMHcd can capture relative displacements of the limbs w.r.t. to the background and nearby limbs.

IMHmd is similar to IMHcd, but instead of using the corresponding pixel in the central cell as a reference flow, it uses the average of the corresponding pixels in all 9 cells. The resulting 9 histograms are normalized as a block.

IMHwd is also similar to IMHcd but uses Haar wavelet like operators rather than non-central differences, as shown in Fig. 4(b).

ST Diff: a scheme inspired by Viola et al. [23] based on simple spatiotemporal differencing rather than flow. For each pixel, its 3×3 stride-8 neighborhood at the next time step is taken and its image intensity is subtracted from each of these 9 pixels. The absolute values are accumulated over each cell to make a 9 bin histogram for the cell, which then undergoes the usual block normalization process

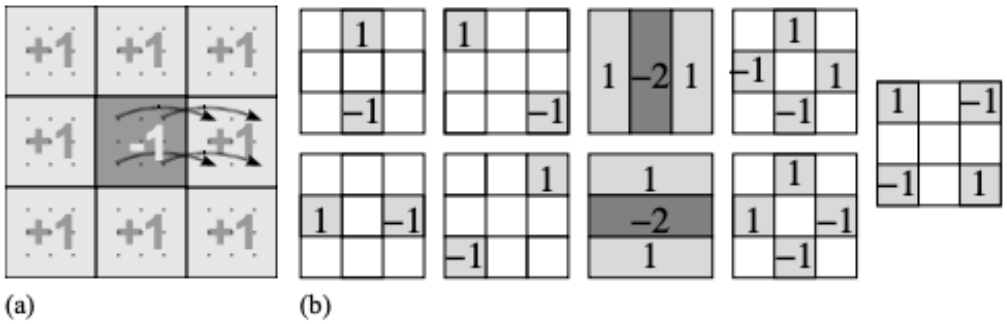


Fig. 4. Different coding schemes for IMH descriptors. (a) One block of IMHcd coding scheme. The block is partitioned into cells. The dots in each cell represent the cell pixels. The arrows emerging from the central cell show the central pixel used to compute differences for the corresponding pixel in the neighbouring cell. Similar differences are computed for each of the 8 neighbouring cells. Values +1 and −1 represent the difference weights. (b) The wavelet operators used in the IMHwd motion coding scheme.

Optical Flow, Motion Boundary and Segmentation for a motion sequence (continued)

Internal/Relative Dynamics Based Coding (continued):

the authors found IMHcd and IMHmd performed best.

Subsequent detectors were the static HOG + IMHcd or HOG + IMHmd.

Note, their study uses Support Vector Machine (SVM) for a solver so they had to cast their problem into K=2 classes and convex objectives (i.e., hit or miss, that is positive or negative or training data containing persons or not containing persons and an objective of false positives per window (?))