Grape Detection Project report

By Eran Geva and Eran Tomer

Goal

Writing an application detecting grapes in an image
Input: Vines' images with grapes
Output: boolean matrix of image's size with:

- 1 if the matching pixel on the image is on a grape
- 0 if it is not

Methodology

Initially all segmentation points in the output are tagged as potential grapes.
Then across multiple phases, we eliminate points with low likelihood of being tagged as a grape.

Preprocessing phase:

1. Reducing the image's color numbers.
2. Extracting the gradient of the reduced color image.

Elimination phase:

1. Eliminating non-grape points by:
    - Crowded mask
    - Angle mask
    - Color masks
2. Image clean-up by multicross mask
3. Eliminating remaining non-grape points by Bagel mask
4. Another image multicross mask clean-up

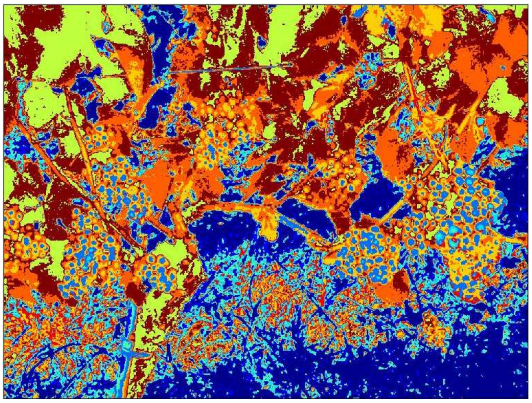
Preprocessing – color reduction

We note that if we reduce the number of colors in the image by mapping similar colors to a same color we can find outlines easily.
Since there exist a wide array of image scenes, a hardcoded color mapping is not practical. We wish to find that mapping dynamically.
For obtaining this we use a clustering algorithm on the colors with the input:

- All colors in the image as vectors of <red, green, blue>
- Expected number of clusters - k

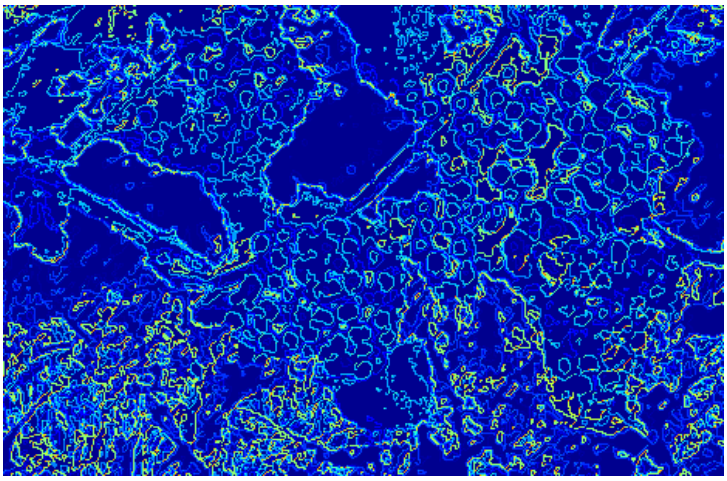
The output is an image where each pixel has an integer value in [1 .. k].
After testing the clustering algorithm with various k values we decided on k=10.
Higher k values resulted with negligible contribution yet the running time increased significantly.
Lower k values resulted with a mapped a wide range of colors to a same cluster thus vital image information was lost.
On the output grapes resemble solid color concentric circles with similar size.


Original image


K-Clustering output

Preprocessing – reduced color image derivation

The clustering image colors are arbitrary but their edges are eminent for grape detection.
We extract the gradient size and direction of the clustered image.
Let us denote this as the clustered gradient image .


K-Clustering gradient sizes

Elimination phase 1 - Independent criteria

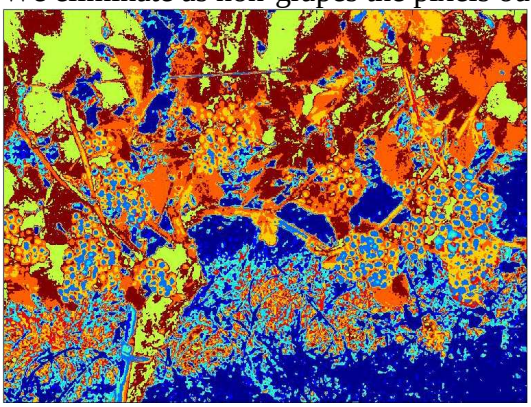
As we mentioned, initially all segmentation points in the output are tagged as potential grapes. Now we'll present several independent criteria masks in which some pixels will be re-tagged as non-grapes (eliminated).
Following this phase we'll apply several incremental phases.

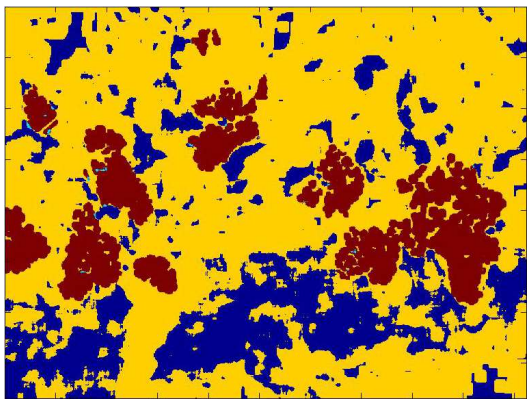
Crowded mask

A grape color is relatively uniform, unlike other areas in the image, therefore grapes' colors are expected to be clustered to a small number of clusters. As a result of this the density of edges in the clustered gradient image is bounded around points contained in grapes. In other words, since the depth, color and surface discontinuity are relatively rare around grape points, the density of edges is expected to be low. Also we see that density of edges in grapes is lower than in the background (grass) but higher than in other regions (leafs).
For each pixel on the clustered gradient image we measure the surrounding box's edge density by:

1. Total edge quantity.
2. The quantity of most frequent gradient sizes.

Empirically we tested various images and searched for boundaries which generate minimal errors and maximal true-eliminations.
We eliminate as non-grapes the pixels outside those boundaries.


Clustered image


Crowded mask output

Legend for each pixel:

	Optimal – no grape	Optimal – grape
Eliminated	**Blue** – eliminated regions without true grapes	**Light blue** - eliminated regions that contain true grapes (false negative)
Still not tagged	**Yellow** - regions not eliminated without true grapes (false positive)	**Red** – regions not eliminated that contain true grapes

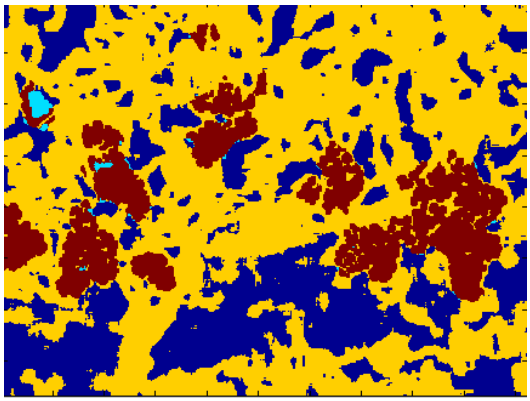
Crowded mask performances (on an arbitrary image):

- Success (Blue): we tagged correctly 24.2% of the non-grape area.
- Failure (Light blue): we tagged incorrectly 0.45% of the grape area.

Angle mask

A grape's shape is round therefore the distribution of its outline's gradient angles is roughly uniform. To detect this we do the following:

- Bucket all gradient angles in a box around every pixel
- Compute a histogram of those angles
- Find relative part of the most frequent angle
- Eliminate pixels outside of tested bounds

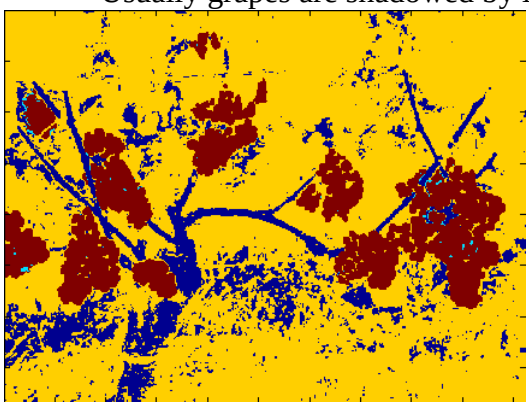


Angle mask output

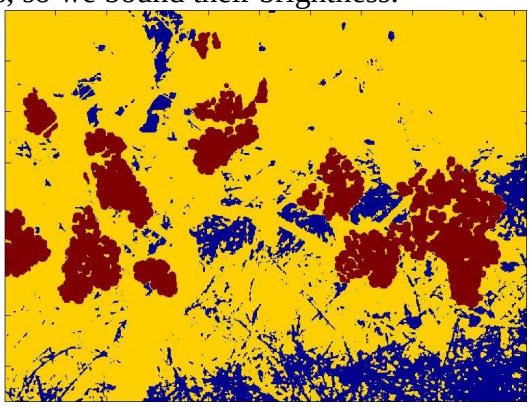
Color masks

We use 2 elimination masks which are based on the colors of pixels.

- Since grapes are green we normalize the green part by the pixel's brightness and eliminate pixels outside a certain range.
- Usually grapes are shadowed by leafs, so we bound their brightness.



Green mask output



White mask output

White mask performances (on an arbitrary image):

- Success (Blue): we tagged correctly 16.59% of the non-grape area.
- Failure (Light blue): we tagged incorrectly 0.04% of the grape area.

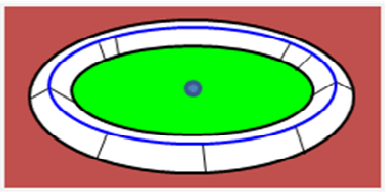
Phases 2, 3, 4 – incremental elimination

Now activate 3 phases to further dilute the un-segmented pixels, each based on the outcome of the previous phase:

- Segmentation clean-up by multicross mask
- Bagel mask on the gradient of the untagged pixels
- Another clean-up by multicross mask

Bagel mask

Grape's outline is round and has a uniform gradient size. Also its inside is relatively empty of edge points.
The following drawing shows a bagel with diameter equal to the measured average grape diameter.

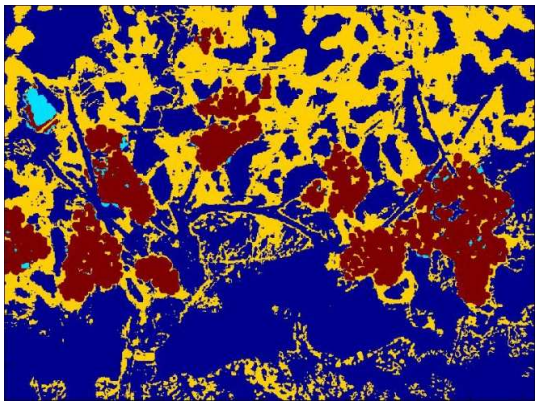


Let us define 2 areas based on the drawing and construct boolean matrices accordingly:

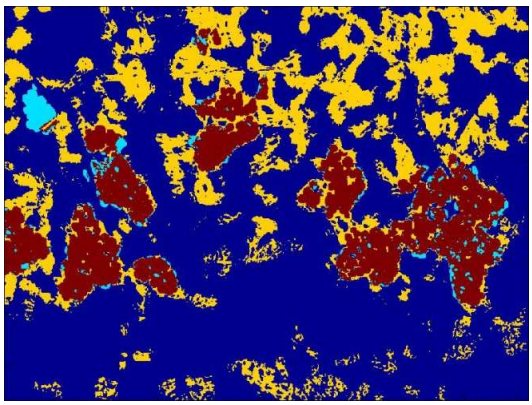
- The inside – In the matrix there is 1 on the **green** area and 0 outside it.
- The bagel – In the matrix there is 1 on the **white** area and 0 outside it.

For each pixel take the surrounding edges (remaining after the phase 1) according to the 2 areas shown on the drawing and filter them out based on the following measures:

- We bound the number of edges inside the bagel (green).
  Edge existence map is a boolean matrix with 1 for every point containing an edge point and 0 otherwise.
  Counting edges around a point is done quickly by convolving the edge existence map with the "inside" matrix.
- We bound the number of most frequent edges on the bagel (white).
  Histogram the gradient sizes on the bagel.
  Find the most frequent one.
  Bound its number of occurrences, if this is outside a tested range, eliminate it.



Bagel mask input



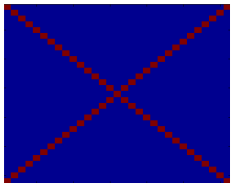
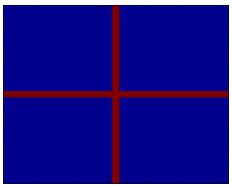
Bagel mask output

White mask performances (on an arbitrary image):

- Success (Blue): we tagged correctly 77.63%of the non-grape area.
- Failure (Light blue): we tagged incorrectly 10.88% of the grape area.

Multicross mask

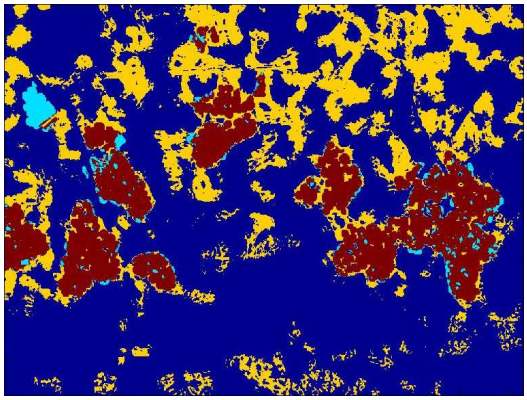
The resulting segmentation of previous phases produced some areas which are suspected as grape points, yet they are too small to be grapes. Moreover grapes are usually convex.
For a given point in the grape, points along some direction from it will also be in the interior of the grape.
To remove those improbable zones we convolve the segmentation map with each of the kernels below a number of times alternately:

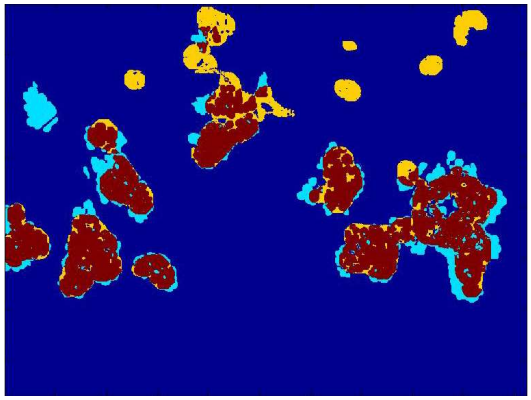



The kernels, where blue is 0 and red is 1.
The “X” kernel is sqrt(2) times smaller than the “+” kernel for normalizing issues.

We define a threshold value which equals to the radius of “+” and “X” and eliminate points with lower results.

We’ll use this clean-up method before and also after the bagel mask to obtain continuous and improved results.


Multicross mask input

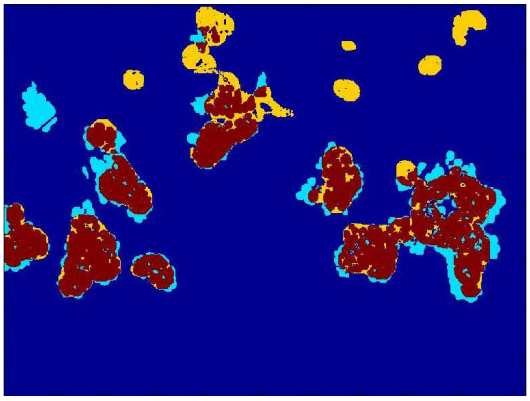

Multicross mask output

White mask performances (on an arbitrary image):

- Success (Blue): we tagged correctly 96.29% of the non-grape area.
- Failure (Light blue): we tagged incorrectly 23.91% of the grape area.

Results


Original image


Final result

We applied the algorithm over the whole given labeled directory.

- On average our successful rate according to the dictated symetric difference evaluation measure is 6.24% (the lower the better)
- Our worst performance over a picture resulted with 22.54%
- Our best performance over a picture resulted with 0.59%
- The standard deviation over the directory 0.047

Conclusion

During the development of the algorithm described in this paper we used many techniques and methods learned throughout the course in addition to many we developed ourselves.
Other then the various techniques mentioned before we also tried to use other methods:

- NCD – learning by normalized compression distance
- Hough transform
- RANSAC
- eliminating by other color combinations
- k-clustering by both colors and x, y coordinates

All these methods resulted with very infirm output.
However we learned that convolution, histograms and k-clustering are powerful tools when used properly.

References

- K-Means Clustering method – the matlab built-in help
- Hough Transform - http://en.wikipedia.org/wiki/Hough_transform
- RANSAC - http://en.wikipedia.org/wiki/RANSAC
- NCD - http://www.complearn.org/ncd.html