IMAGE SEGMENTATION: TRAFFIC SIGN DETECTION EXAMPLE

Aleksandr Galov

LXFT LISTED NYSE

OUTLINE

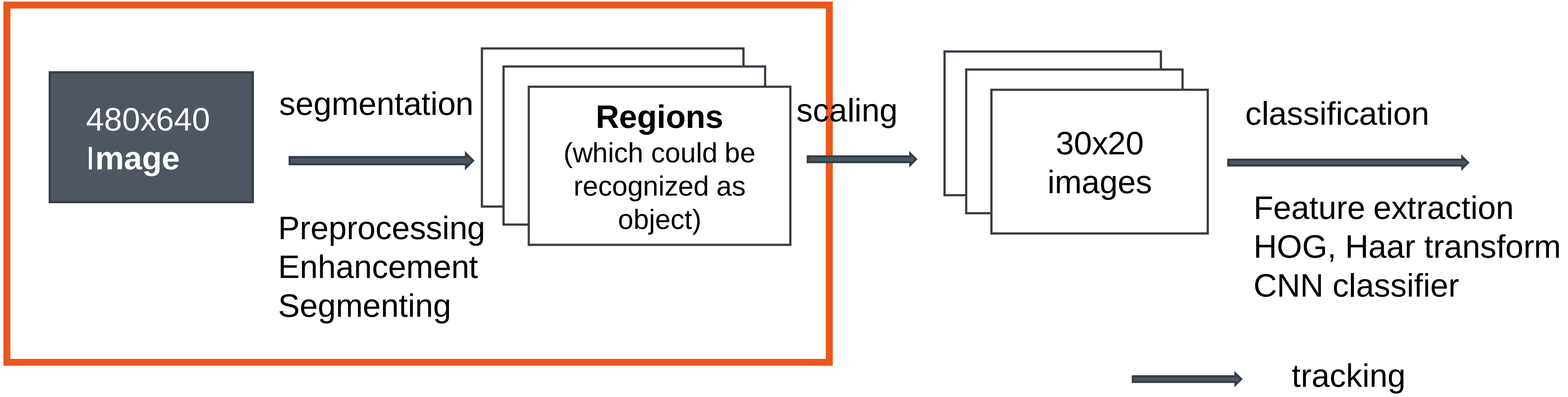
1. Introduction to image segmentation
2. Traffic sign segmentation task
3. K-means
4. Grayscale thresholding
5. Color segmentation

IMAGE SEGMENTATION

- is the process of partitioning a digital image into multiple segments*
- is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics*

*https://en.wikipedia.org/wiki/Image_segmentation

COMPUTER VISION: TYPICAL PIPELINE



ROLE OF SEGMENTATION:

- Convert complex image to simple image segments
- Group pixels of similar colors
- Draw bounding boxes around those pixels

DIFFERENT TYPES OF IMAGE SEGMENTATION

Intensity based

Otsu threshold
Fuzzy logic
Watershred

Similarity based

Seeded region growing
Split and merge

Graph based

RAG merging

Hybrid methods

Subpixel algorithms

Felzenszwalb’s segmentation
Quickshift
K-means
Slic

Theory based

Artificial Neural Network
Hard clustering
Fuzzy clustering

Discontinue based

Sobel operator
Zero-crossing
Perrit operator
Canny edge
Laplacian operator

Choose methods depending on current task

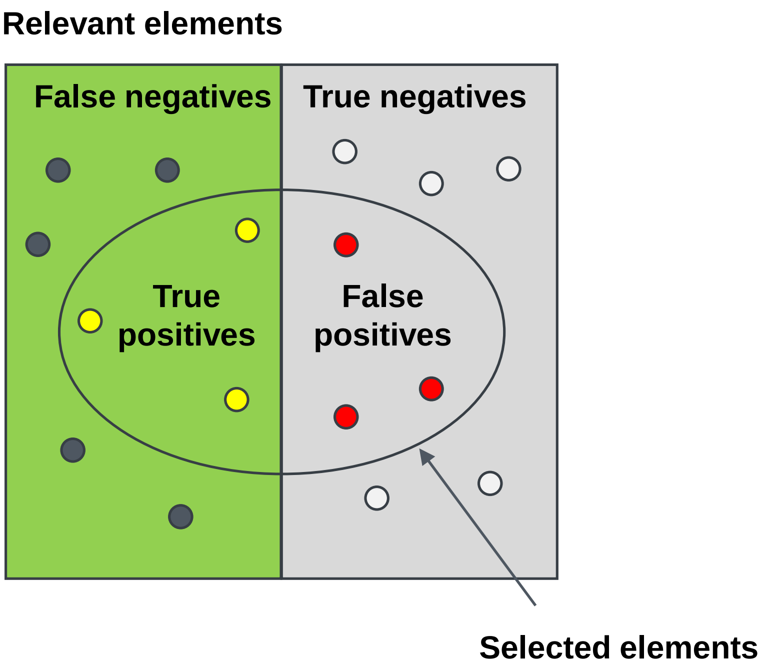
PERFORMANCE

Requirements:

- High recall (detection rate):

$$recall = \frac{relevant\ elements\ \cap selected\ elements}{relevant\ elements}$$

- Keep false positive in check



POSSIBLE STRATEGIES

- Sliding window
  - Control all objects
  - Maximum number of false positives!
- Oversegmentation
  - Select almost all objects of interests
  - Many false positives
- Precise algorithms
  - Be accurate with false negatives!

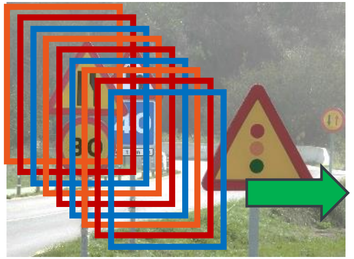


Image from https://pixabay.com/

POSSIBLE STRATEGIES

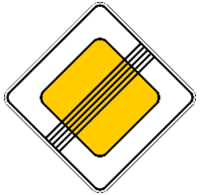
- Sliding window
  - Control all objects
  - Maximum false positives!
- Oversegmentation
  - Select almost all objects of interests
  - Many false positives
- Precise algorithms
  - Be accurate with false negatives!

- **Complex target**
- **Fast classifier**
- **“Brute force” methods**

Preferred for “simple target”

TRAFFIC SIGN SEGMENTATION EXAMPLE

- Designed to be easily recognized by drivers
- Selected colors are used
- Contain pictogram / string of characters



CHALLENGES:

- Background
- Illumination
- Obstacles
- Etc …

TRAFFIC SIGN SEGMENTATION: METHODS

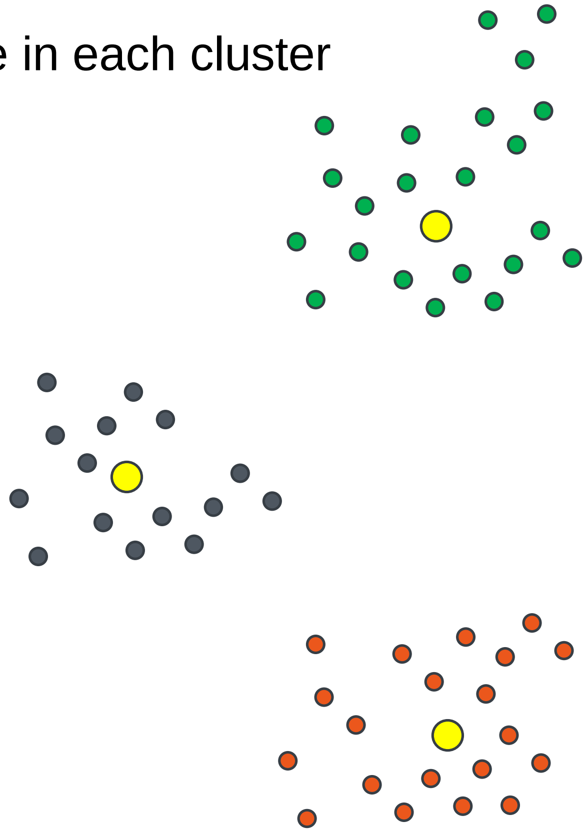
- K-means
  - Can be used as “brute force” approach
  - Can be applied both for grayscale and color images
- Otsu threshold
  - For grayscale image
  - Simple
- Color thresholding
  - For color images
  - Even more simple!

K-MEANS: GENERAL IDEA

- Group pixels over clusters minimizing mean distance in each cluster

Algorithm:

1. Create initial set of k means (i.e. randomly)
2. Repeat until convergence:
    - Assign each pixel to the cluster with least squared Euclidean distance
    - Recalculate the new means



K-MEANS: IMPLEMENTATION

Possible vector components:

- Values of red, green, and blue for RGB image
  - $\mathbf{d}^2{}_1 = (\mathbf{R}_i - \mathbf{R}_j)^2 + (\mathbf{G}_i – \mathbf{G}_j)^2 + (\mathbf{B}_i - \mathbf{B}_j)^2$
- Distance between pixels
  - $\mathbf{d}^2{}_2 = (\boldsymbol{x}_i - \boldsymbol{x}_j)^2 + (\boldsymbol{y}_i – \boldsymbol{y}_j)^2$

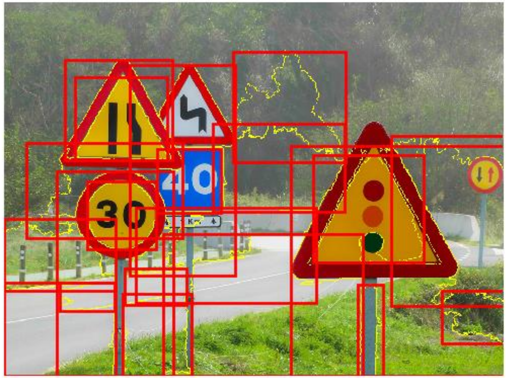
Basic settings:

- Compactness (vary weights between color and distance):
  - $a*\mathbf{d}^2{}_1 + b*\mathbf{d}^2{}_2$
- Number of segments can be varied
- Possible to use for grayscale images

K-MEANS: EXAMPLE

High compactness

Low compactness



Segmented image

- Better than sliding window, but image is still oversegmented
- Can be used as “brute force” approach

Image from https://pixabay.com/

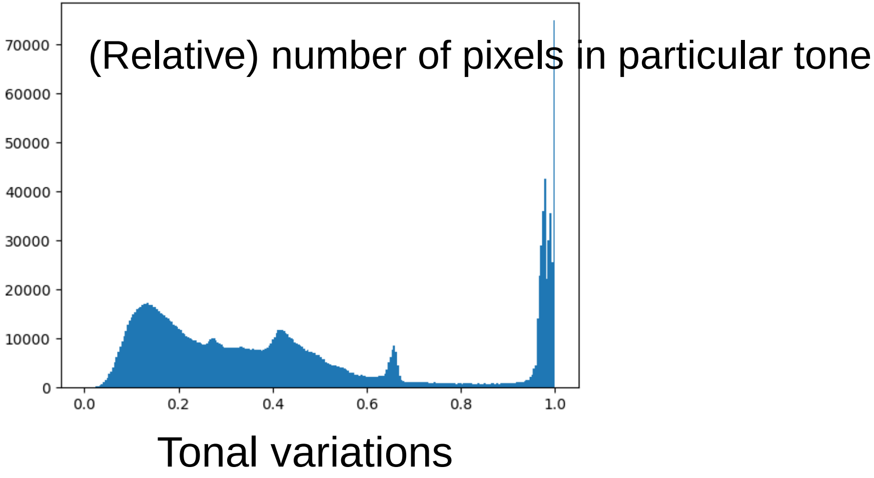
OTSU THRESHOLD

- Use histogram of grayscale image
- Find optimal threshold to separate background and foreground

IMAGE HITOGRAM:

Original image:*

Histogram:



*By Shadowlink1014 - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=2099956

OTSU THRESHOLD: GENERAL IDEA

t – pixel intensity threshold: $i_{\text{background}} < t$, $i_{\text{foreground}} > t$

Class probabilities:

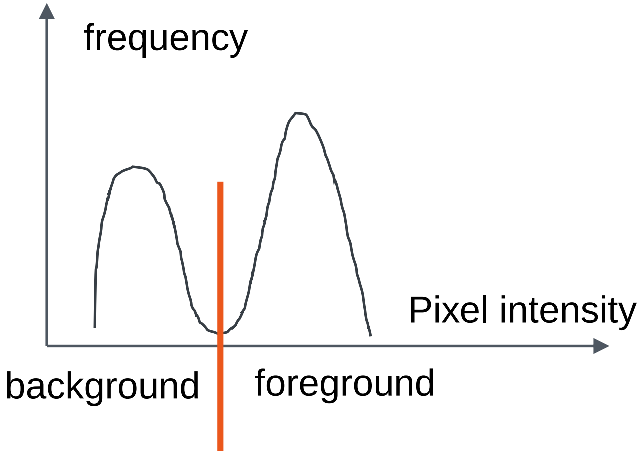
$$\omega_0(t) = \sum_{i=0}^{t-1} p(i)$$

$$\omega_1(t) = \sum_{i=t}^{L-1} p(i)$$

Class means:

$$\mu_0(t) = \sum_{i=0}^{t-1} i\frac{p(i)}{\omega_0}$$

$$\mu_1(t) = \sum_{i=t}^{L-1} i\frac{p(i)}{\omega_1}$$



Intra-class variance:

$$\sigma_b^2(t) = \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2$$

Need to find optimal *t* that maximize intra class variance!

OTSU THRESHOLD: ALGORITHM

1. Compute histogram and probabilities *p(i)* of each intensity level
2. For all possible values (0:255) of threshold t_i :
   1. Compute means μ_0, μ_1 and probabilities ω_0, ω_1 for both classes
   2. Compute intra-class variances σ_b^2
   3. Select threshold t_{max} which corresponds to maximum variance
3. Apply threshold for initial image to produce binary image:
   - Pixel is black if intensity $< t_{max}$
   - Pixel is white if intensity $\geq t_{max}$
4. Label all connected components for segmentation

OTSU THRESHOLD: SIMPLE EXAMPLE

Original image*

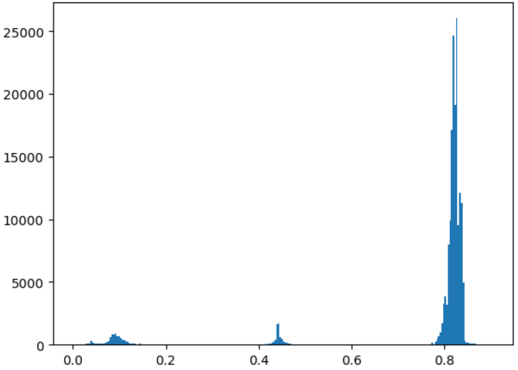
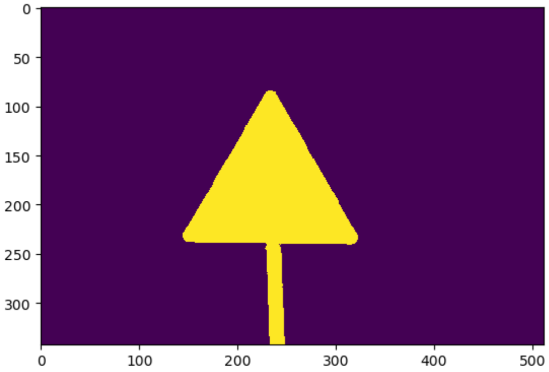


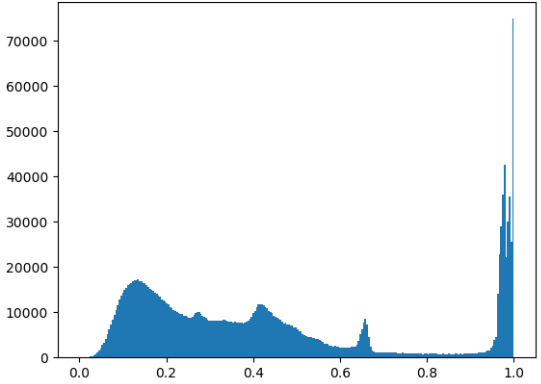
Image histogram

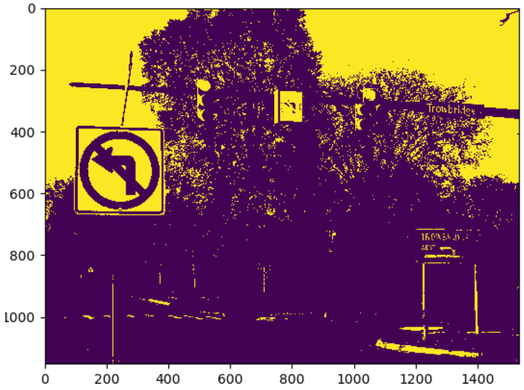


Segmented image

- Histogram can be easily separated into classes

OTSU THRESHOLD: MORE COMPLEX EXAMPLE



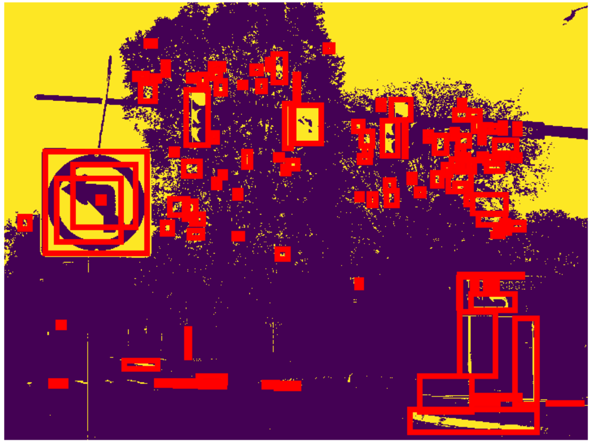


Original image **Image histogram** **Segmented image**

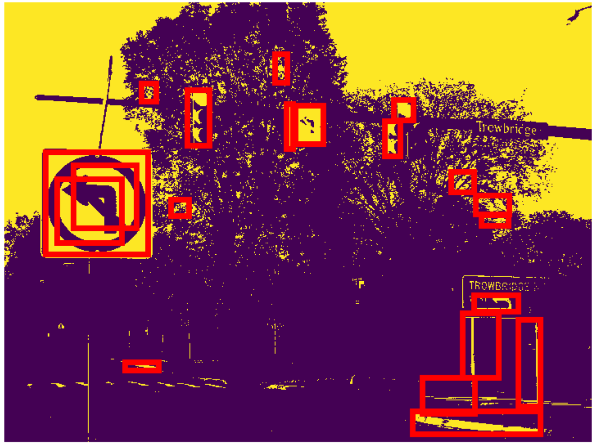
- Multimodal histogram
- **Segmented image is too noisy!**

NOISE REDUCTION: SELECT SIZE OF COMPONENTS

Segmented image



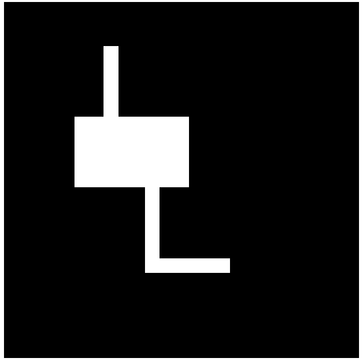
Mark components with more than 100 pixels (Many false positive)



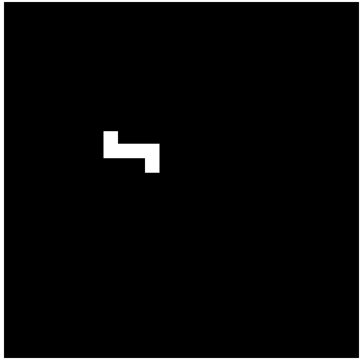
Mark components with more than 500 pixels (Desired performance)

- Find connected components which have more than X pixels
- Draw border rectangles

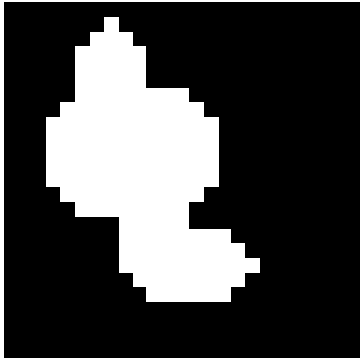
NOISE REDUCTION: MATHEMATICAL MORPHOLOGY



Original



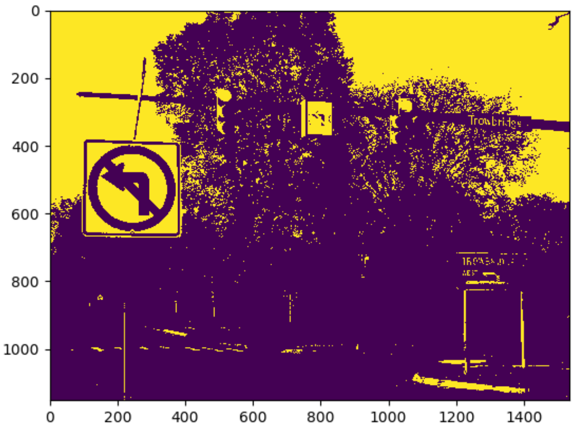
Erosion



Dilation

- Erosion - strips away a layer of pixels from both the inner and outer boundaries of regions
- Dilation - adds a layer of pixels to both the inner and outer boundaries of regions
- Use union/intersection/complement together with erosion and dilation for image modification

NOISE REDUCTION: EROSION EXAMPLE



Segmented image

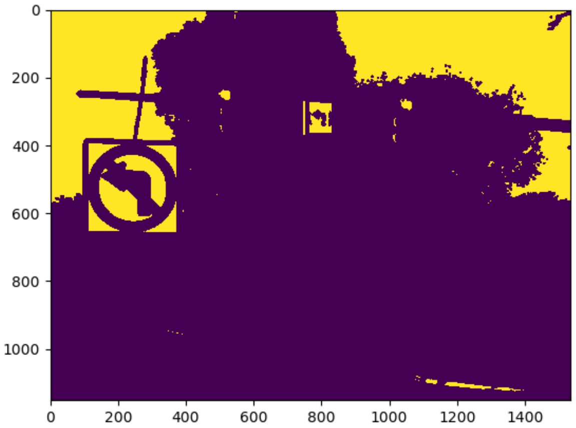
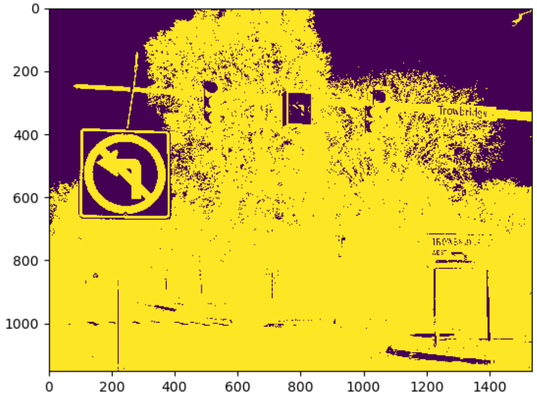


Image after erosion

- Erosion – allows to remove small components from the image
- However road sign is divided by several components!

NOISE REDUCTION: INVERTED DILATION



Inverted segmented image

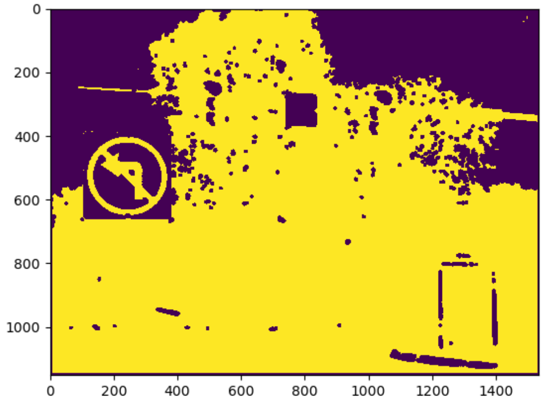
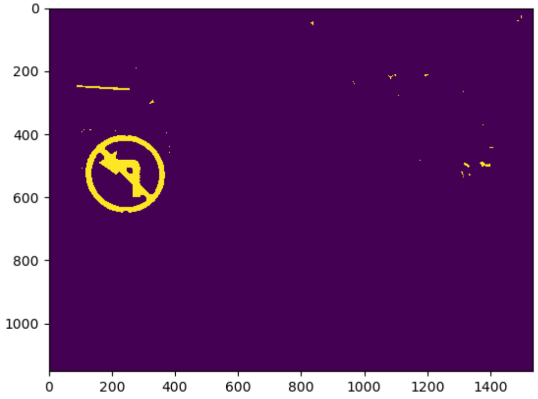


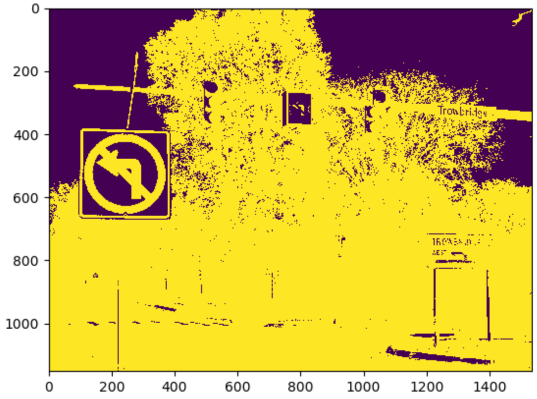
Image after dilation

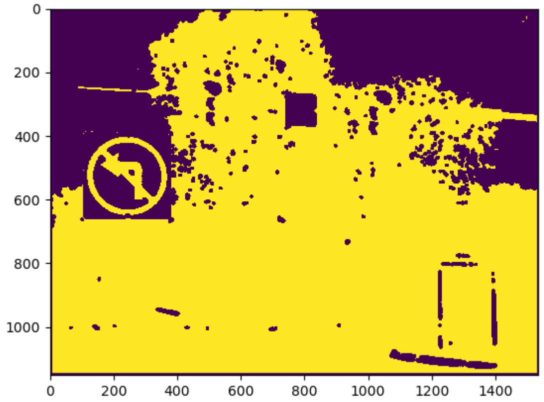


Remove background

- Inversion of image allows to select inner part of road sign

NOISE REDUCTION: INVERTED DILATION





Inverted segmented image **Image after dilation** **Final result**

- Draw border rectangles for components with sufficient number of pixels

COLOR THRESHOLDING: BASIC STEPS

- Choose desired function for pixel color
  - i.e. $f(p) = R / (R + G + B)$
  - i.e. $f(p) = \alpha * R + \beta * G + \gamma * B$
- Calculate function values for all pixels
- Select threshold T for calculated values
- Apply segmentation: produce binary image
  - pixel p is white if $f(p) \geq T$
  - pixel p is black if $f(p) < T$
- Find connected components and border rectangles

RGB image:

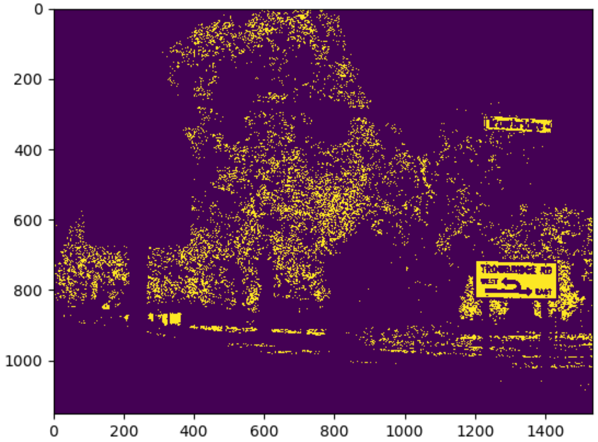
255 000 000 – **red**

000 255 000 – **green**

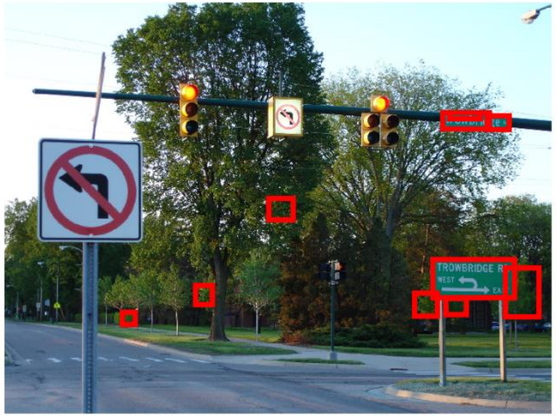
000 000 255 – **blue**

COLOR THRESHOLDING: GREEN COLOR EXAMPLE

Original image



Apply color threshold for green color



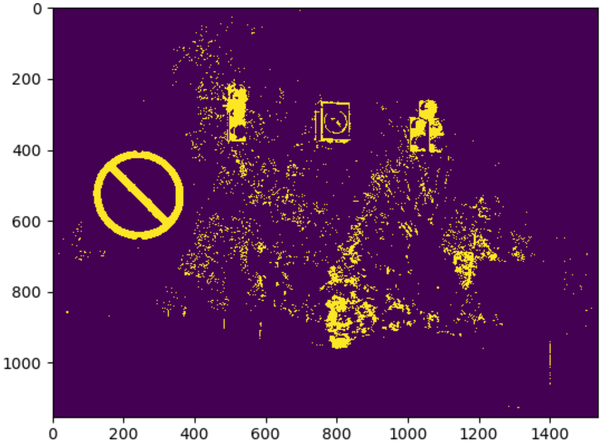
Draw rectangles around connected components with sufficient number of elements

- Function: $f(p) = \dfrac{G}{R + G + B}$
- Threshold: $f(p) > 0.4$

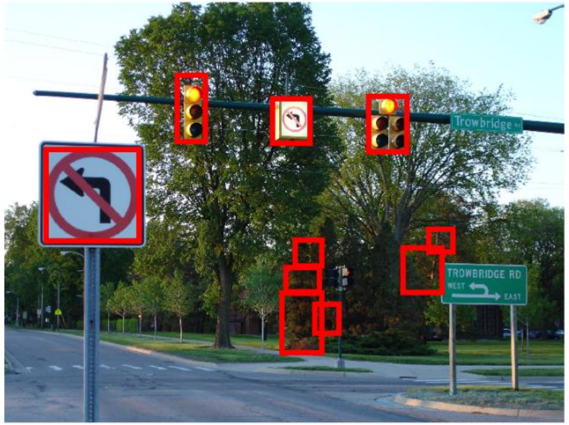
LUXOFT

COLOR THRESHOLDING: RED COLOR EXAMPLE

Original image



Apply color threshold for red color

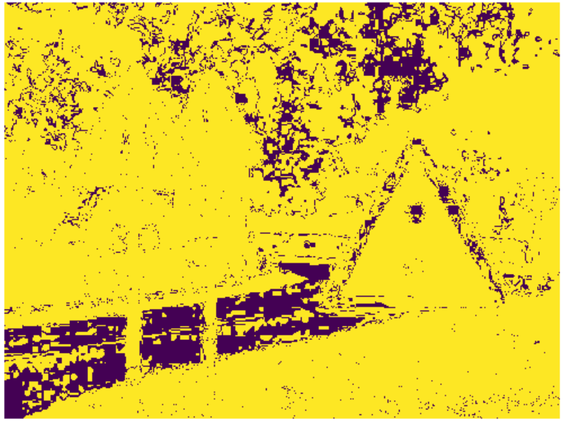


Draw rectangles around connected components with sufficient number of elements

- Function: $f(p) = \dfrac{R}{R + G + B}$
- Threshold: $f(p) > 0.4$

COLOR THRESHOLDING: GRAY COLOR EXAMPLE

Original image*



Apply color threshold for gray colors

Remove gray colors from original image

- Function: $f(p) = |R - B| + |B - G| + |G - R|$
- Threshold: $f(p) > 20$

Image from https://pixabay.com/

COLOR SEGMENTATION: HSV

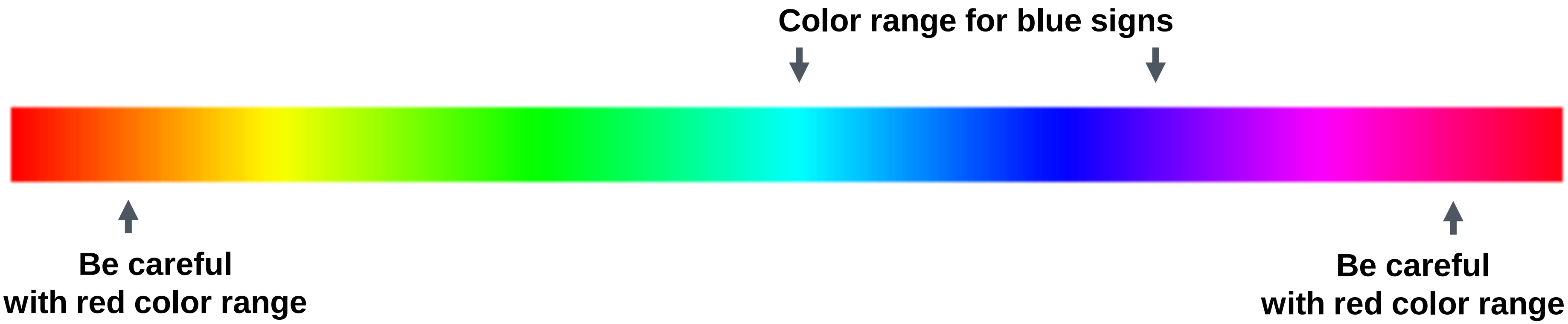
- Separates *luma* from *chroma*
- Easy to separate colors
- Robustness to lightning changes

HSV image:

XXX ___ ___ – **hue**

___ XXX ___ – **saturation**

___ ___ XXX – **value**



COLOR SEGMENTATION: IMAGE PREPROCESSING

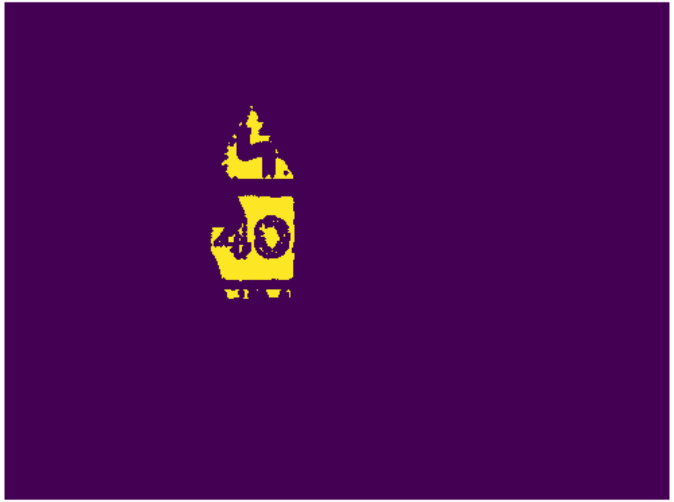
Original image

Image after histogram equalization

Image from https://pixabay.com/

HSV COLOR SEGMENTATION: BLUE RANGE

Enhanced image



HSV blue range after erosion

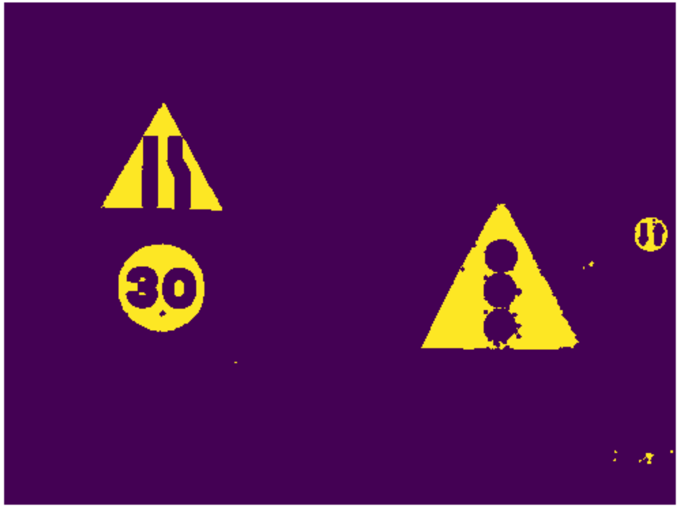
Hue [140: 170]

Saturation [100: 255]

Value [100: 255]

HSV COLOR SEGMENTATION: YELLOW RANGE

Enhanced image



HSV yellow range after erosion

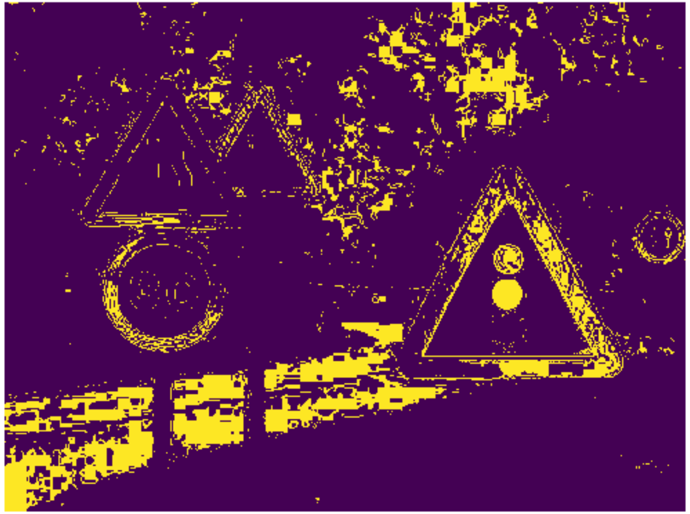
Hue [25: 40]

Saturation [100: 255]

Value [100: 255]

HSV COLOR SEGMENTATION: LOWER RED RANGE

Enhanced image



HSV red range

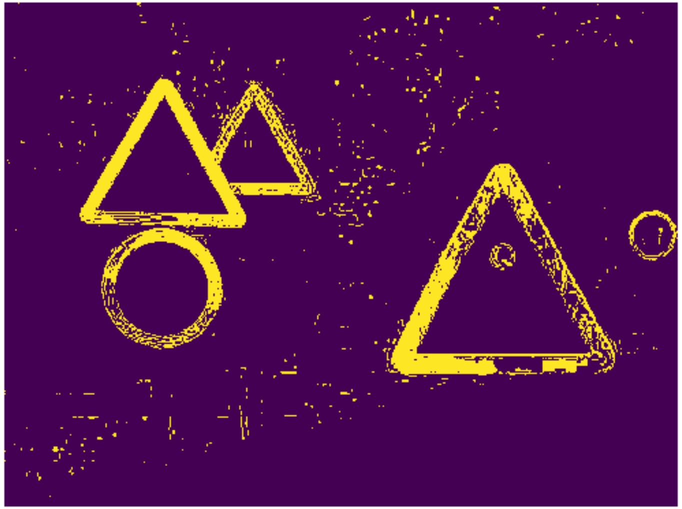
Hue [0: 20]

Saturation [100: 255]

Value [100: 255]

HSV COLOR SEGMENTATION: UPPER RED RANGE

Enhanced image



HSV red range

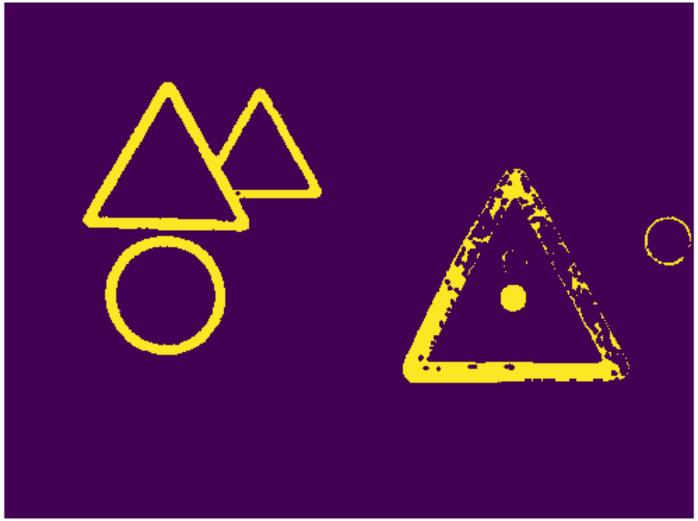
Hue [200: 255]

Saturation [100: 255]

Value [100: 255]

HSV COLOR SEGMENTATION: ALL RED RANGE

Enhanced image



HSV red range after erosion and gray color removal

Hue [0: 20] U [200: 255]

CONCLUSION

- A lot of different methods exists
- Keep attention to recall / number of false positives
- Use color segmentation if possible
- Find connected components with sufficient number of pixels
- Remove noise with mathematical morphology
- Play with image preprocessing

GOOD LUCK!

LUXOFT

LXFT LISTED NYSE