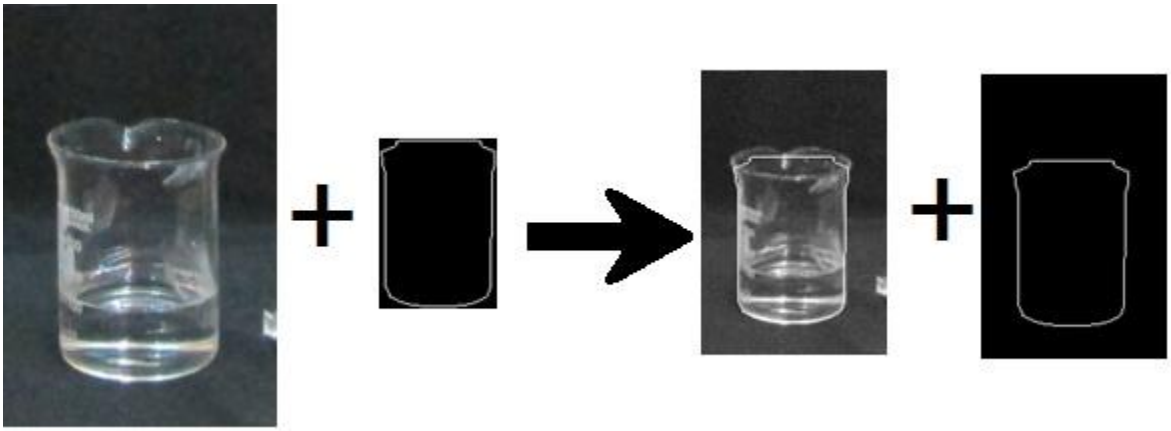
Find object in image using template match



Find object that fit Template *Itm* in image *Is*. The template does not have to fit the size of the object in the image. The program scans various of size ratios of template to image to find the location and size ratios of the best match. The function can use various of methods to trace the template in the image including: Generalized hough transform, Normalize crosscorrelation and other forms of template match. This options are described in the Input(optional) section in the description of the function: MAIN_find_object_in_image (this is the main working function of the program and described in page 5). The output is the boundary and location of the object in the image with the object boundary marked on it.

Input:

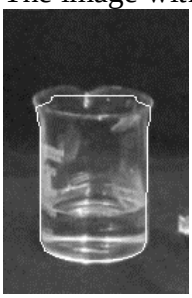
1) Color image with the object to be found.

2) Template of the object to be found. The template is written as binary image with the boundary of the template marked 1(white) and all the rest of the pixels marked 0.

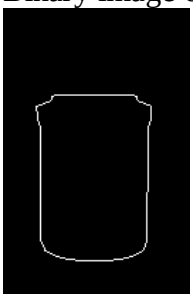
Template of object could be created by extracting the object boundary in image with uniform background, this could be done (for symmetric objects) using the code at: http://www.mathworks.com/matlabcentral/fileexchange/46887-find-boundary-of-symmetric-object-in-image

Output

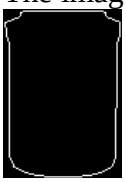
The image with the template marked upon it in the location of and size of the best match.



Binary image of the borders of the template in for the best match (borders of the found object)



The image and template resized to size ratio where the best match where found.



Location on the image (in pixels) were the best match were.
Score of the best match found in the scan (the score of the output).

Instructions

For scanning group of templates on group of images and finding the single best template match to each image see script description of script: *ScanMultipleTemplateOnMultipleImages.m*. in the following pages.

For finding single template in single image see function: *MAIN_find_object_in_image* in the following pages (Or use *ScanMultipleTemplateOnMultipleImages.m* with one template file and one image file).

ScanMultipleTemplateOnMultipleImages.m *(script)*

Scan group of templates on group of images and finding the single best match to each image

The script receives two directories. One directory (*SystemDir* given in line 2 of the script) contains images in which the template should be found in jpg format. Second directory (*TemplateDir* line 1 of the script) contain template images of the vessel in binary tif format. These template files must be binary tif file that contain template of close contour. The script find the location of size of the vessel corresponding to the template in the image. If the templates directory contains more than one template it will scan all templates on each image and will pick the one that give the best match. Template of object could be made by extracting the object contour from background using the script at: http://www.mathworks.com/matlabcentral/fileexchange/46887-find-boundary-of-symmetric-object-in-image

Output:

All output of the scripts will appear as files in the directory of the input image, if the input image named x.jpg all output refer to this image will appear in file names x_*description*.tif.

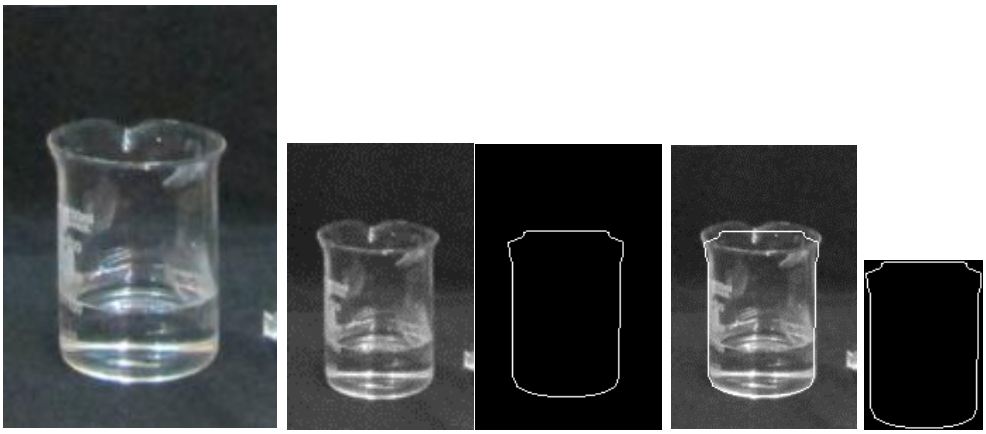


Figure 1 From left to right: The original image (x.jpg) used as input. The resized input image in greyscale (x_SYSTEM.tif) received as output. The boundary of the vessel in the image as binary image (x_BORDERS.tif) received as output (this image size is the same as x_SYSTEM.tif. The image with the found boundary marked upon it (x_MARKED.tif) again as output. Binary image with template of the object (x_TEMPLATE.tif) which could be used to trace the object in other images.

The script writes the boundaries of the object found in the image "x.jpg" in in binary image "x_BORDERS.tif" in the same directory as the original image.

The script also write the image x.jpg (resized and greyscale) with the boundary of the object marked white in file "*x*_MARKED.tif".

The template resized to the size were the best match where made is written as "x_TEMPLATE.tif" in the directory of x.

The input image x.jpg resized to the size in which the best match where made is written as x_SYSTEM.tif.

Note that the borders written in x_BORDERS.tif and the template written in x_TEMPLATE.tif both fit in size and location to the object in the image x_SYSTEM.tif (which is a resize version of the input image x.jpg).

Instructions:

1) Open file *ScanMultipleTemplateOnMultipleImages.m.*
2) Enter the directory in which the images are stored (in jpg color format) in *SystemDir* In line 2 of the script.
3) Enter the directory in which the templates images are stored (in tif binary format) in *TemplateDir* In line 1 of the script.
4) Run script
5) After the script finished running the output should appear in the same directory of the original images (*SystemDir*).

Examples

See directory "EXAMPLE IMAGES" for example input images.

See directory "EXAMPLE TEMPLATES" for example templates image directory.

These two directories are located in the source code directory.

MAIN_find_object_in_image(*Is,Itm,search_mode,border,tm_dilation,neg_cor***)**
Find object that fit Template *Itm* in image *Is.* The template does not have to fit the size of the object in the image. The program scan various of size ratios of template to image to find the location and size ratios of the best match. The function can use various of methods to find the template in the image including: generalized hough transform, normalize crosscorrelation and other form of template match. This option are described in the Input(optional) section below.

Input (Essential):

Is: Color image with the object to be found.

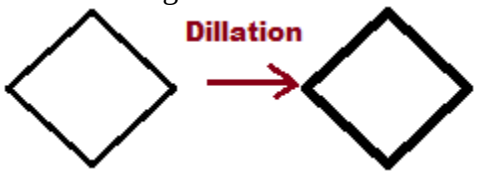
Itm: Template of the object to be found. The template is written as binary image with the boundary of the template marked 1(white) and all the rest of the pixels marked 0. The template must be line of close contour.
Template of object could be created by extracting the object boundary in image with uniform background, this could be done (for symmetric objects) using the code at:
http://www.mathworks.com/matlabcentral/fileexchange/46887-find-boundary-of-symmetric-object-in-image

Input (Optional):

search_mode: The method by which template *Itm* will be searched in image *Is:*
search_mode='*hough*': use generalized hough transform to scan for template.
search_mode='*template*': use cross-correlation to scan for template in the image (default).

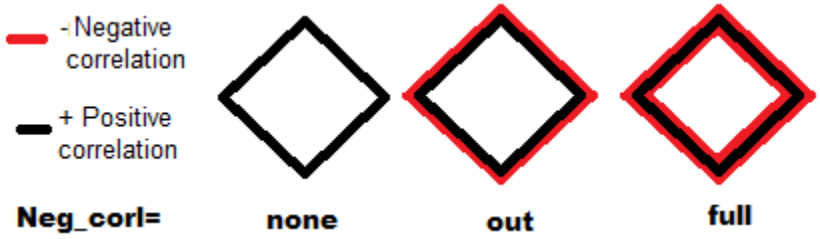
border: Only in case of *search_mode*='template', the *border* parameter determine the type of image to which the template will be matched (edge, gradient, greyscale).
border='sobel': Template Itm will be matched (cross-correlated) to the to 'sobel' gradient map of the image *Is.*
border='canny': Template Itm will be matched (cross-correlated) to the to 'canny' binary edge map of the image *Is* (default).
Else Template *Itm* will be matched (crosscorrelated) to the greyscale version of the image *Is.*

tm_dilation: The amount of dilation for of the template. How much the template line will be thickened (in pixels) for each side before crosscorelated with the image. The thicker the template the better its chance to overlap with edge of object in the image and more rigid the recognition process, however thick template can also reduce recognition accuracy. The default value for this parameter is 1/40 of the average dimension size of the template *Itm.*



neg_corl: Only in case of *search_mode*='template'. Matching the template to the edge image is likely to give high score in any place in the image with high edge density which can give high

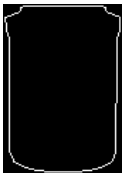
false positive, to avoid this few possible template match option are available:



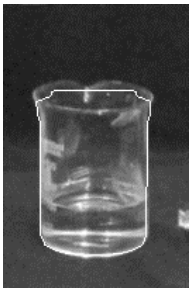
neg_corl='out': Use negative template (negative correlation) surrounding the template contour (in small radious around the template line) but only in the outside of the template. Crosscorrelation of this areas with edges in the canny image will reduce the match score of the template in this location (default).
neg_corl='full': Use negative template (negative correlation) around the template contour line (in small radious around the template line) for both the inside and outside of the template. Crosscorrelation of this areas with edges in the canny image will reduce the match score of the template in this location (default).
neg_corl='none': Use the template as it is.

Output

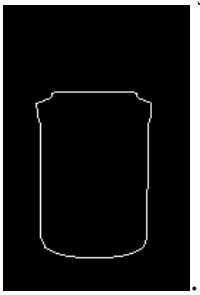
Isresize: The image (*Is*) resized to size where the best match where found
***Itmresize*:** The template (*Itm*) resized to size where the best match where found. Note that the *Itmresize* template match exactly the size of the object in the image Isresize.



Ismarked: The image (*Isresize*) with the template marked upon it in the location of and size of the best match.



***Iborders*:** Binary image of the borders of the template/object in *Isresiz* for the best match (Edges of the found object in the image *Isresize*).


.

Ysizesys,Ysizeitm the size of the y axes of *Isresize* and *Itmresize* respectively.
ybest xbest: location on the image (in pixels) were the template were found (for the resize template/image in Isresize,Itmresize).
Borders_XY: array of x,y coordination of the border point in *Iborder*.
BestScore: Score of the best match found in the scan (the score of the output).

Algorithm:

The function scan the various of ratios of image to template and for each ratio search for the template in the image. The size ratio and location in the image that gave the best match for the template are chosen.
Scanning of size ratios of image to template is done by two loops:
First loop: shrink the image *Is* by 0.5% in each cycle until it reach the size of the original template *Itm*. For each cycle in the loop scan the original template *Itm* in the resize version of Is. If the best match have higher score then previously best match write it.
Second loop: shrink the template by 0.5% in each cycle until it reach minimal dimension of 100 pixels. For each cycle in the loop scan for the resize template *Itm* in the original image *Is*. If the best match have higher score then previously best match write its location and the size ration.
Use size ratio and location that gave best match as output.

Template_match_vessel_extraction(*Is,Itm,expmode,edgetype,numdilate, thresh****)***
Find template *Itm* in greyscale image *Is* using various of forms of template match specified by the optional input parameters.
Return the *x,y* coordniates of the best match.
Also return the *score* of the best match.
(there is no resizing or rotating of the template or image during scan).
For the various of working mode of the function see the optional INPUT (optional) section. All method are in general based on crosscorrelation between the template and the image or edge image.

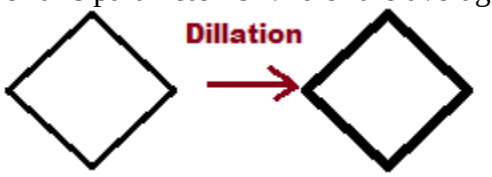
INPUT (essential)

Is: Greyscale picture were the template *Itm* should be found.
Itm: Binary edge image of the template with edges marked 1 and the rest 0.

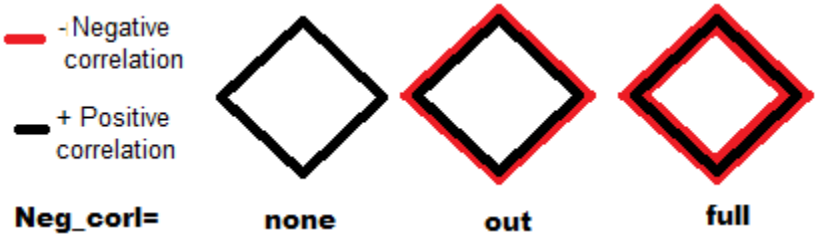
INPUT (optional)

border: The *border* parameter determine the type of image to which the template will be matched (edge, gradient, greyscale).
border='sobel': Template Itm will be matched (crosscorrelated) to the to 'sobel' gradient map of the image *Is*.
border='canny': Template Itm will be matched (crosscorrelated) to the to 'canny' binary edge map of the image *Is* (default).
Else Template *Itm* will be matched (crosscorrelated) to the greyscale version of the image *Is*.

tm_dilation: The amount of dilation for of the template. How much the template line will be thickened (in pixels) for each side before crosscorelated with the image. The thicker the template the better its chance to overlap with edge of object in the image and more rigid the recognition process, however thick template can also reduce recognition accuracy. The default value for this parameter is 1/40 of the average dimension size of the template *Itm*.



neg_corl: Matching the template to the edge image is likely to give high score in any place in the image with high edge density which can give high false positive, to avoid this few possible template match are available:



neg_corl='out': Use negative template (negative correlation) surrounding the template contour (in small radious around the template line) but only in the outside of the template. Crosscorrelation of this areas with edges in the canny image will reduce the match score of the template in this location (default).
neg_corl='full': Use negative template (negative correlation) around the template contour line (in small radious around the template line) for both the inside and outside of the template. Crosscorrelation of this areas with edges in the canny image will reduce the match score of the template in this location (default).
neg_corl='none': Use the template as it is.

OUTPUT

x,y: Coordninates of template *Itm* in image *Is* for the best match (Location the edge point [1,1]) of the template *Itm* in I*s* for the match with the highest score.

score: Score of the best match.

Generalized_hough_transform(Is, Itm)

Find template *Itm* in greyscale image *Is* using generalize hough transform
Return the *x,y* coordniates of the best match.
Also return the *score* of the best match.
(there is no resizing or rotating of the template or image during scan).

INPUT

Is: Greyscale picture were the template *Itm* should be found
Itm: Binary edge image of the template with edges marked 1 and the rest 0

OUTPUT

x,y: Coordninates of template Itm in image *Is* for the best match (Location the edge (point [1,1]) of the template *Itm* in I*s* for the match with the highest score).

score: Score of the best match.