LESSON 10

Coding with Matlab

A complete example about the similarity in images



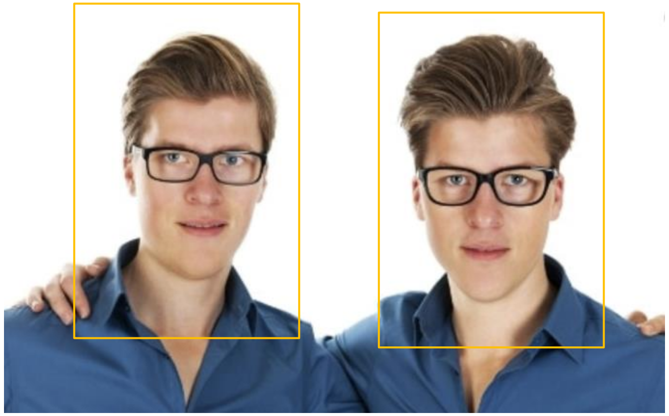

Lesson outline

- Concept of Cross-Correlation
- Matlab first steps with images
- Coding an example for similarity
- **Main points**

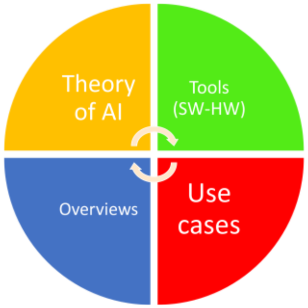



THEORY

Similarity in images via cross-correlation



What is relevant what is useless?



Similarity by Cross-correlation

- Is a measure of similarity of two vectors (matrices) as a function of the displacement of one relative to the other.
- The function returns
  - the similarity value
  - and the position with the better alignment of the two images.

[similarity, displacement] = image_xcorr(I1, I2)

Similarity by Cross-correlation

- The "core of cross-correlation":
  for image-processing applications in which the brightness of the image and template can vary due to lighting and exposure conditions, the images can be first normalized
  - This is typically done at every step by subtracting the mean and dividing by the standard deviation.
    Template = t(x,y), Subimage f(x,y)

<u>Repeat</u> shifting the subimage *f* to get the alignment points checking where is the maximum value of NCC *f(x+i,y+j)*

Normalized Cross-Correlation = (NCC)

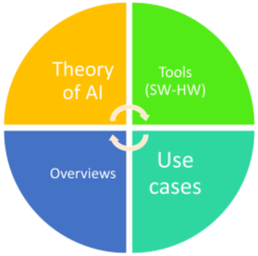
$$t * f = \frac{1}{n}\sum_{x,y}\frac{1}{\sigma_f\sigma_t}\left(f(x,y)-\mu_f\right)\left(t(x,y)-\mu_t\right).$$




Toolboxes

Matlab

First step with images and similarity



Knowing **what to learn** from the coding lesson?

- Matlab first steps with images
- Understanding the structure and steps of the coding example for the image similarity
- During the exam will be no exact coding
- It necessary to understand the main procedures (+ what are inputs and outputs)
- For example, today we will better understand the cross-corr.

Why similarity is so important?

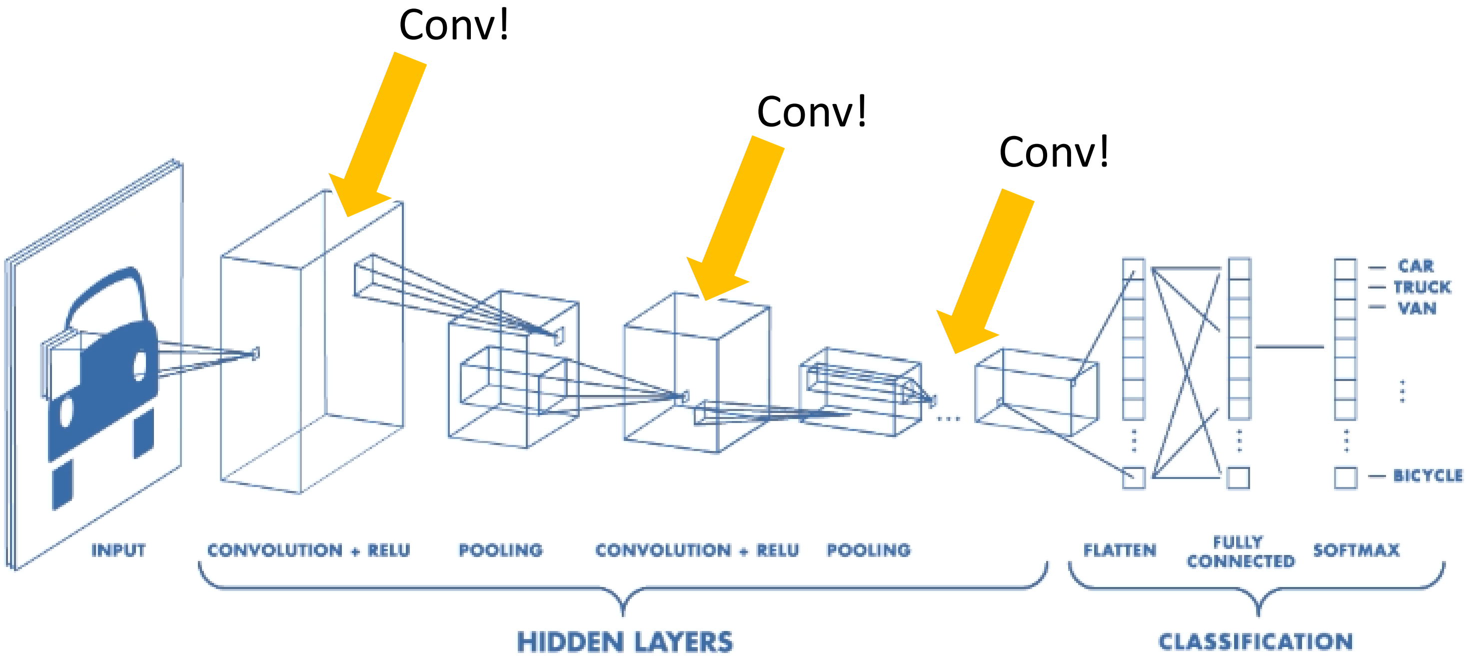
- Because it is one of the basis/kernel/primitive of the **pattern recognition** and **machine learning**

- **You/The Intelligent system can recognize a pattern/object/vector/image because it is similar to a previously labelled one seen the learning phase**

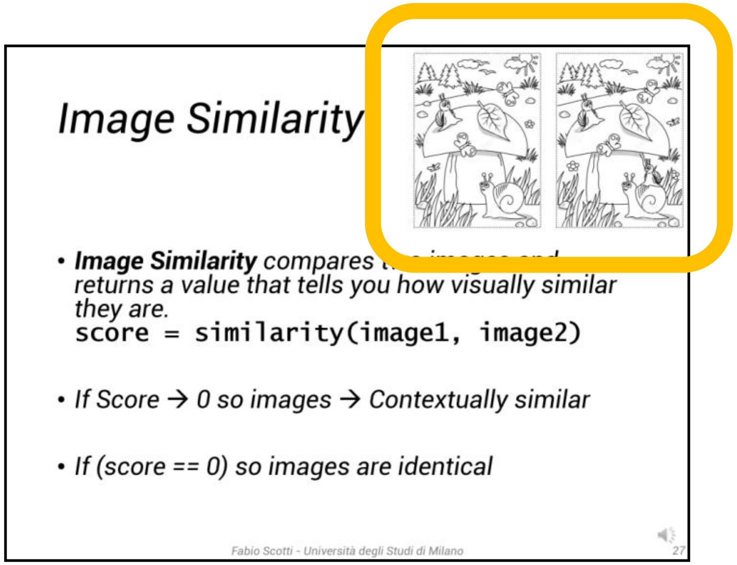
Why similarity is so important?

Convolution ←→ Correlation

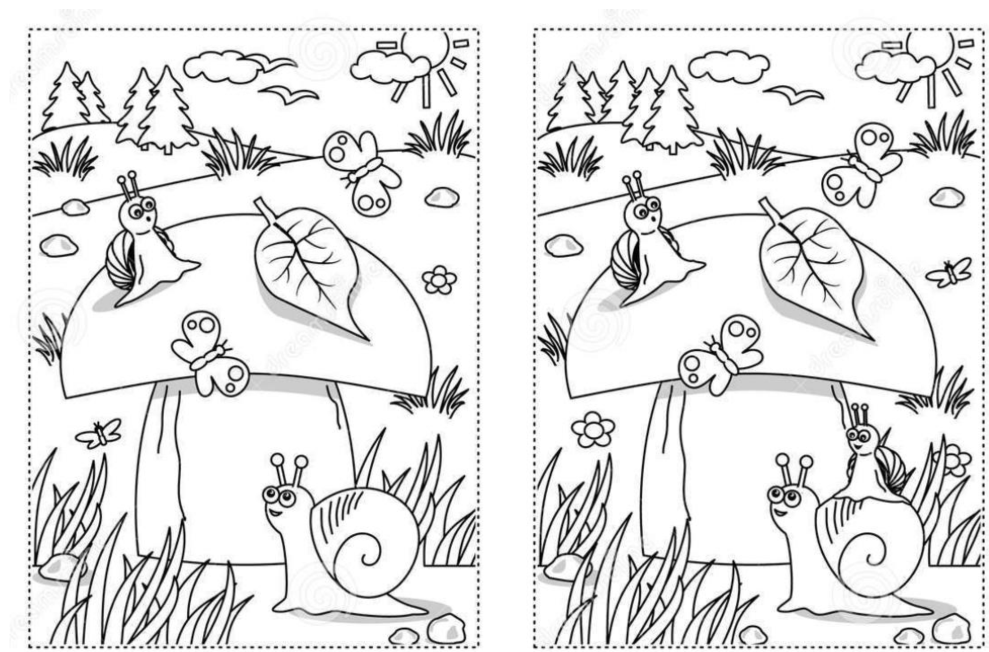
Convolutional Neural Networks



Have you noticed the puzzle?
(Lesson 9)



Did you find all the differences?
Are you sure?



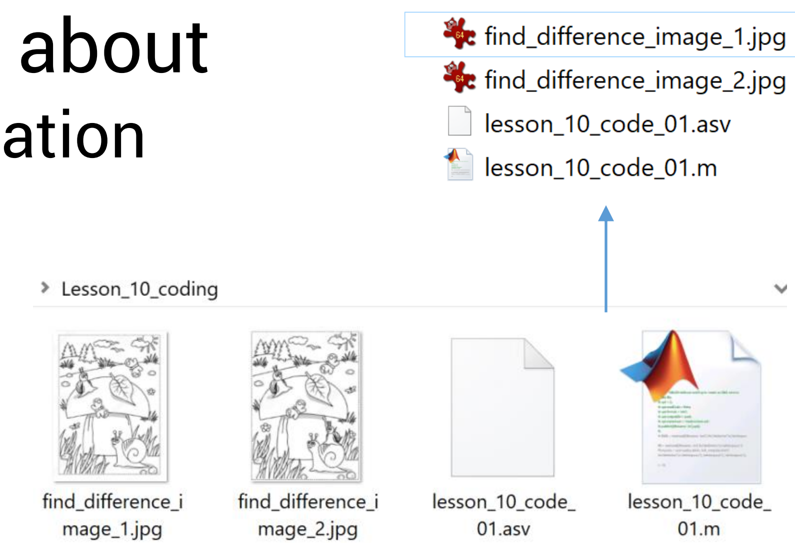
How to automate the task?

All what we need is

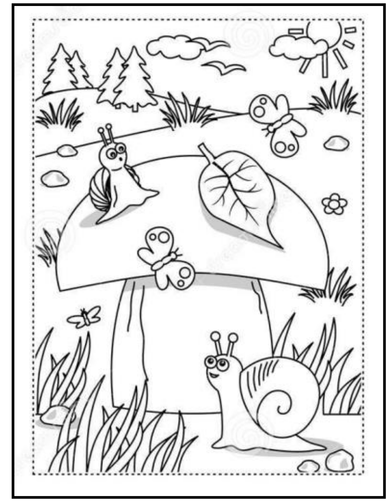
- Concept of image cross-correlation
- …and a little of matlab coding

The material you need

- Matlab (last version)
- Download the files of the lesson in local folder
- Do not jump to the solution….
- Follow the process,
  you will gather information about
  - Image format and representation
  - Image manipulation
  - Image plotting
  - Image similarity



Loading the files



Not compiled!

```
close all
clear all  % do not use it in case of debug...

img_template = imread('find_difference_image_1.jpg');
img_subimage = imread('find_difference_image_2.jpg');
```

```
Name              Size              Bytes  Class

img_subimage      691x505x3       1046865  uint8
img_template      736x555x3       1225440  uint8
```

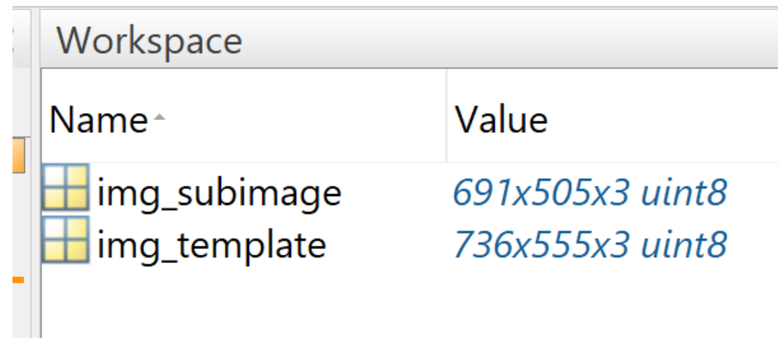
Color image (RGB)

Unsigned char

Unsigned 8-bit integers.
The uint8 range is from 0 to 255

Have a look to the variables

- The workspace



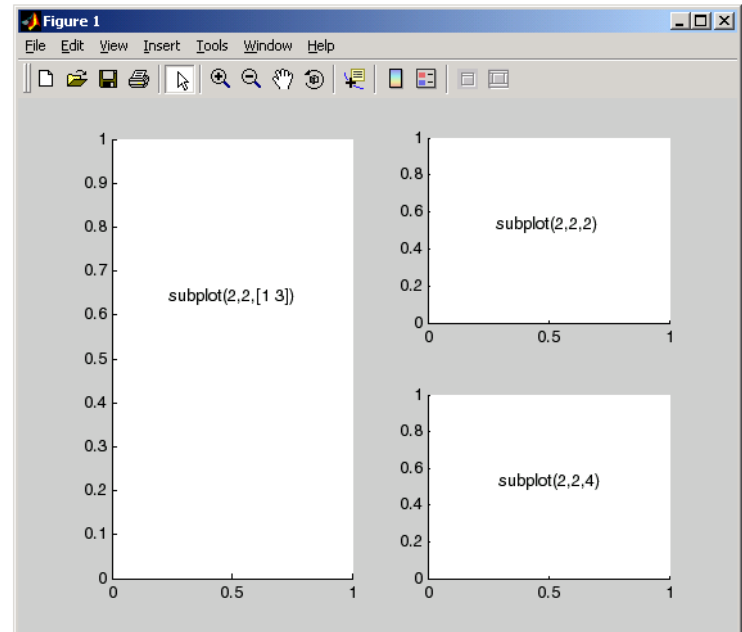
- >> whos

Name	Size	Bytes	Class
img_subimage	691x505x3	1046865	uint8
img_template	736x555x3	1225440	uint8

Plotting images

handle

```
h1 = figure;
subplot(1,4,1)
imshow(img_template, [])
title('template')


subplot(1,4,2)
imshow(img_subimage, [])
title('subimage')
```



Plotting images (2)

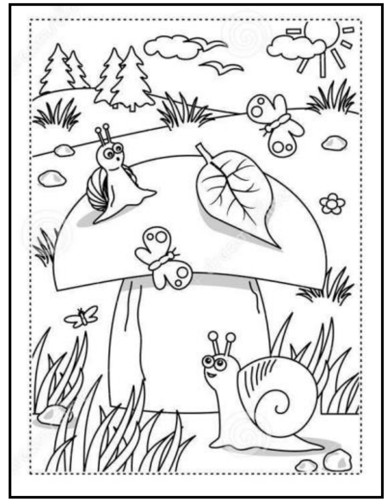

```
h1 = figure;
subplot(1,4,1)
imshow(img_template, [])
title('template')


subplot(1,4,2)
imshow(img_subimage, [])
title('subimage')
```

Auto color scale **(Square brackets)**

imshow(I,[]) displays the grayscale image I scaling the display based on the range of pixel values in I. **imshow** uses [min(I(:)) max(I(:))] as the display range, that is, the minimum value in I is displayed as black, and the maximum value is displayed as white.

How to see the difference?

The general idea is to subtract two images to find the differences



Name	Size	Bytes	Class
img_subimage	691x505x3	1046865	uint8
img_template	736x555x3	1225440	uint8

1) Images are different in size → we can't just subtract them (impossible in every coding language)
2) We have to place properly the second image to avoid false detections

Color conversion

```
% color 2 gray conversion
img_template_gray = rgb2gray( img_template );
img_subimage_gray = rgb2gray( img_subimage );
```

```
Name                    Size              Bytes  Class

h1                      1x1                   8  matlab.ui.Figure
img_subimage            691x505x3       1046865  uint8
img_subimage_gray       691x505          348955  uint8
img_template            736x555x3       1225440  uint8
img_template_gray       736x555          408480  uint8
```

The handle of the figure

gray images

Image normalization (mean subtraction)

```
img_template_gray_norm = img_template_gray - mean(mean(img_template_gray)) ;
img_subimage_gray_norm = img_subimage_gray - mean(mean(img_subimage_gray)) ;
```

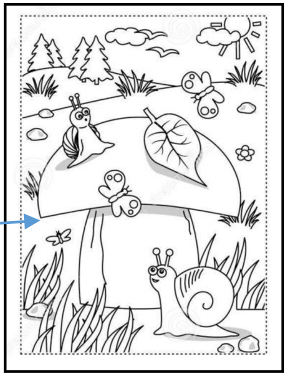
Just without the means (no standatd deviations)

Two times..

```
mean   Average or mean value.
    S = mean(X) is the mean value of the elements in X if X is a vector.
    For matrices, S is a row vector containing the mean value of each
    column.
    For N-D arrays, S is the mean value of the elements along the first
    array dimension whose size does not equal 1.
```

```
>> mean(mean(img_template_gray))

ans =

  228.9717
```

It's quite white!



white == 255
@uint8

2D correlation (similarity evaluation)

GENERAL

Normalized cross-correlation = (NCC) $t * f =$

$$\frac{1}{n}\sum_{x,y}\frac{1}{\sigma_f\sigma_t}\left(f(x,y)-\mu_f\right)\left(t(x,y)-\mu_t\right).$$

MATLAB → with no scaling (direct!)

```
% xcorr2(A,B) computes the crosscorrelation of matrices A and B.
crr = xcorr2(img_template_gray_norm , img_subimage_gray_norm);
```

2-D Cross-Correlation

The 2-D cross-correlation of an M-by-N matrix, X, and a P-by-Q matrix, H, is a matrix, C, of size $M+P-1$ by $N+Q-1$.

$$C(k,l)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}X(m,n)\,\overline{H}(m-k,n-l),\qquad \begin{array}{l}-(P-1)\le k\le M-1,\\ -(Q-1)\le l\le N-1,\end{array}$$

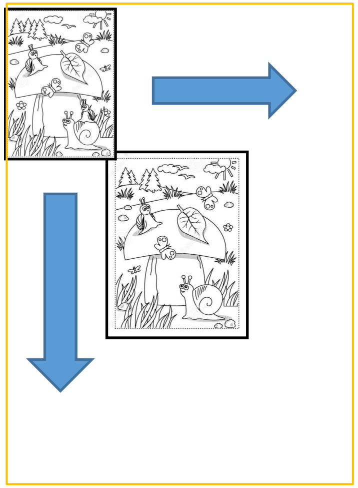
2D correlation (similarity evaluation)

```
% xcorr2(A,B) computes the crosscorrelation of matrices A and B.
crr = xcorr2(img_template_gray_norm , img_subimage_gray_norm);
```

Normalized cross-correlation = (NCC) $t * f = \frac{1}{n}\sum_{x,y}\frac{1}{\sigma_f\sigma_t}\left(f(x,y)-\mu_f\right)\left(t(x,y)-\mu_t\right).$

Repeat shifting the subimage *f* to get the alignment points checking where is the maximum value of NCC *f(x+i,y+j)*

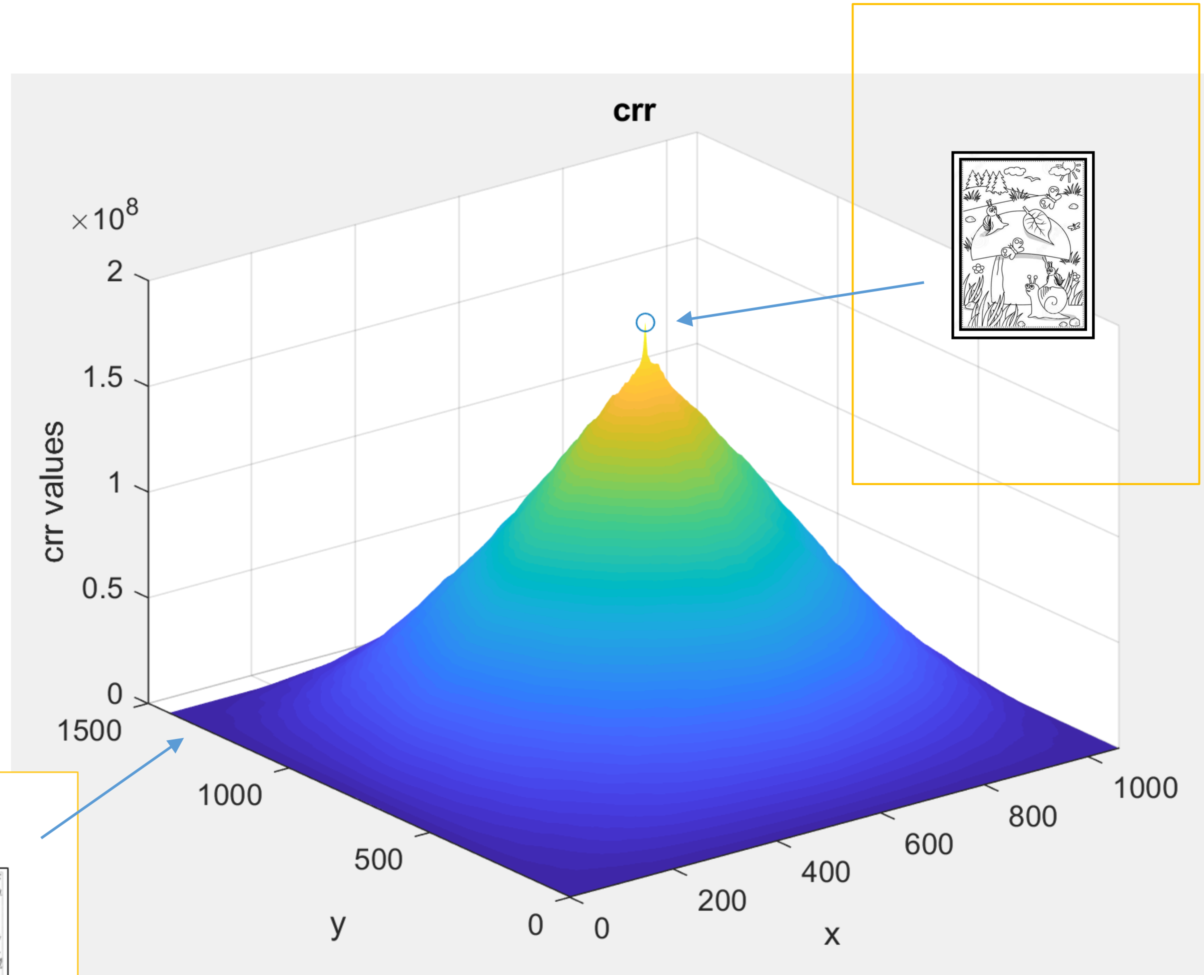
```
>> whos crr
  Name          Size                 Bytes  Class     Attributes

  crr         1426x1059            12081072  double

Name                     Size                  Bytes  Class

h1                        1x1                      8  matlab.ui.Figure
img_subimage            691x505x3            1046865  uint8
img_subimage_gray       691x505               348955  uint8
img_template            736x555x3            1225440  uint8
img_template_gray       736x555               408480  uint8
```



How is the output of a xcorr2?

```
%%
figure
surf(crr);
shading interp;
title('crr')
```

just to remove black
line around the tiles

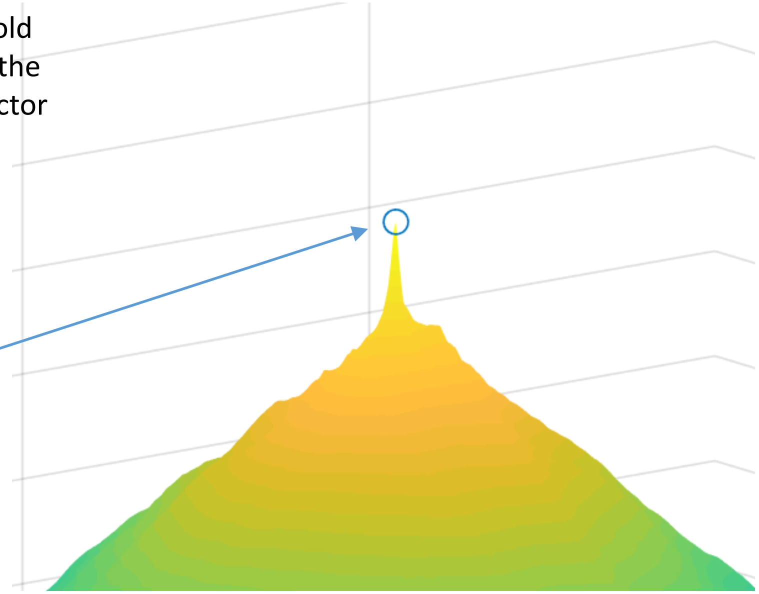


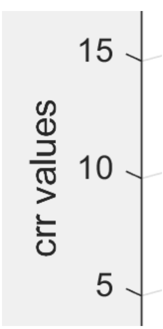
A real spike! Locate the max in ccr!

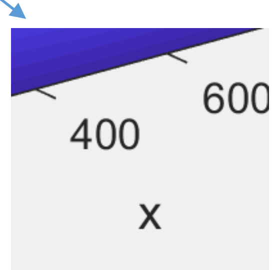
operator (:) unfold the elements of the matrix in a 1D vector

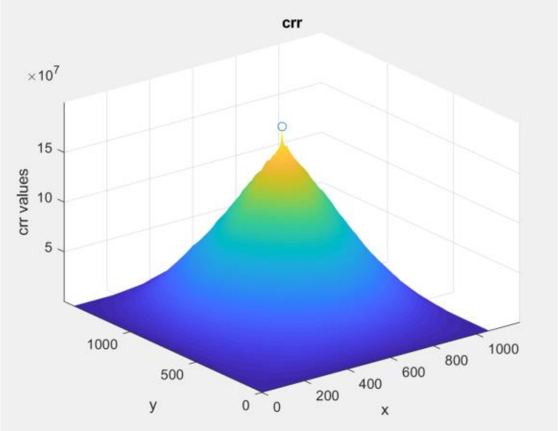
```
% find the maximum
[ssr,snd] = max( crr(:) );
[ij,ji] = ind2sub(size(crr),snd);
```

```
%%
hold on
plot3( ji, ij, ssr,'o')
xlabel('x');
ylabel('y');
zlabel('crr values');
```









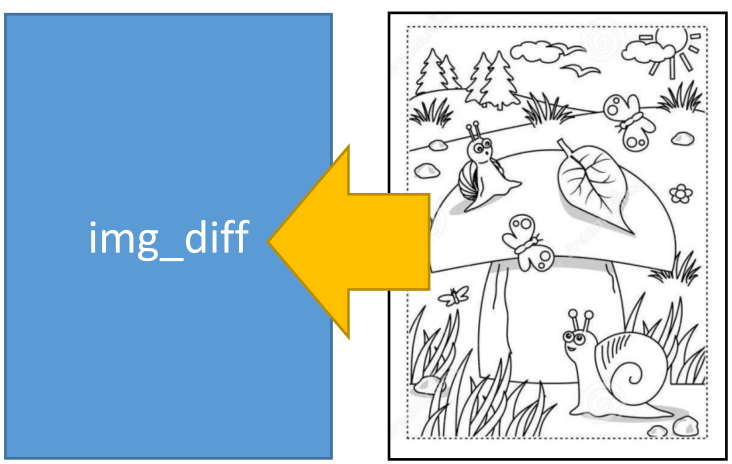
See the difference?
→ Image subtraction (Part 1)

size

The basic idea is to see the differences by subtracting the original template with a proper image containing the subimage properly sized and shifted

img_delta = template – img_diff

```
% copy the original image  (**just for the EXTERNAL FRAME**)
img_diff = img_template_gray ;
```



Template

This is just to
1) Get the proper dimensions since img_diff must be subtracted
2) Copy the borders since the subimage is missing the borders to have a more readable comparison

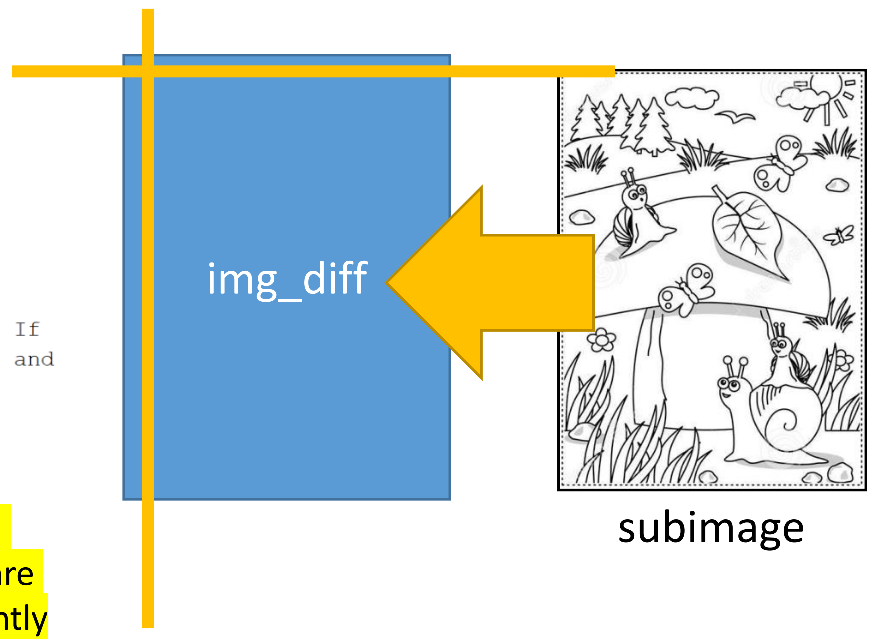
See the difference? → Image subtraction (Part 2)

The proper shift

```
% Place the smaller image inside the larger image. Rotate the smaller image
% to comply with the convention that MATLAB® uses to display images.

img_diff(ij:-1:ij-size(img_subimage_gray,1)+1,ji:-1:ji-size(img_subimage_gray,2)+1) = ...
    rot90(img_subimage_gray,2) ;
```

This notation is not in the exam!

```
% find the maximum
[ssr,snd] = max( crr(:) );
[ij,ji] = ind2sub(size(crr),snd);
```



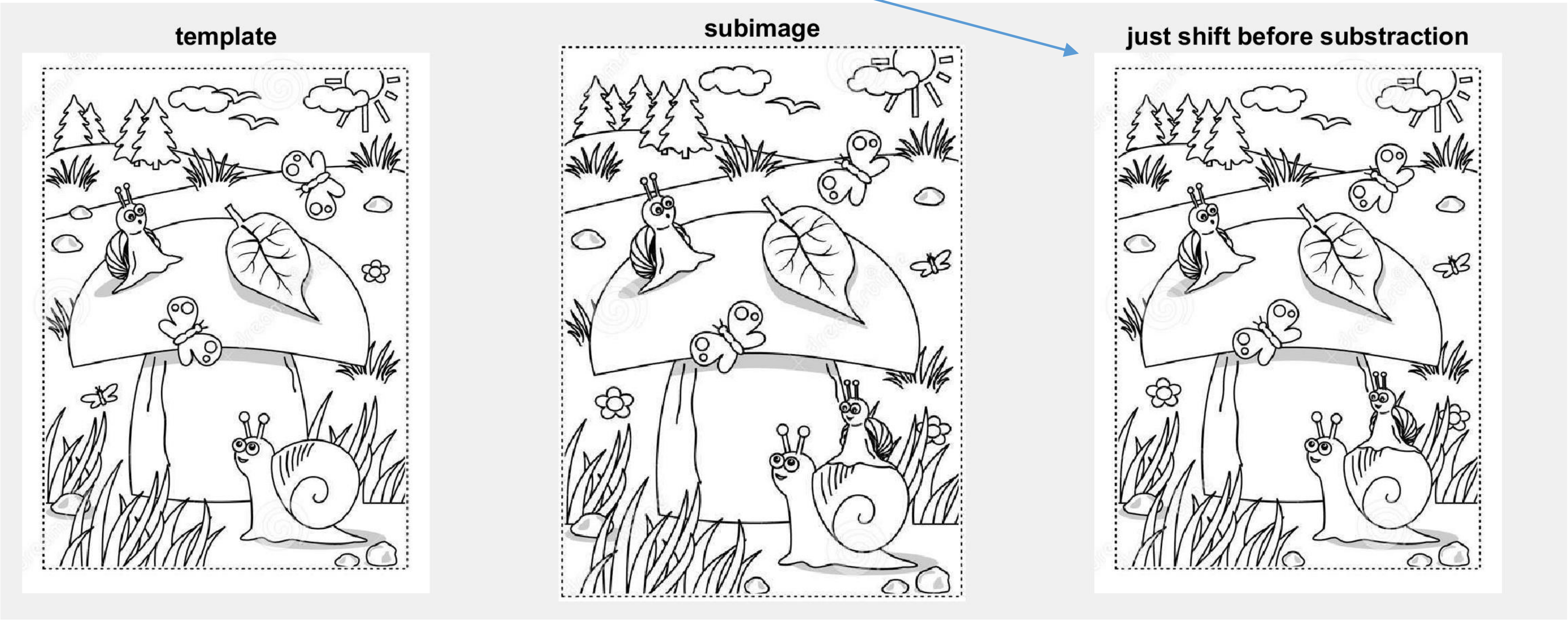

```
rot90  Rotate array 90 degrees.
   B = rot90(A) is the 90 degree counterclockwise rotation of matrix A. If
   A is an N-D array, rot90(A) rotates in the plane formed by the first and
   second dimensions.

   rot90(A,K) is the K*90 degree rotation of A, K = +-1,+-2,...

   Example,
       A = [1 2 3      B = rot90(A) = [ 3 6
            4 5 6 ]                     2 5
                                        1 4 ]
```

Memory and visualizations are ordered differently

See the difference? → Image subtraction (check)

Open the previous figure by using the handle

```
figure(h1)
subplot(1,4,3)
imshow( img_diff )
title('just shift before substraction')
```

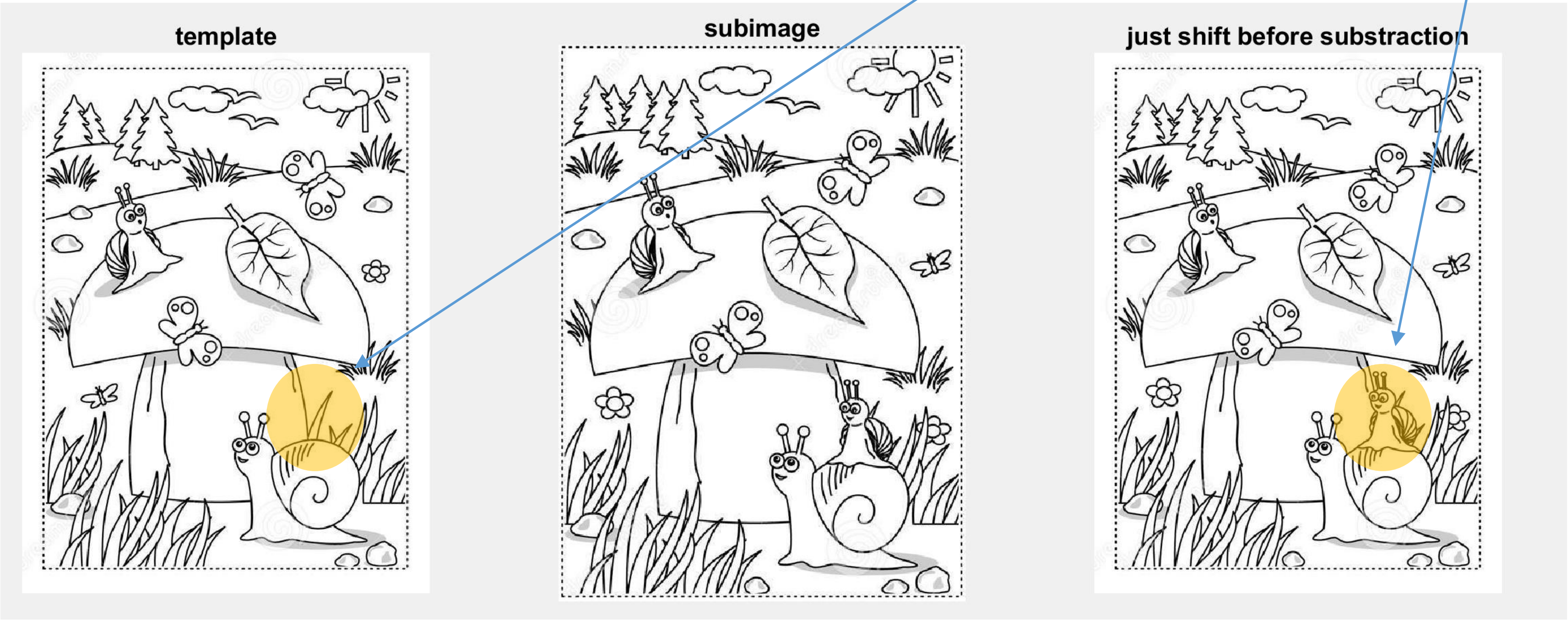


See the difference?
→ Image subtraction (check 1)

Open the previous figure by using the handle

```
figure(h1)
subplot(1,4,3)
imshow( img_diff )
title('just shift before substraction')
```

OK We have the subimage inside **img_diff**



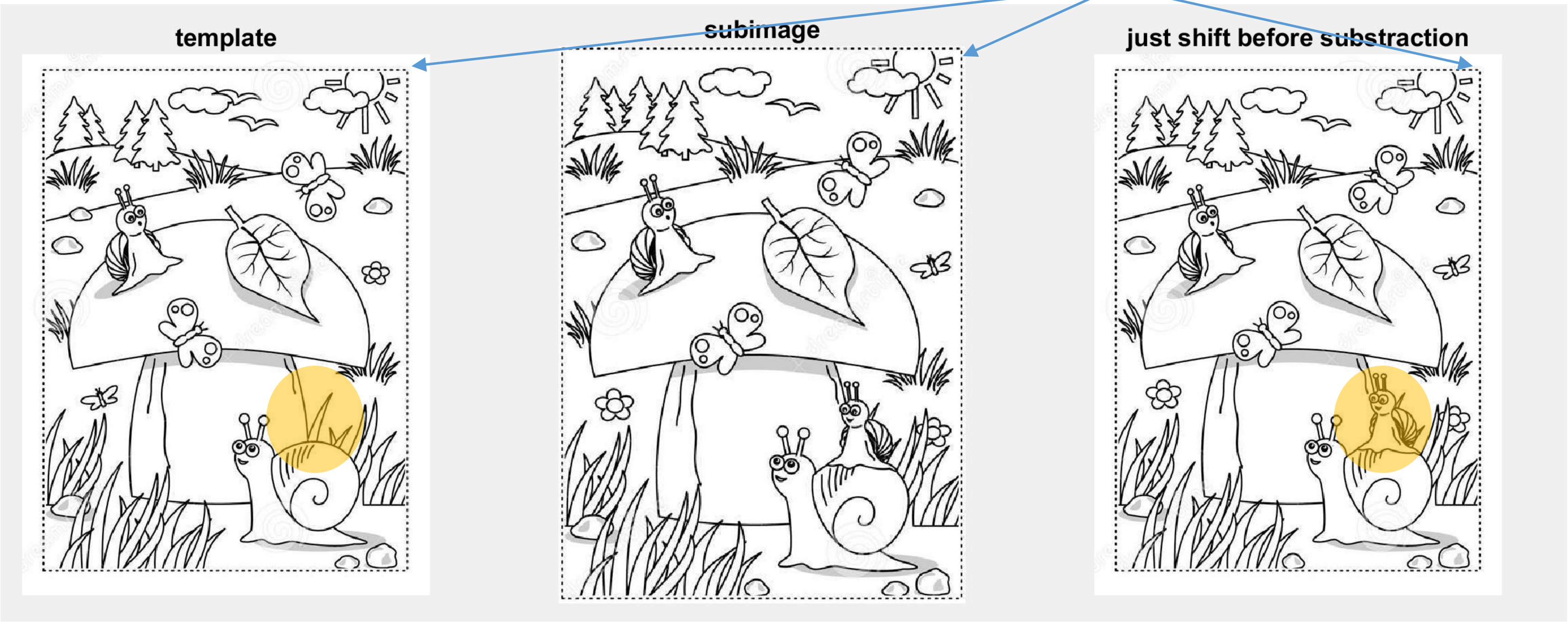
See the difference?
→ Image subtraction (check 2)

Open the previous figure by using the handle

```
figure(h1)
subplot(1,4,3)
imshow( img_diff )
title('just shift before substraction')
```

OK We have the subimage inside

OK With the correct shifts



Creating the delta image

We need a double image not uint8!!

We need to be ready to store negative values in the delta!

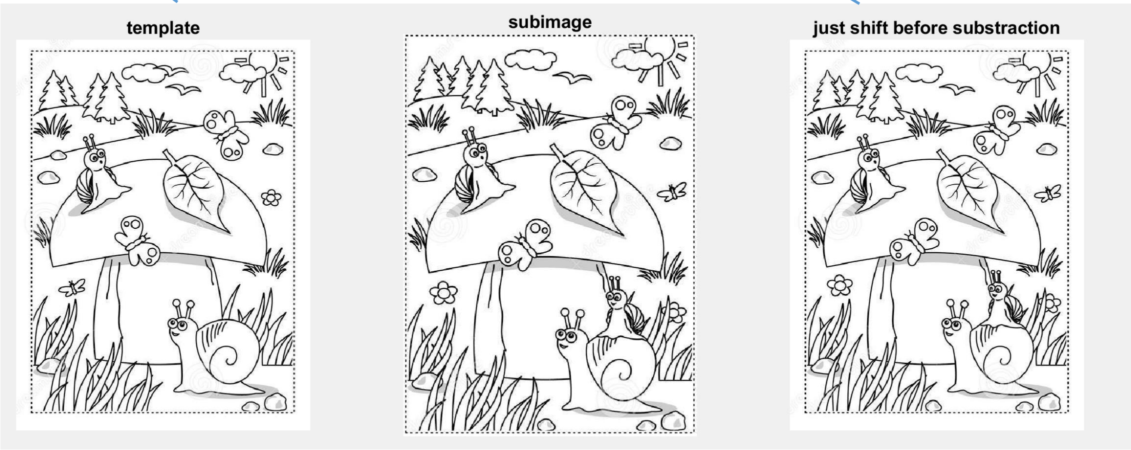
```
% img_diff2 = the template - the shifted subimage (with just the frame of template)
% we use double since we need negative values (not allowed in the uint8 format
img_delta = double(zeros(size(img_template_gray)));
img_delta = double(img_template_gray) - double(img_diff);
```

Creating the delta image (2)

We need to be ready to store negative values in the delta!

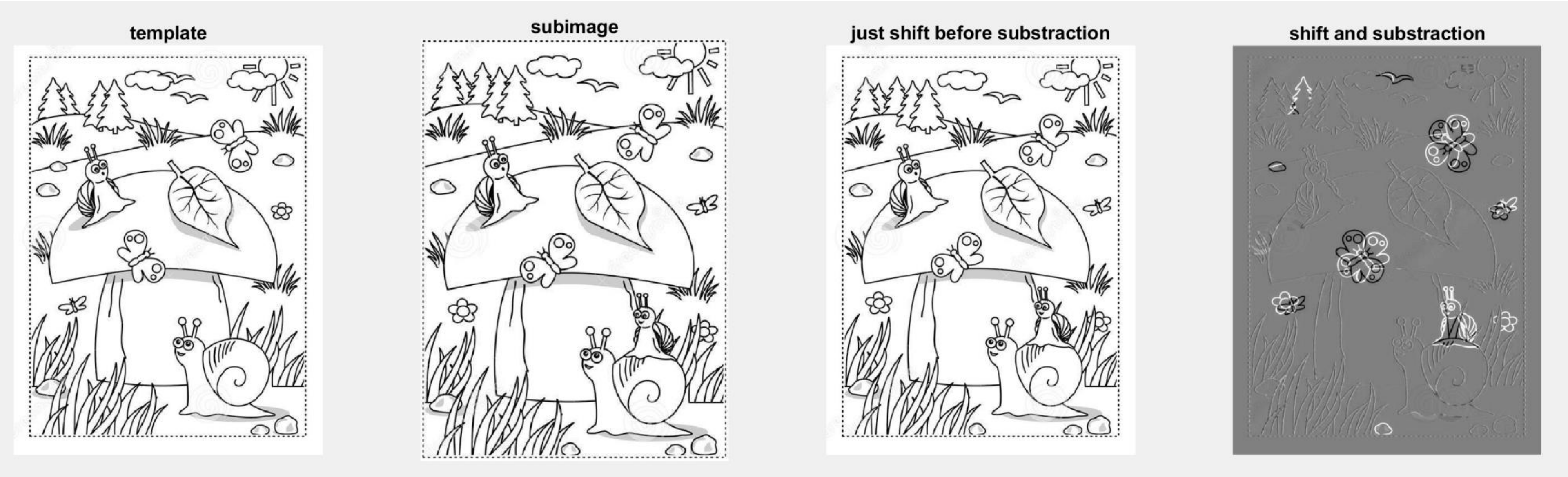
```
% img_diff2 = the template - the shifted subimage (with just the frame of template)
% we use double since we need negative values (not allowed in the uint8 format
img_delta = double(zeros(size(img_template_gray)));
img_delta = double(img_template_gray) - double(img_diff);
```

Make a «cast»! To avoid any conversion problems!

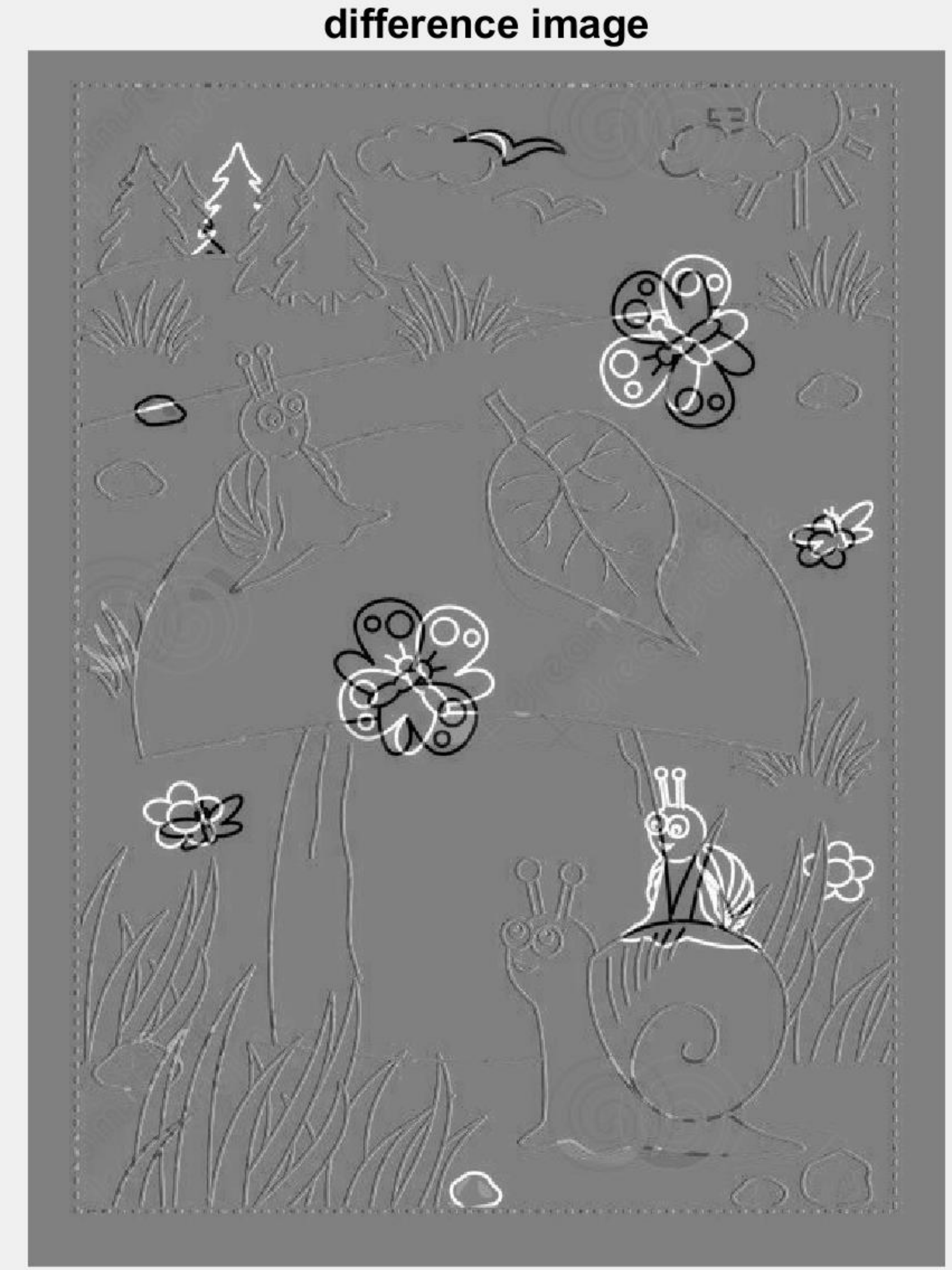


Checking the delta image

```
% figure(h1)
subplot(1,4,4)
imshow( img_delta, [] )
title('shift and substraction')
```



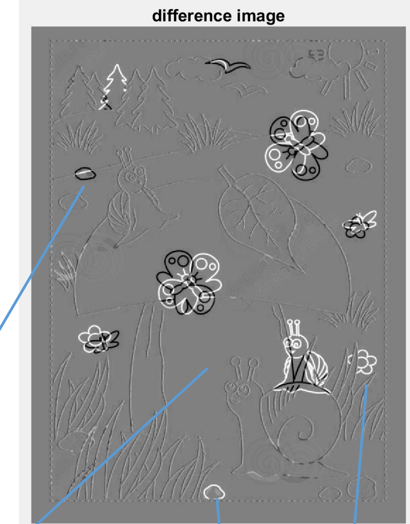
Checking the delta image



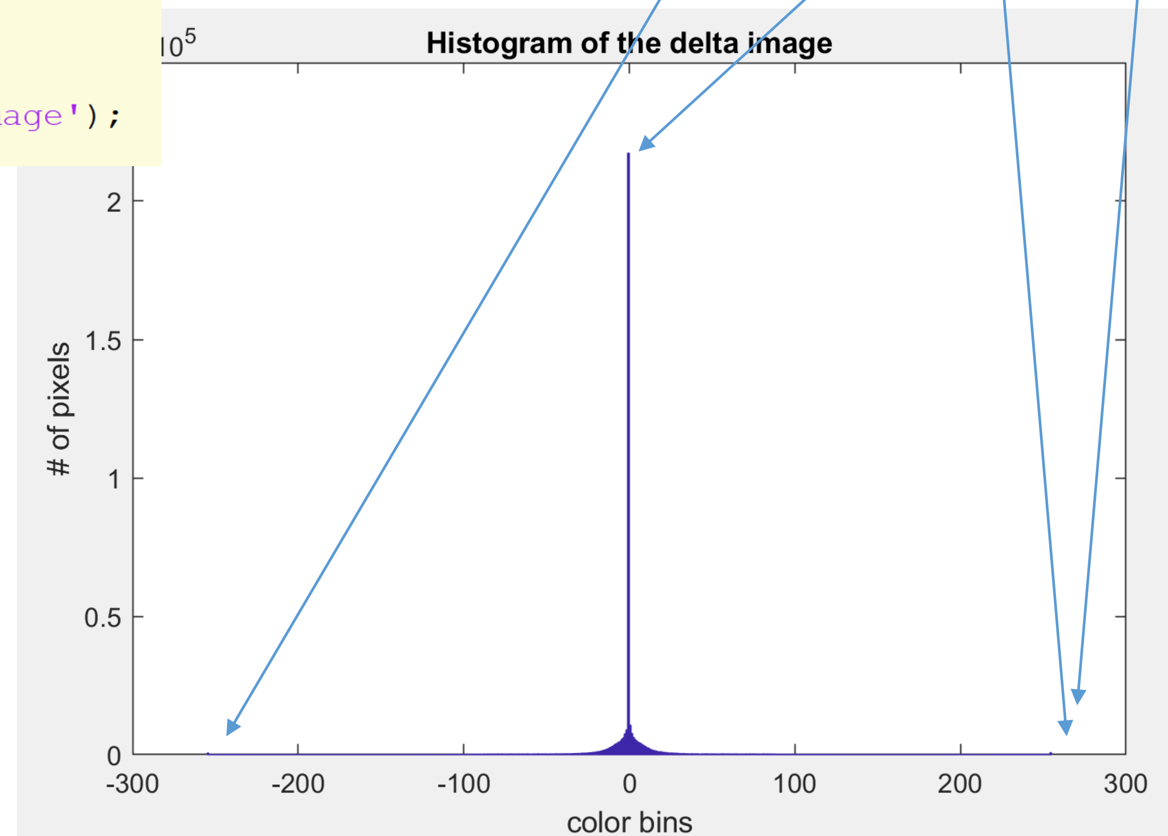
Improving the visualization

```
%% improving the visualization
figure
subplot(1,2,1)
hist( img_delta(:), [2*255]);
xlabel('color bins')
ylabel('# of pixels')
title('Histogram of the delta image');
```

Zero values are grey
(histogram stretching)



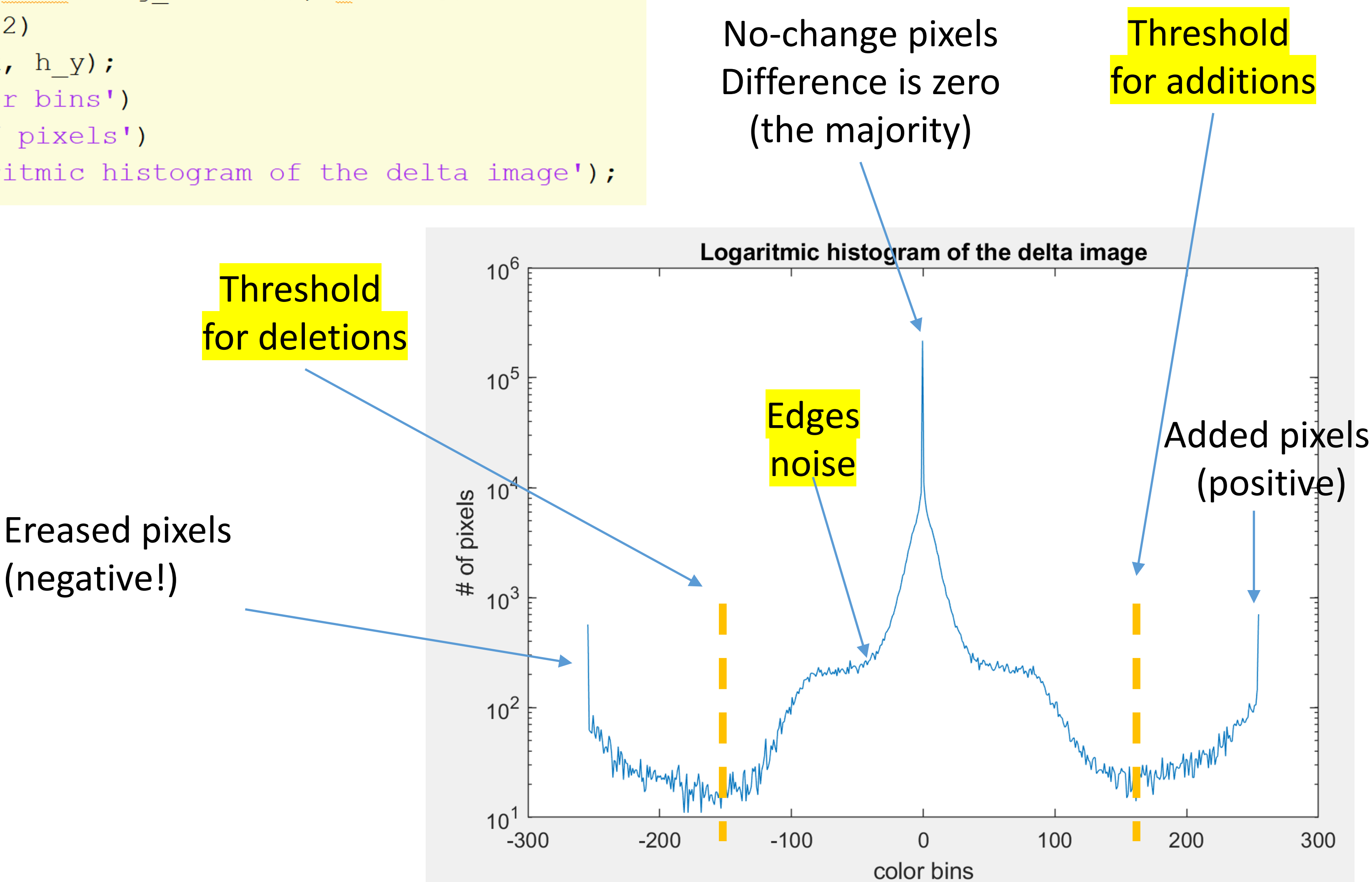
Understanging how to use the image histogram in your work!



Sometimes histograms are too “flat” to really understand what is happening

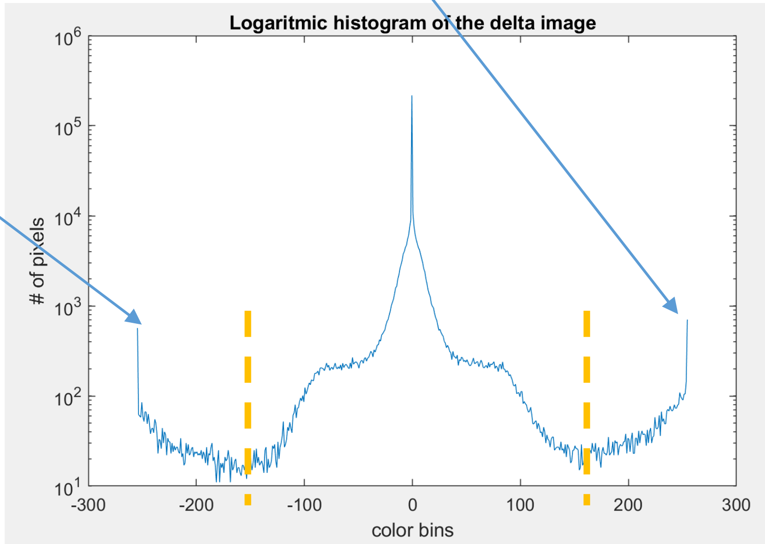
Improving the visualization

```
subplot(1,2,2)
[h_y, h_x]= hist( img_delta(:), [2*255]);
subplot(1,2,2)
semilogy(h_x, h_y);
xlabel('color bins')
ylabel('# of pixels')
title('Logaritmic histogram of the delta image');
```



Separating the additions and deletions

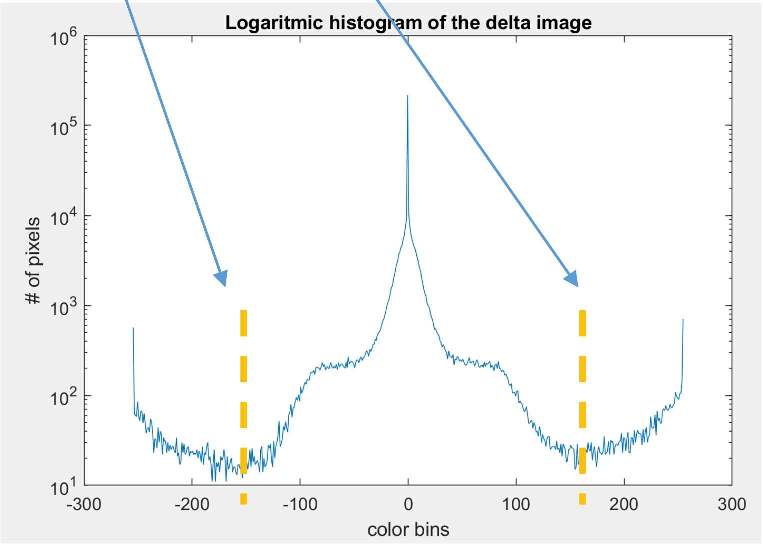
```
%% finding the good thresholds
max_value = max(img_delta(:));
min_value = min(img_delta(:));

additions = (img_delta > 0.5 * max_value ) ;
deletions = (img_delta < 0.5 * min_value );
```



Separating the additions and deletions

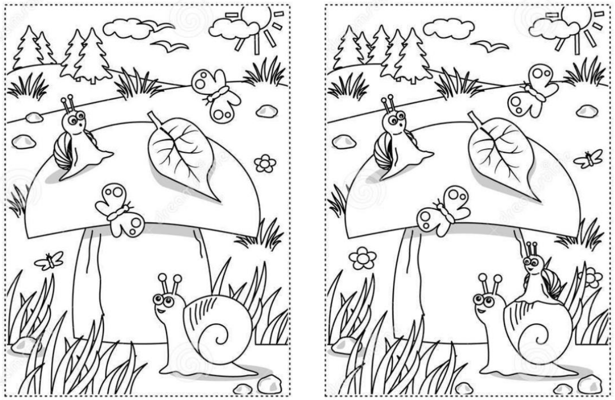
```
%% finding the good thresholds
max_value = max(img_delta(:));
min_value = min(img_delta(:));

additions = (img_delta > 0.5 * max_value ) ;
deletions = (img_delta < 0.5 * min_value );
```

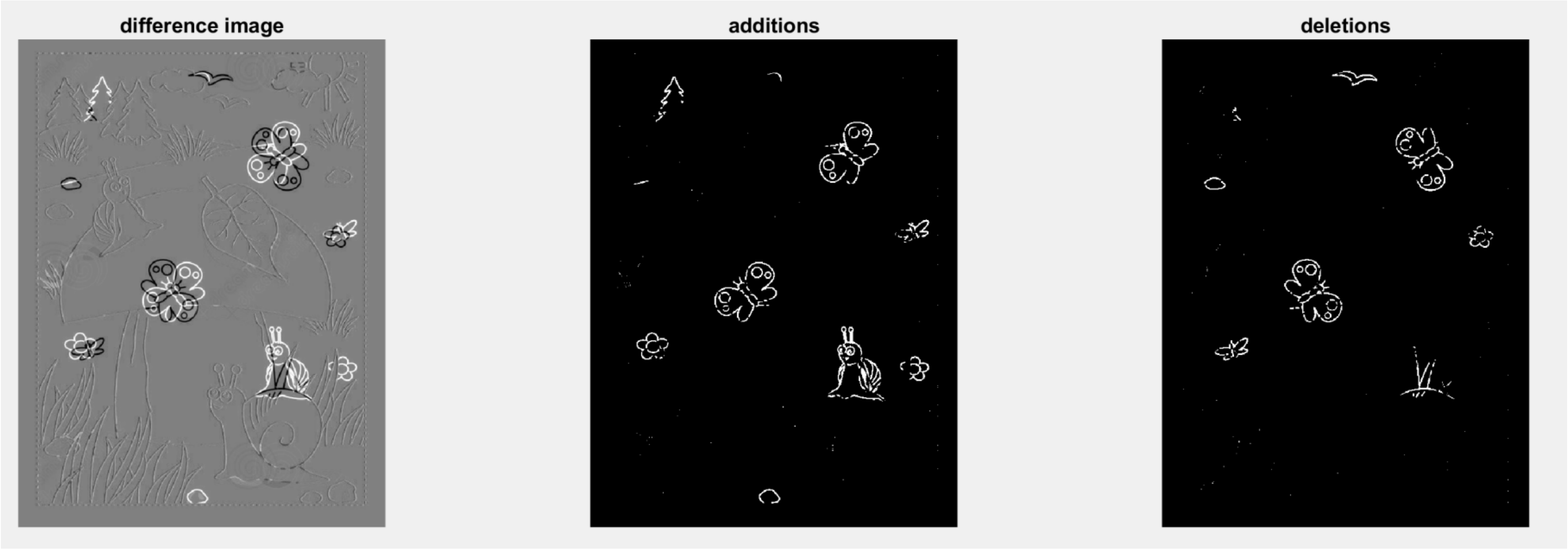
```
>> whos additions
  Name            Size          Bytes  Class

  additions       736x555       408480  logical
```



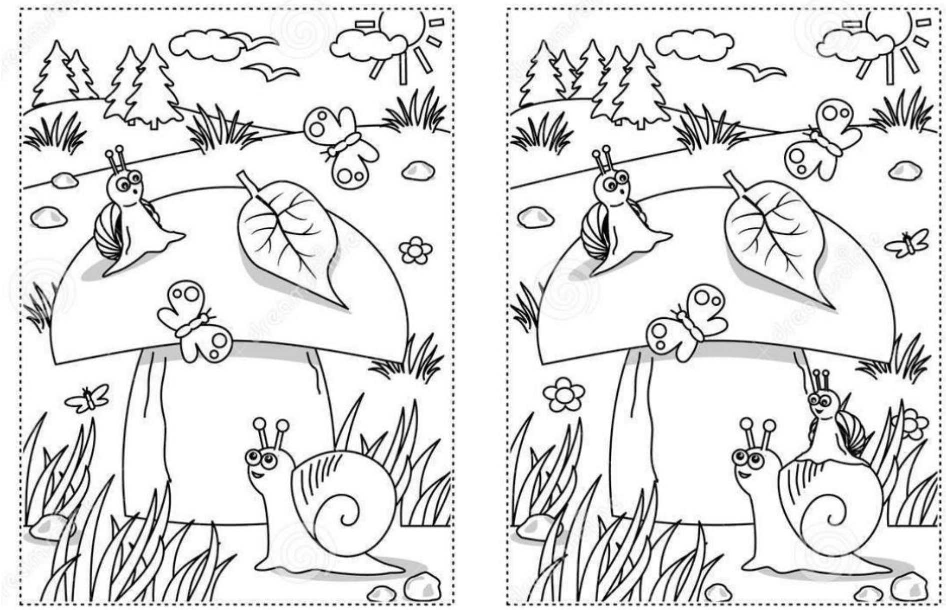
Separating the additions and deletions

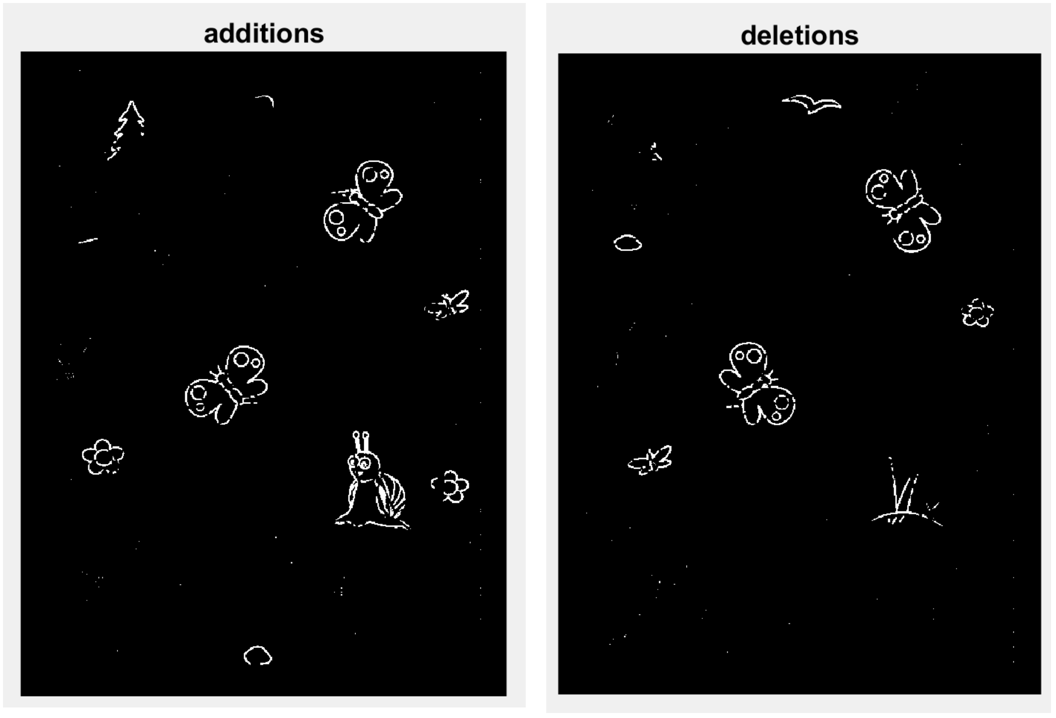
```
max_value = max(img_delta(:));
min_value = min(img_delta(:));

additions = (img_delta > 0.5 * max_value ) ;
deletions = (img_delta < 0.5 * min_value );

figure
subplot(1,3,1)
imshow( img_delta , [])
title('difference image')

subplot(1,3,2)
imshow( additions , [])
title('additions')

subplot(1,3,3)
imshow( deletions , [])
title('deletions')
```



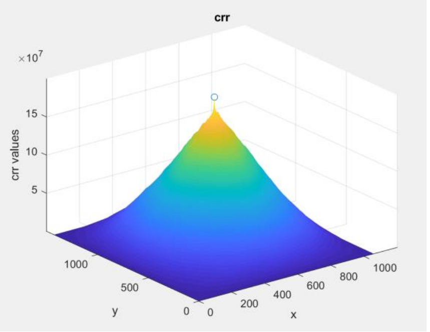


Overview





Homeworks



- How is changing the CRR if you put in input the same template image as subimage
  - What about the maximum?
  - What about the shape of the CRR?

- How is changing the CRR if you put two completely different images in input?
  - What about the maximum?
  - What about the shape of the CRR?

Main points

- Cross-Correlation
- Matlab first steps with images
- Coding an example for similarity