# Image equalization and geometric transforms

# Outline

- Equalization
  - Display the histogram of an image
  - Apply graylevel stretching to an image
  - Apply uniform equalization to an image and check results
  - Apply non uniform equalization (Contrast Limited Adaptive Histogram Equalization, CLAHE) and check results
  - **Matlab functions**: imread , rgb2gray , imhist, histeq , adapthisteq
- Geometric transforms
  - Application of a known transform to an input image
  - Estimate the transform that maps one input image into a target one
  - **Matlab functions**: maketform, imtransform, cp2tform, cpselect

# Graylevel stretching

- Load and convert to grayscale the image test-hist-01.png

  img = **rgb2gray**(**imread**('test-hist-01.png'));

- Display the image and its histogram

  **imshow**(img); figure; **imhist**(img)

- Apply graylevel stretching to the image

  imgS = **imadjust**(img, double([min(img(:)) max(img(:))]/255.0));

- Display the output image. Any comment?

- Apply graylevel stretching with 4% saturation to the image

  imgSS = **imadjust**(img,**stretchlim**(img, [0.02 0.98]));

# Equalization

- Load and convert to grayscale the images test-hist-01.png and test-hist-02.png
  - imread, rgb2gray
- Equalize the images using global equalization

  **histeq**(img)

- Equalize the images using CLAHE

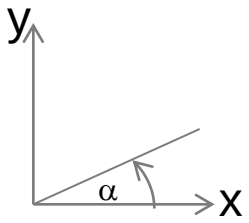
  **adapthisteq**(img, 'clipLimit',0.1, 'NumTiles',[8 8])

# Geometric transforms

- Expression of a generic affine transform (not including the projective transform)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Rotation by $\alpha$ counterclockwise about the origin




$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Uniform scaling by $\sigma$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

# Geometric transforms

- In Matlab, in order to apply a known affine geometric transform **A** to an input image imgIn and transform it into the output image imgOut the following steps should be followed:

  T = **maketform**('affine', A);

  imgOut = **imtransform**(imgIn,T);

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$


**Assignment**: read the image ferrari.jpg and rotate it by 30° about the origin

# Geometric transforms

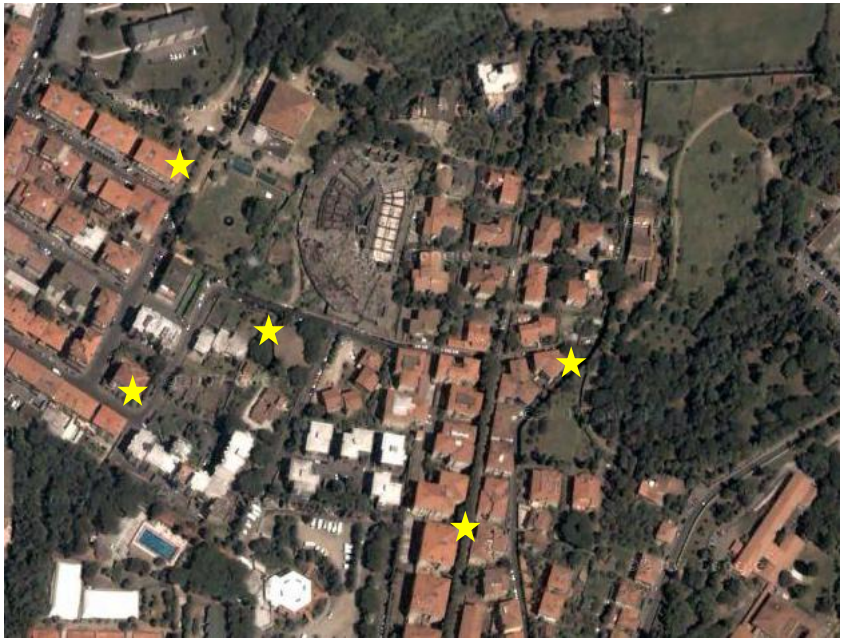
- Inverse problem: **image registration**
  - You are given an input and a target image. How to compute the values of the parameters that map the input to the target?

Input: sat-1.png

Target: sat-2.png

# Geometric transforms

- Choose a model for the transformation: e.g. affine
- Find “enough” pairs of corresponding points in the two images to constrain the values of the model parameters

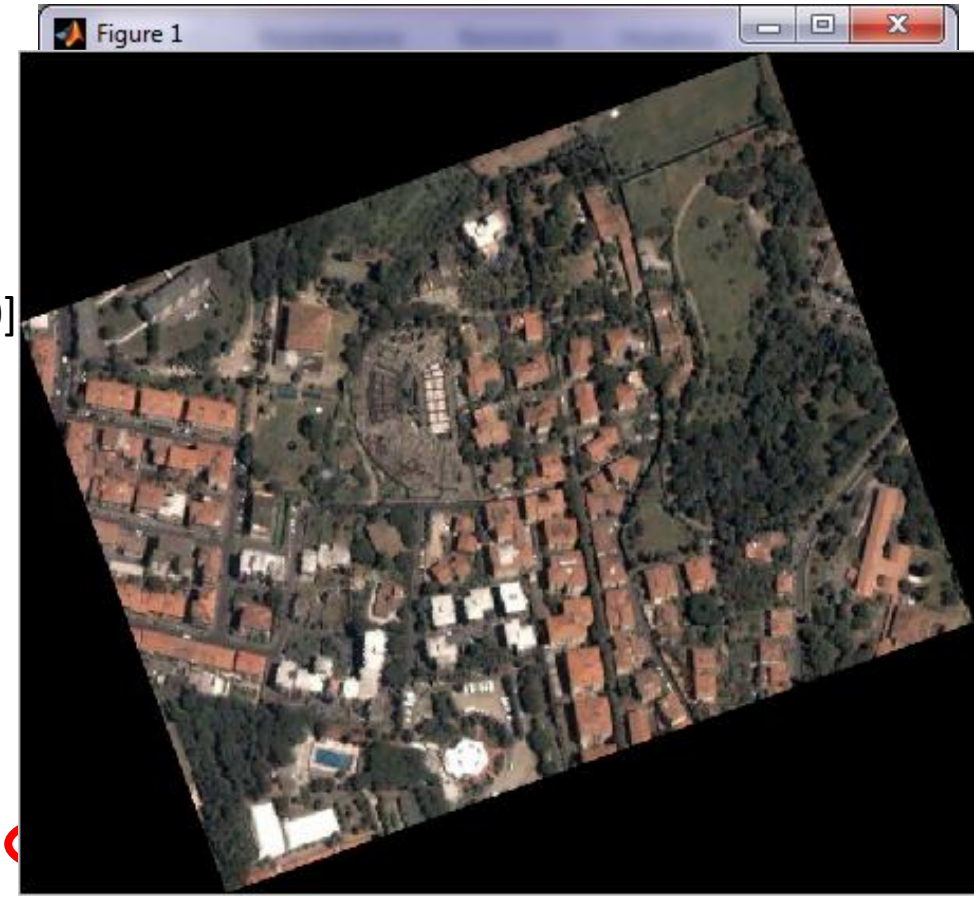


Input: sat-1.png

Target: sat-2.png

# Geometric transforms

- Given the coordinates of corresponding points (use **impixelinfo**) the matlab function **cp2tform** returns the T-structure




```
inputP=[130 135; 93 311; 342 392; 466 94]
targetP=[121 192; 146 371; 407 362; 421 40]

T = cp2tform(inputP, targetP, 'affine');

imgRegistered = imtransform(imgInput, T);

figure; imshow(imgRegistered)
```

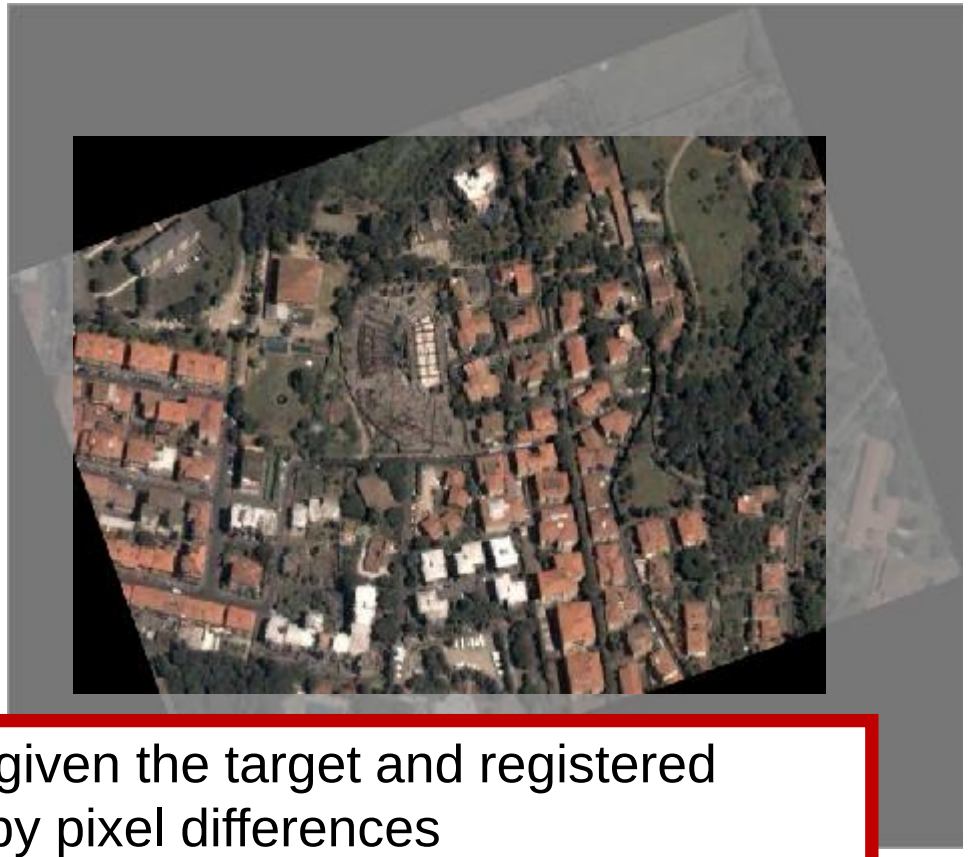
# Geometric transforms

- However, we would like to overlap the registered and target images so as to check where they are different: the coordinates of the registered image should match the rows and cols of the target

# Geometric transforms

imgRegistered = **imtransform**(imgInput, T, …

'XData',[1 size(imgTarget,2)], 'YData', [1 size(imgTarget,1)]);



**Assignment**: write a function that given the target and registered images returns a map of the pixel by pixel differences

# Geometric transforms

- Looking for an even simpler solution? Try **cpselect**

…

[inputP targetP] = **cpselect**(imgInput(:,:,1), imgTarget(:,:,1), 'Wait',true);

T = **cp2tform**(inputP, targetP, 'affine');

…