M2 – Indexation d'images

Description et matching de deux images

Grayscale histogram

Open **histogram.py**

1. Load 'lena.jpg'
2. Compute and draw the L2 normalized grayscale histogram
3. Draw on the same fig the normalized grayscale histograms of 'box.png'
4. Compute L2 distance between normalized grayscale histograms of 'lena' and 'box'
5. Redo between 'lena' and 'messigray'
6. Redo with color histograms.

Gradient computation

Open **gradient.py**

1. Load 'rectangle.png'
2. Compute and show gradient in X and Y
3. Compute and show gradient magnitude and angle
4. Compute and show gradient orientation histograms
    - Without quantification
    - With 8 bins quantification
5. Rotate the image (45 degrees) and compute gradient orientation histograms
6. Try another image

Feature detection (Harris corner detection)

Open **feature_detection.py**

1. Compute and show Harris corners on 'chessboard' image
2. Try on 'Lena'
3. Change the parameters

Feature description (SIFT)

Open **feature_description.py**

- Compute and show SIFT descriptors
- Try with different images.

- What could you say about : the number of keypoints ? Their localisation ?

Feature matching

Open **matcher_ocv3.py**

- Compute and show matches between 'box' and 'box_in_scene'
- Between 'box' and 'lena'
- Try to invert the 2 images

What does that mean that 2 descriptors are matched ?
How the matches are computed ? What is displayed on the screen ?

What is the number of matches regarding the number of descriptors ?
How the number of matches vary according to images content ?

- Take an image and create different modifications using gimp or any image editor (rotate, crop, blur, compress, mask some part of the image, etc.)
- Compute the matches between the original image and its modifications

How does the number of matches behave according to the different modifications ?

DB Search

Open **flann.py**

Given a query descriptor (vector), search for the nearest descriptors in the DB.

- Compare in term of speed and accuracy a brute-force approach with an indexing scheme
- What is the influence of :
  - The DB size (number of descriptors) ?
  - The descriptor size (dimension of each descriptor) ?
- What happened when the number of query descriptors grow?

Curse of dimensionality

Open **flann.py**

For a query vector, using a brute-force approach, print for increasing vector dimensions:

- The ratio between the distance to its nearest neighbor and its second nearest neighbor in the database
- The ratio between the distance to its nearest neighbor and its farthest neighbor in the database

What could you say about distances between vectors when the dimension increase?