Introduction to Image and Video Processing

due June 9 15:59

Alexia Briassouli (alexia.briassouli@maastrichtuniversity.nl)

Spring 2023

Instructions:

Each of these projects will count for 1/2 of your final grade. They will be checked for plagiarism (software + text). **You are requested to hand in one zip file with title "YourLastName_project1" (e.g. John Doe should submit "Doe_project1.zip").** The zip file should contain:

1. An about 10 page report with your answers to the questions and figures with your results. Page limit is flexible, content matters.
2. The code for producing these results *with clear comments in the code!*. You can use any programming language you are comfortable with (preferably Python or Matlab). You should explain what you think are important parts of the code in the report (e.g. if you use a special trick that you are proud of).
3. You should include references and starter code links that you used (see lecture 1 slides for details).

Your grade will depend on how clearly you present and explain your results in the report and code. You are allowed some freedom to explore solutions (e.g. if you are asked to come up with your own method), so *there is no one correct answer*. However, you should demonstrate you have understood the class material and how it applies to these projects.

1 Human perception




1. In this exercise, you will demonstrate and witness properties of the human visual system through a selective frequency image filtering procedure, described below. You will use two different images of the same size to create *one* new image by adding a low-pass version of the first one to a high-pass filtered version of the second. You should use image pairs that are provided with this exercise like the two on the left or the two on the right (in the folder, face11.jpg, face12.jpg are one pair, face31.jpg, face32.jpg another). These images have been aligned so the exercise result looks interesting, so if you use your own make sure faces or significant features are aligned.
   (a) Apply low-pass filtering to the first image and high-pass filtering to the second in the frequency domain. Add the two filtered images and display the result in the spatial domain. Explain in which domain you add the two filtered images (spatial or frequency) and why it matters.

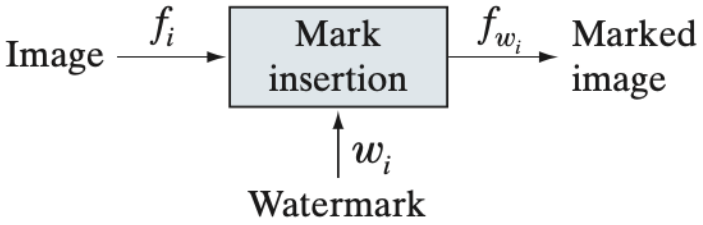
(b) Explain why in your new image (created above), one image is visually dominant over the other *as viewing distance changes*, based on the filtering you applied and properties of the human visual system (*Hint: for this, use your common sense, what you know about low-pass and high-pass filtering, your experimental results and briefly look up online how distance and perception are related to image frequency content).*

2 Hide a Secret Message in an Image - DCT domain watermarking

A watermark can be hidden in an image if it is inserted in the transform domain. To find if an image is watermarked, the watermark has to be detected in the transform domain. Here you will add and detect your own watermark in the DCT domain, on an image of your choice. This exercise is based on example 8.30 in Sec. 8.12 from Gonzalez (4^{th} ed.), so it will help to read Sec. 8.12 and example 8.30.

2.1 Watermark Insertion

Choose an image you like, to which you will insert an invisible watermark following the steps below. You can work in the color or grayscale domain.



1. Compute the 2-D DCT of the image to be watermarked in a block-wise manner. You can use 8×8 or 16×16 blocks.
2. Choose the K most important *non-DC* coefficients, $c_1, c_2, ..., c_K$ using a zig-zag scan, so that the image is not degraded too much when you reconstruct the image using K coefficients per block. You have to determine what is a good value for K after some testing. Display the image reconstructed with (1) all coefficients, (2) K coefficients ($K < 64$ for 8×8 blocks).
3. Create a watermark by generating a K-element pseudo-random sequence of numbers, $\Omega = \omega_1, \omega_2, ..., \omega_K$ that follow a Gaussian distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$.
4. Embed the watermark from Step 3 into K most important *non-DC* DCT coefficients in each block from Step 2 (i.e. the K coefficients you chose before except the DC one) using the zig-zag scan. Add the watermark using the following equation:

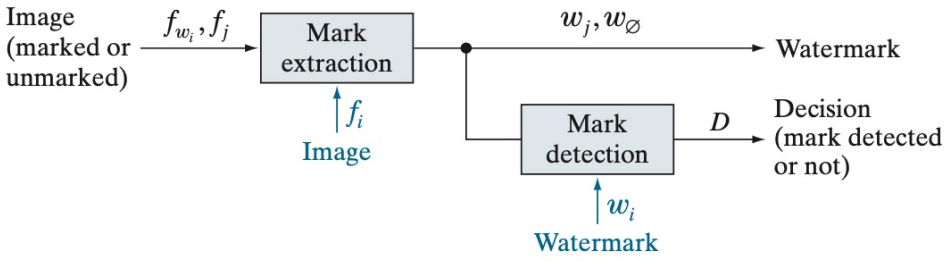
   $$c_i' = c_i \cdot (1 + \alpha\omega_i), \ 1 < i \leq K$$

   for a constant α, $\alpha > 0$ that controls the strength of the watermark. Test different values of α to find the one that does not degrade image quality (i.e. the watermark is hidden) but is non-zero.
5. Replace the K non-DC DCT coefficients c_i with c_i'.
6. Compute the inverse DCT of the DCT with the new coefficients c_i' to obtain the watermarked image. Display (1) the original, (2) the watermarked images and (3) their difference image. Discuss the appearance of the watermarked and difference images, in relation to the steps you carried out above.
7. Display the histogram of the difference image (between the original and watermarked images) and discuss its appearance, also in relation to the steps you carried out above.

2.2 Watermark Detection

Now you will detect if a mystery image is watermarked for two cases:

1. Mystery image 1: Create a new watermarked image repeating the procedure above, using the same Gaussian distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$. Use the same α.
2. Mystery image 2: Create a new watermarked image using the same original image and a new watermark. To do this, repeat the procedure above, using a different Gaussian distribution $\hat{\Omega}$ with mean $\mu = 0$ and a different variance σ^2 of your choice. Use the same α.



For each mystery image:

1. Compute the 2D blockwise DCT of the mystery image.
2. Keep its K most important (zig-zag scanned) non-DC DCT coefficients, now denoted as $\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_K$.
3. Estimate an approximation of the watermark in your mystery image:

$$\hat{\omega}_i = \frac{\hat{c}_i - c_i}{\alpha c_i}, \ 1 \leq i \leq K \tag{1}$$

   where $\hat{c}_i, c_i, \alpha$ are known from above.

4. Measure the similarity of $\hat{\omega}_i$ with ω_i using the correlation coefficient.
   - You can approximate the correlation coefficient using the equation from the book:

$$\gamma = \frac{\sum_{i=1}^{K}(\hat{\omega}_i - \bar{\hat{\omega}})(\omega_i - \bar{\omega})}{\sqrt{\sum_{i=1}^{K}(\hat{\omega}_i - \bar{\hat{\omega}})^2 \sum_{i=1}^{K}(\omega_i - \bar{\omega})^2}}, \ 1 \leq i \leq K$$

     where $\bar{\omega}$ is the mean of the watermark sequence $\omega_1, \omega_2, \ldots, \omega_K$ and $\bar{\hat{\omega}}$ the mean of the approximated watermark sequence $\hat{\omega}_1, \hat{\omega}_2, \ldots, \hat{\omega}_K$.

   - *For full points*, calculate the correlation coefficient using the general definition:

$$\gamma = \frac{E[\Omega\hat{\Omega}]}{\sigma_\Omega \sigma_{\hat{\Omega}}}$$

     where you should generate several versions of the watermarked images (with the 2 different watermarks based on Ω and $\hat{\Omega}$) in order to estimate γ.

5. Compare the measured similarity γ with a threshold T of your choice, to make the decision:

$$D = \begin{cases} 1, & \text{if } \gamma \geq T \\ 0, & \text{else} \end{cases}$$

   If $D = 1$, an invisible watermark $\omega_1, \omega_2, \ldots, \omega_K$ is present in your mystery image. There is no watermark if $D = 0$. Show and discuss your results, including how γ changes.

3 PCA - Recognition

In this exercise you will use PCA for face recognition. Use N samples of 10 different faces from one of the following datasets, where the faces are centered and aligned (each face is in the same position in the images you will use).

Make your dataset **unbalanced**, i.e. pick most faces with (for example) beards/glasses/same skin color/hairdo, and fewer without those features.

Set N equal to 100 (it can be lower if your code is slow). Use images of a manageable size or resize the ones you choose to a manageable size.

Datasets you can use:

1. Eigenfaces gray faces: Zip file"dataset lfwcrop_grey.zip" in this link
2. Eigenfaces kaggle dataset
3. CelebA dataset
4. Worldface dataset
5. More datasets
6. A few more datasets

Figure 2: Sample faces from Worldface

1. Find the eigenfaces for the 10 faces using their N variations as samples of each image. Explain step by step how PCA is implemented in your code. Show the resulting eigenfaces and explain their appearance in relation to your dataset. *For full points, calculate the eigenvectors, eigenvalues to find the principle components. If you use PCACompute you still get points, but not full points.*
2. Reconstruct 2 faces: one that belongs to the most represented group in your data and one that belongs to the under-represented group. Use (1) all eigenfaces and (2) 2 (or another small number of) eigenfaces. Display and discuss the results.
3. Find a way (also from online sources) to calculate the variance of captured by (1) all eigenfaces and (2) 2 or the smaller number of eigenfaces that you chose. Display and explain your results.
4. Reconstruct each of these 2 faces using 'wrong' PCA weights, chosen by you to emphasize some features and not others. Display and explain your results.

4 Motion Energy Images

Humans can easily and quickly recognize actions from low resolution information of the motion in a video. One way that has been used for representing videos of one activity recorded by a static camera is using Motion Energy Images (MEIs). MEIs are binary templates that capture the "shape" of the activity over a certain time period (number of video frames). You can find additional information + images' source here: Motion Energy Images - The Recognition of Human Movement Using Temporal Templates

We consider part of the video, made up of w frames, during which an activity takes place (like sit down, walk several steps, run...). **Motion Energy Images** are then defined as the pixel regions of the frame where there was motion:

$$MEI(x, y, t) = \begin{cases} 1 \text{ if there is motion in pixel (x,y) over frames } t - w + 1 \text{ to } t \\ 0 \ else \end{cases} \tag{2}$$

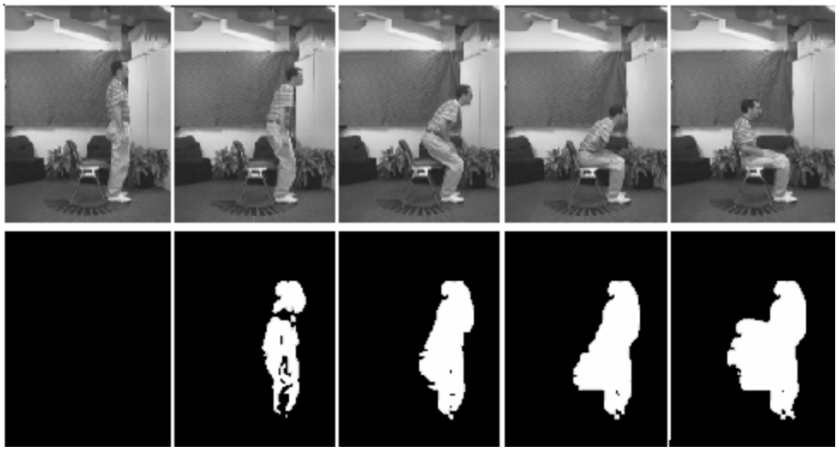


Figure 3: Examples of MEIs for different video subsequences of a "Sitting" video

Choose **two** videos from **one** of the datasets below. Make sure the videos are recorded from a static camera and contain one person carrying out one activity.

- KTH actions dataset
- Weizmann actions dataset
- Diving dataset
- AIST Dance dataset
- AIST++ Dance dataset
- Let's Dance: Learning From Online Dance Videos
- Everybody Dance Now
- Dance

1. Choose two "interesting" video sub-sequences during which a characteristic part of the activity takes place for each video by setting w to an appropriate value (e.g. 10, 20 frames). Calculate the optical flow during these 2 video subsequences using inbuilt functions from opencv. Explain how it works and display and discuss characteristic optical flow estimates for 2 different values of w. *Note: Calculate the optical flow only for the set of video frames you are examining, for reduced computational cost.*

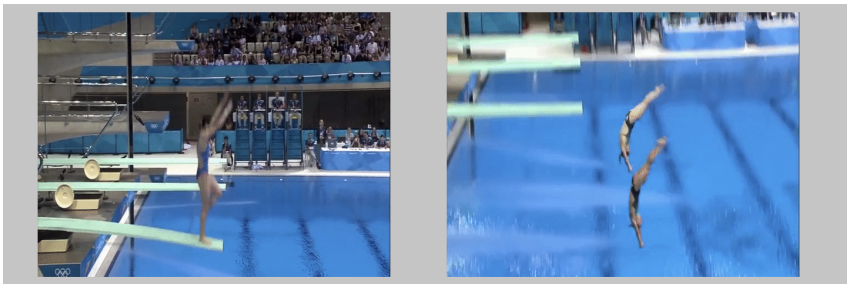


Figure 4: Diving dataset

2. Find the MEIs in these two sub-sequence of w frames. Choose w to be a reasonable temporal window that captures a characteristic part of the activity (there is no one correct value for w). Display the resulting MEIs for two different subsequences of w frames. Use what you consider are reasonable values of w, and explain why you chose them.

- Clean up the MEIs (binary images) with one or more morphological operations of your choice. Explain why you choose these operations. Display and discuss the results.
- Find the outline of the MEI using a method of your liking. Display it.
- Extract the shape descriptor for the MEI outlines of the actions using Fourier descriptors or Hu or Zernike moments, using ready-made functions. *You do not need to write this code from scratch, or explain it (as it's not the focus of this project), just use it as a tool to extract the shape descriptor.* Some suggestions for code - if you use them, make sure to refer to the author according to their license:
  - Fourier descriptors
  - Zernike in Python
  - OpenCV Hu moments in Python
- Do a simple comparison of the shape descriptors you found between the three actions (e.g. by finding the Mean Squared Error or any other difference measure between the Hu descriptors of the three MEIs etc). Show and discuss your results.

THE END

And now that you're done with the project, you can sit back and enjoy this video that shows how AI is modeling dances today! How TikTok dances trained an AI to see: