Mini-Project 1: “Où est Charlie ?“

Barbara Jobstmann

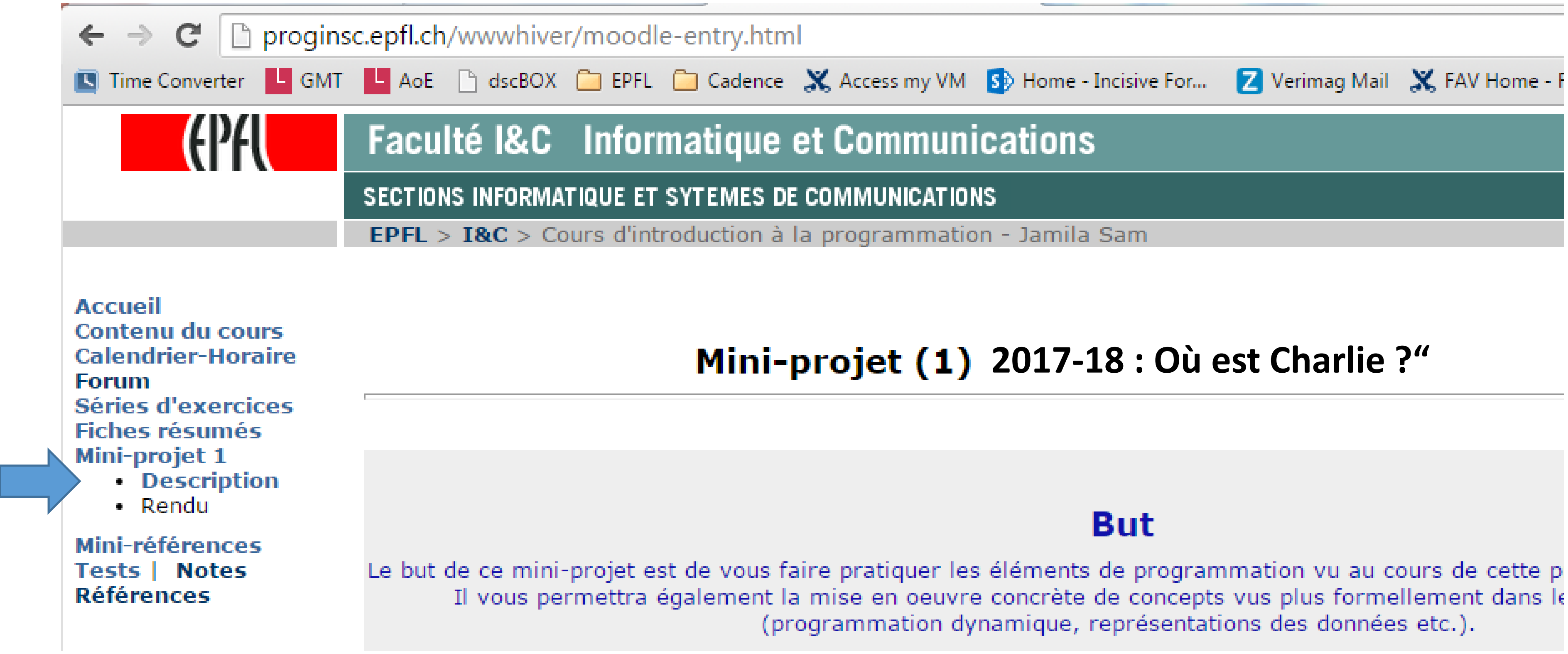
26.10.2017

Outline

- Administrative
  - Information/Starting point
  - Submission and Groups
  - Submission Server and Tokens
- Project
  - Goal
  - Overview
  - Provided Code
  - Project Details:
    - Representation of Images
    - Part 1: Image Processing “Traitement d’images”
    - Part 2: Distance of Images “Distance entre images”
    - Part 3: Localization “Localisation du motif”
    - Part 4: Cross-Correlation “Corrélation croisée”

Information about the Project

- Detailed project description and provided material: under "Project 1" ⟶ "Description" at http://proginsc.epfl.ch/wwwhiver/moodle-entry.html



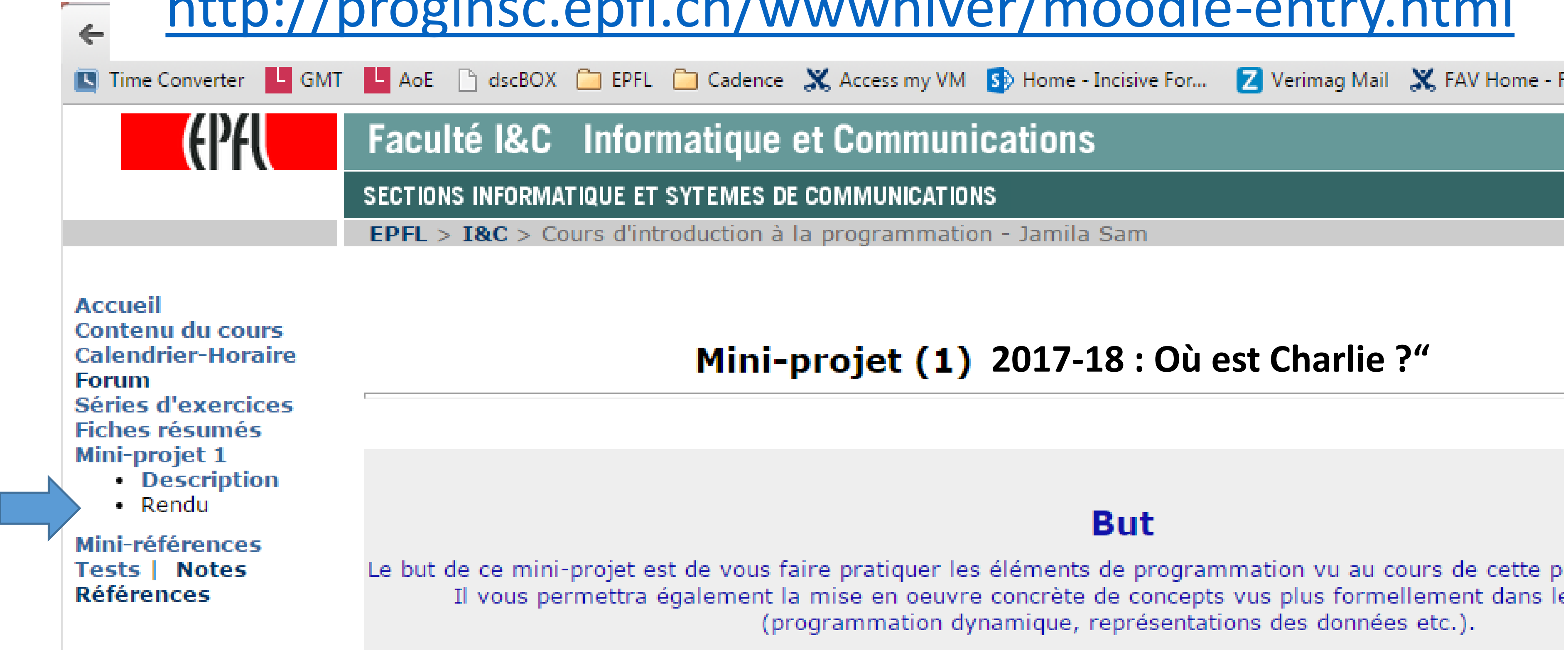
Mini-projet (1) 2017-18 : Où est Charlie ?"

But

Le but de ce mini-projet est de vous faire pratiquer les éléments de programmation vu au cours de cette p
Il vous permettra également la mise en oeuvre concrète de concepts vus plus formellement dans le
(programmation dynamique, représentations des données etc.).

Submission

- **Deadline: Nov 13th, 1pm**
- Groups of (at most) 2 students
- Submission: under “Project 1” → “Rendu” at http://proginsc.epfl.ch/wwwhiver/moodle-entry.html



Faculté I&C Informatique et Communications

SECTIONS INFORMATIQUE ET SYTEMES DE COMMUNICATIONS

EPFL > I&C > Cours d'introduction à la programmation - Jamila Sam

Accueil
Contenu du cours
Calendrier-Horaire
Forum
Séries d'exercices
Fiches résumés
Mini-projet 1
- Description
- Rendu

Mini-références
Tests | Notes
Références

Mini-projet (1) 2017-18 : Où est Charlie ?“

But

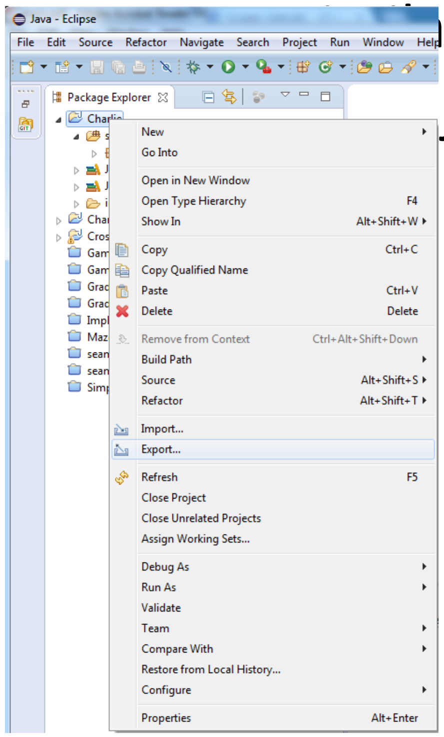
Le but de ce mini-projet est de vous faire pratiquer les éléments de programmation vu au cours de cette p
Il vous permettra également la mise en oeuvre concrète de concepts vus plus formellement dans le
(programmation dynamique, représentations des données etc.).

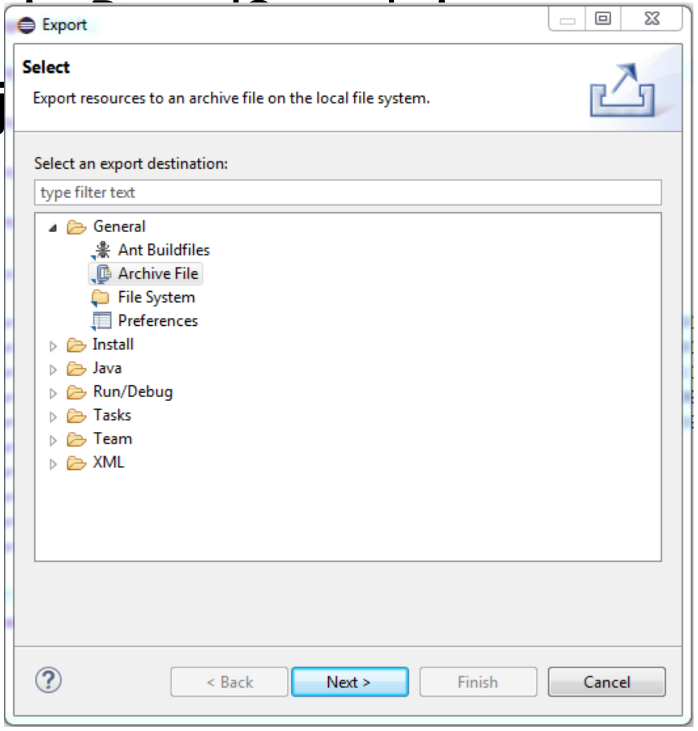
Submission Content

- Eclipse Archive file (zip-file < 20kB) that includes
  - ImageProcessing.java
  - DistanceBasedSearch.java
  - Collector.java
  - SimilarityBasedSearch.java
  - Main.java

Submission Content

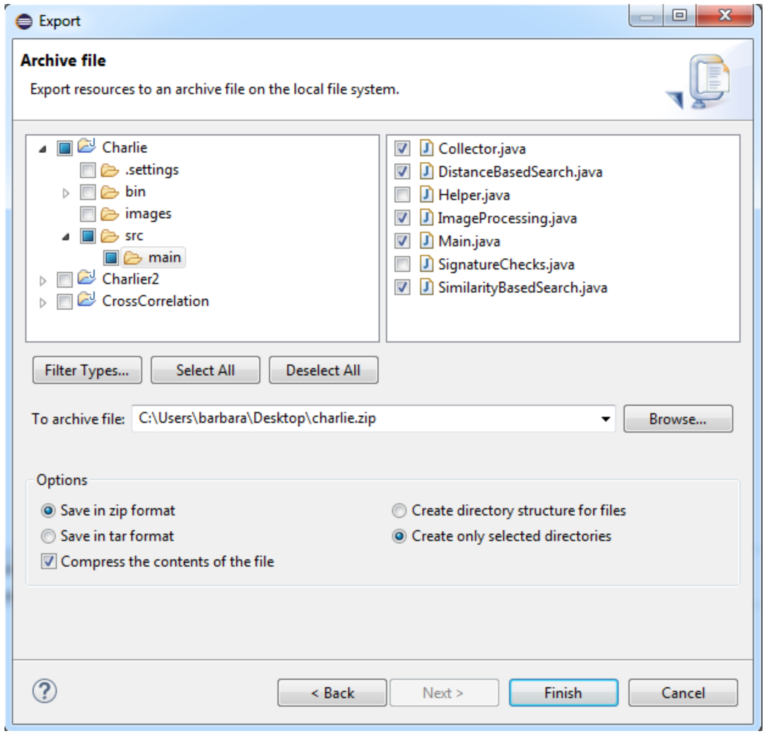
- Eclipse Archive file (zip-file < 20kB) that includes
  - ImageProcessing.java
  - DistanceBasedSearch.java
  - Collector.java





Export resources to an archive file on the local file system.

Select an export destination:



Export resources to an archive file on the local file system.

- [✓] Collector.java
- [✓] DistanceBasedSearch.java
- [ ] Helper.java
- [✓] ImageProcessing.java
- [✓] Main.java
- [ ] SignatureChecks.java
- [✓] SimilarityBasedSearch.java

To archive file: C:\Users\barbara\Desktop\charlie.zip

Options

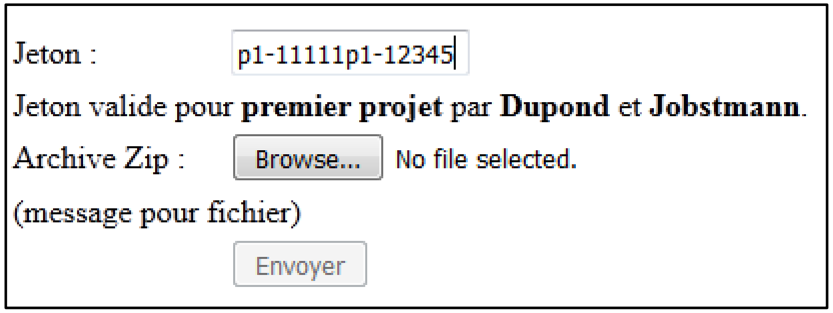
- [✓ Save in zip format]
- Save in tar format
- [✓] Compress the contents of the file
- Create directory structure for files
- [✓ Create only selected directories]

Submission Server

- Will open one week before the deadline:
  - From Mon, Nov 6th until Fri, Nov 10th 4pm.
  - **No submissions over the weekend!**
  - Reopen on Mon, Nov 13th from 9am to 1pm (strict deadline).
- Each student will need a token (specific key) to submit.
- Tokens will be send out per email one week before the submission deadline.
- Each submission required two token: one from each group member. If you work alone, you need to use your token twice.
- You can submit a new version using the same token.
- **TODO:** submit initial (incomplete) version before the deadline to get familiar with the submission process

Submission Server – Examples

- Example tokens: p1-11111 and p1-12345
- Example of submission with 2 students



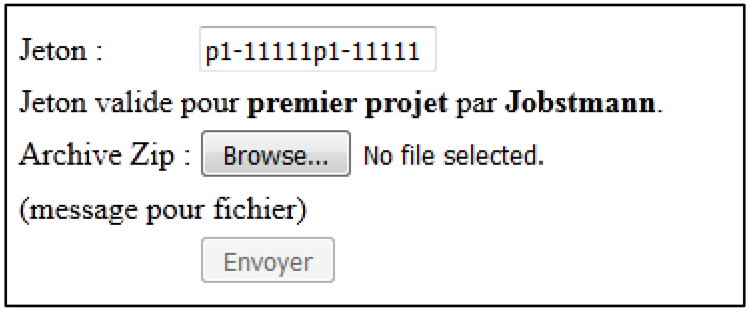
Jeton : p1-11111p1-12345

Jeton valide pour **premier projet** par **Dupond** et **Jobstmann**.

Archive Zip : Browse... No file selected.

(message pour fichier)

- Example of submission with 1 student



Jeton : p1-11111p1-11111

Jeton valide pour **premier projet** par **Jobstmann**.

Archive Zip : Browse... No file selected.

(message pour fichier)

Submission – Cheating

- The project is graded.
- The exchange of ideas between groups or with third parties is permitted and even recommended.
- **The exchange of code is strictly forbidden!**
- **Plagiarism will be controlled and will be considered cheating.**
- In case of cheating, you will receive a rating of "NA“: Art. 18 “**Fraude de l'ordonnance sur la discipline**“ https://www.admin.ch/opc/fr/classified-compilation/20041650/index.html
- Note that at anytime, you will need to be able to explain your code.

Outline

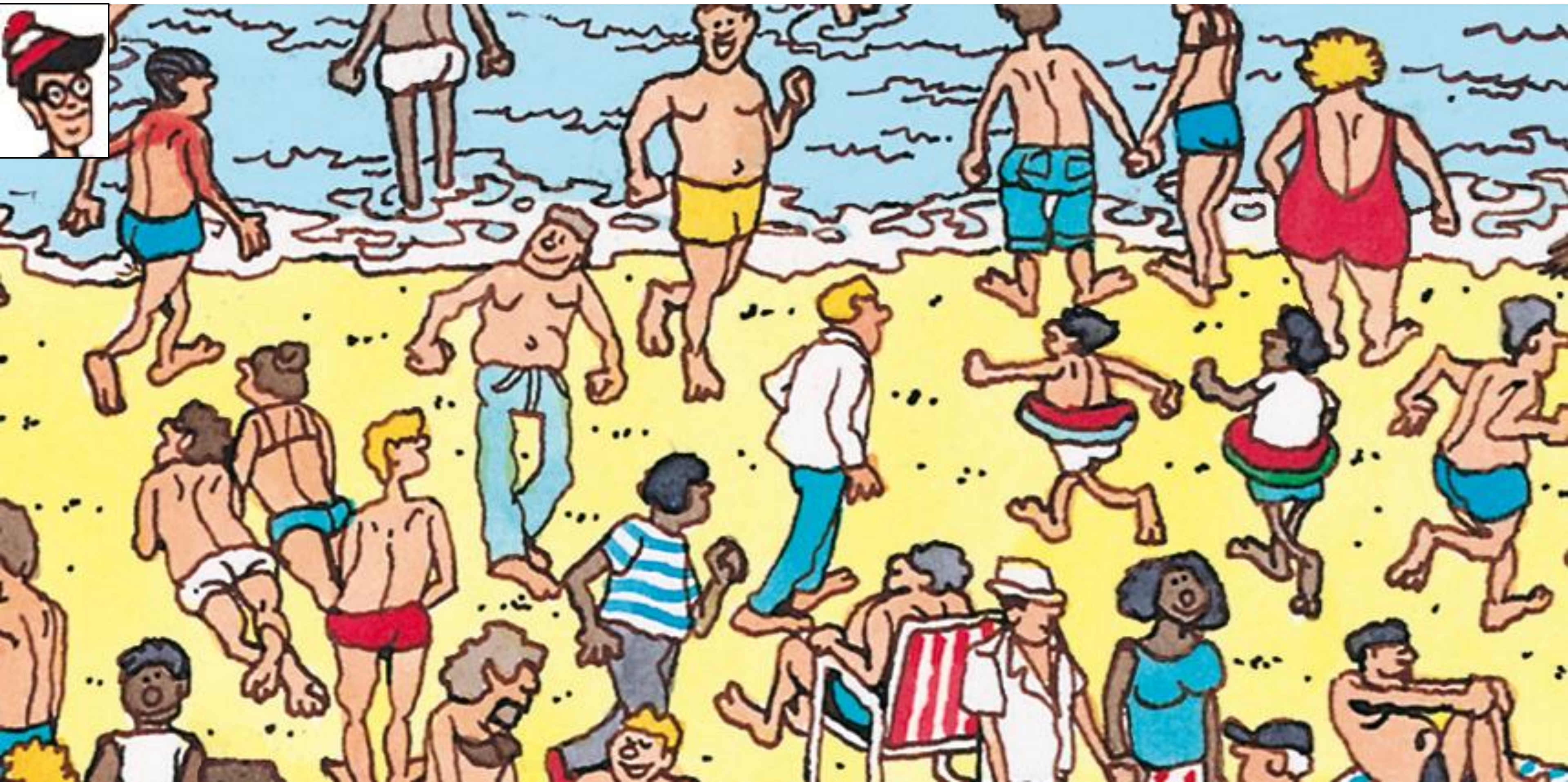
- Administrative
  - Information/Starting point
  - Submission and Groups
  - Submission Server and Tokens
- Project
  - Goal
  - Overview
  - Provided Code
  - Project Details:
    - Representation of Images
    - Part 1: Image Processing “Traitement d’images”
    - Part 2: Distance of Images “Distance entre images”
    - Part 3: Localization “Localisation du motif”
    - Part 4: Cross-Correlation “Corrélation croisée”

Goal

- Our goal is to detect a patter in an image, e.g., Charlie at the beach

Our Approach

- Slide pattern over image and compute a measure (difference/similarity) for every position



Measures

How different or similar is a pattern to an image of same size?

~

~

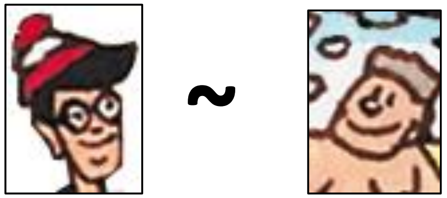
Representation of Image

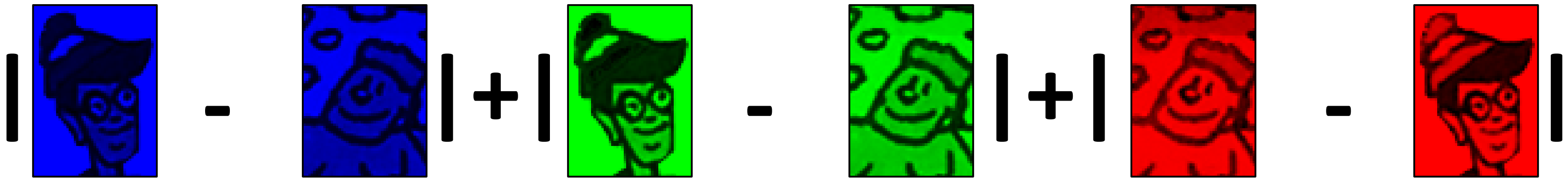
- Matrix of pixels
- Each pixel has Red, Green, and Blue-value (RGB)

(More details will be presented later.)

Measure 1: Difference

- Compute mean absolute error for each pixel by computing the absolute error per color channel and average over all colors and pixels





Measure 2: Similarity

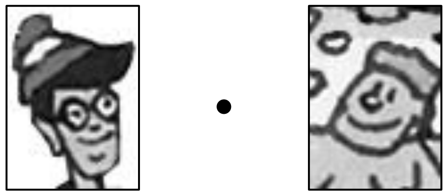
- Compute (zero-normalized) inner product between gray images

$$\begin{pmatrix} a_1 & b_1 & c_1 \end{pmatrix} \cdot \begin{pmatrix} a_2 \\ b_2 \\ b_3 \end{pmatrix} = a_1 \cdot a_2 + b_1 \cdot b_2 + c_1 \cdot c_2$$

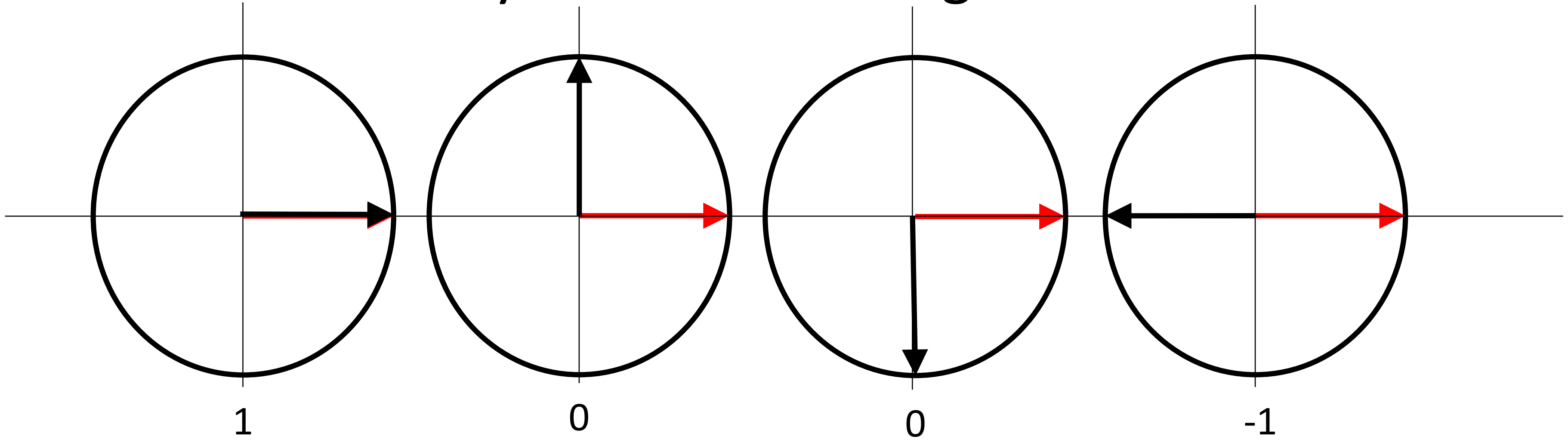
Sliding an inner product over an image correspond to computing the **cross-correlation** between the image and the pattern.
Cross Correlation is a **measure of similarity** of two function.
It is used, e.g., in **pattern recognition**, single particle analysis, electron tomography, averaging, cryptanalysis, and neurophysiology.

Inner Product

- Compute (zero-normalized) inner product between gray images

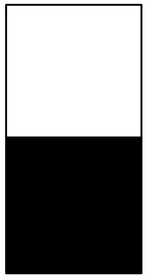


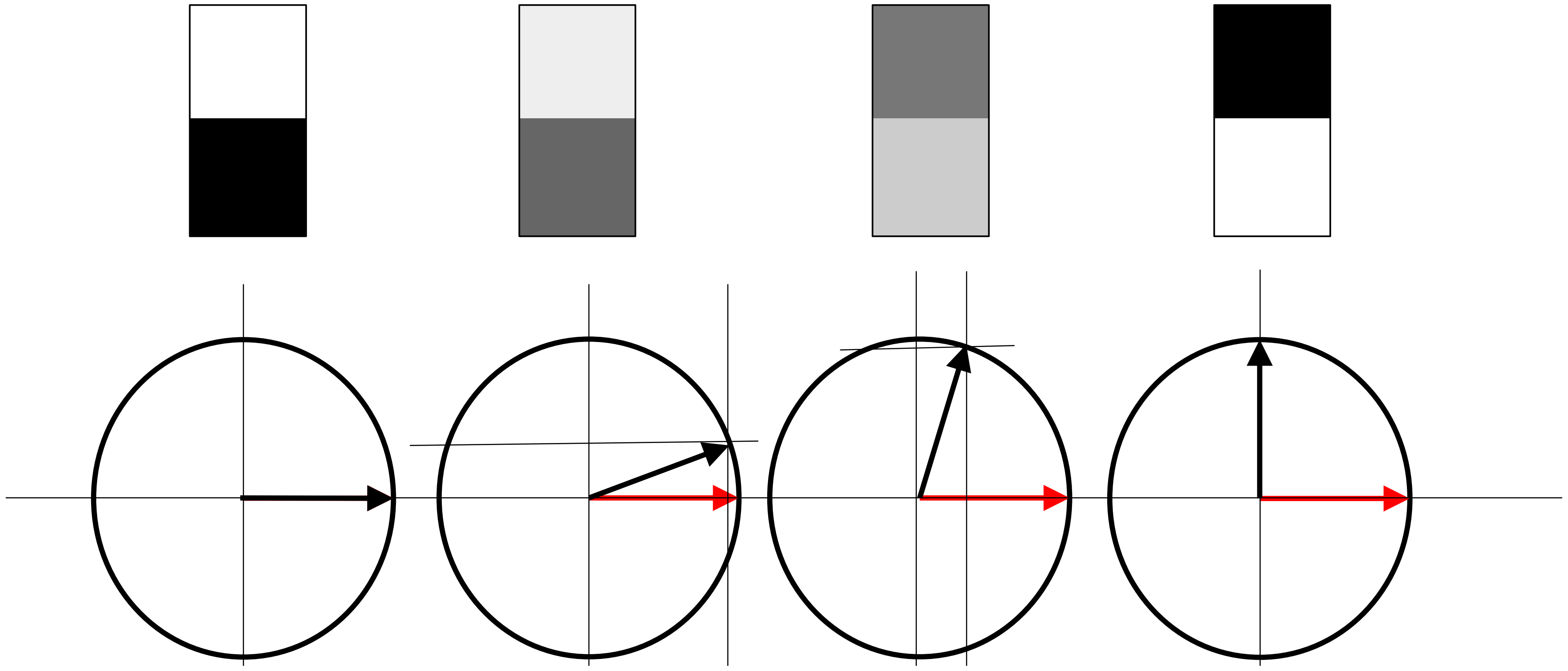
- Image = a vector
- Inner product of two vectors (of length 1) indicates how similar they are w.r.t. the angle



Inner Product for Gray Images

- Gray Images = Vector with positive values
- Assume gray image with only two pixels

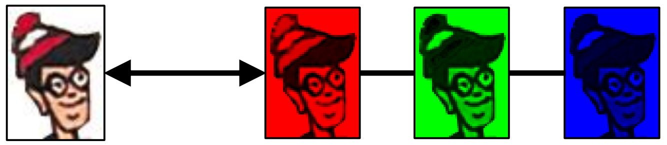


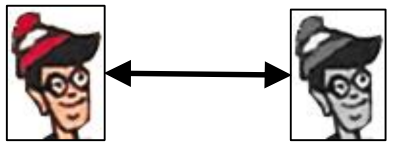


Inner product: 1 0.95 0.3 0

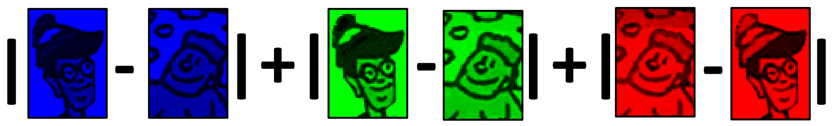
Project Overview

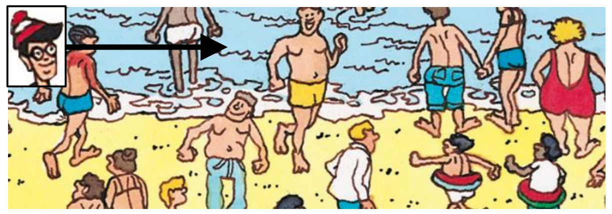
1. Image Processing





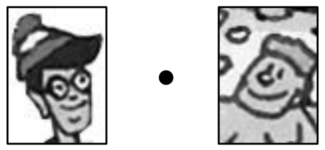
2. Distance Computation

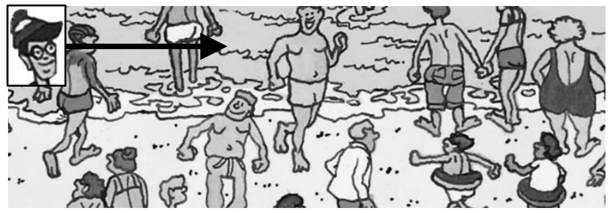




3. Localization: find best location of pattern

4. Cross-Correlation





Project Overview

1. Image Processing

    ImageProcessing.java

2. Distance Computation

    DistanceBasedSearch.java

3. Localization

    Collector.java

4. Cross-Correlation

    SimilarityBasedSearch.java

Handling Multiple Files (Classes)

- Up to now all you programs were contained in a single file.
- In this project you will be using **several files**
  - Given a static method m1() defined in a file A.java, and a static method m2() defined in a file B.java,
  - If you want to call m2 in the body of m1 you must use the following syntax; B.m2();
- E.g., in Main.java:

```
...
int[][] gray   = ImageProcessing.toGray(image);
...
```

Provided Code (1)

```
class Helper
```

- Read and write images to two-dimensional integer array

  ```
  public static int[][] read(String path)
  public static boolean write(String path, int[][] array)
  ```

- Display image

  ```
  public static void show(int[][] array, String title)
  ```

  Image will pop-up and program will be paused until image is closed.

- Draw rectangle at given position of given dimension into image

  ```
  public static void drawBox(int row, int col, int width, int height,
  int[][] image)
  public static void drawBox(int row, int col, int width, int height,
  int[][] image, int strokeWidth, int strokeColor)
  ```

- Example:

  ```
  int[][] image = Helper.read(“charlie.png");
  Helper.show(image, "Original");
  ```

Provided Code (2)

`class Main`

- Examples of how to use and test the methods
- These tests are **not exhaustive.**

`class SignatureChecks`

- Checks that the signatures of the required methods are correct (to simplify automatic testing).
- Does not check any functionality!

Outline

- Administrative
  - Information/Starting point
  - Submission and Groups
  - Submission Server and Tokens
- Project
  - Goal
  - Overview
  - Provided Code
  - Project Details:
    - Representation of Images
    - Part 1: Image Processing “Traitement d’images”
    - Part 2: Distance of images “Distance entre images”
    - Part 3: Localization “Localisation du motif”
    - Part 4: Cross-Correlation “Corrélation croisée”

Representation of Images

- Digital image = raster of pixel
  (or picture elements)
- Resolution = number of pixels used to represent an image, e.g., 1024x768 means
  - 1024 pixels from left to right
  - 768 pixels from top to bottom
- In this project: images are represented as two-dimensional arrays (of integers or doubles)
- E.g., `int[][] image = new int[30][50];`
  is an image with 30 rows (from top to bottom) and 50 columns (left to right) and a total of 30x50=1500 pixels.

Representation of Images

- Digital image = raster of pixel (or picture elements)
- Resolution = number of pixels used to represent an image, e.g., 1024x768 means
  - 1024 pixels from left to right
  - 768 pixels from top to bottom
- In this project: images are represented as two-dimensional arrays (of integers or doubles)
- E.g., `int[][] image = new int[30][50];`
  is an image with 30 rows (from top to bottom) and 50 columns (left to right) and a total of 30x50=1500 pixels. **Order in the project: first row, then column**

Refresher: Arrays in Java

Example	Functionality
`image.length`	Length of an array (height of image = number of rows)
`image[4]`	Access the element at position 4 **Recall**: first element is at position 0; last element is at position length-1
`image[4].length`	Length of element at position 4 (width of row 4)
`image[4][1]`	Access to element at row 4 and column 1
`new boolean[7]`	Create a new 1-dim. boolean array with 7 entries (0-6)
`new int[4][5]`	Create a new 2-dim. integer array with 4 rows (0-3) and 5 columns (0-4)

Example	Functionality
`Arrays.copyOf(msg, msg.length)`	Copies the specified array, truncating or padding with false (if necessary) so the copy has the specified length.
`Arrays.copyOfRange(message,0,10)`	Copies the specified range of the specified array into a new array.

Color Images (RGB Values)

- Each pixel has a color defined by an RGB (Red-Green-Blue) value.
- The RGB color model is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors.
  - Each RGB value is represented by three Bytes (3x8 bits), one Byte for each color.
  - Each base colors can have an intensity between 0 (min) and 2^8-1=255 (max) in decimal, or equivalently from 00 to ff in hexadecimal.
  - In JAVA, the RGB value is stored as integer.

Refresher: Numbers in Java

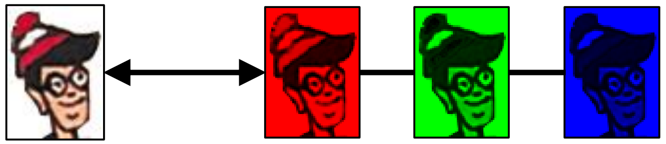
- Decimal (base 10):
```
int decValue = 13;
```
- Binary (base 2: 1bit):
```
int binValue1 = 0b00000000000000000000000000001101; //32-bits
int binValue2 = 0b1101; //leading zeros are not required
```
Starting with Java 7 you can use underlines for readability. Underlines are optional.
```
int binValue1 = 0b00000000_00000000_00000000_00001101;
```
- Hexadecimal (base 16: 4bits):
```
int hexValue1 = 0x00_00_00_0d;
int hexValue2 = 0xd; //leading zeros are not required
```
- Color in JAVA: integer (4 bytes = 32 bits), e.g.,

Color	Unused/alpha	Red	Green	Blue
In binary	00000000	00100000	11000000	11111111
In hexad.	00	20	c0	ff
In decimal	0	32	192	255

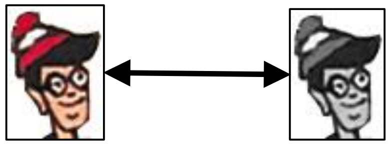
Task 1: Image Processing

ImageProcessing.java

1. Extract Red, Green, or Blue value from a pixel
   - Select bits from integer



2. Convert image to gray image
   - Iterate over image and compute average of red, green, and blue value for each pixel



Selecting Bits from Integer

start end

```
int val1 = 0b00000000_10000000_00001010_00000001;
```

- Step 1: shift right >>

  ```
  int val2 = val1 >> 9;
  ```

  Output: 0b00000000_10000000_0000101

- Step 2: mask &

  ```
  int val3 = val2 & 0b111;
  ```

  Output: 0b~~00000000_00000000_0000~~101

  0b101

Merging Bits into Integer

```
int val1 = 0b1010;
int val2 = 0b10000001;
```

Goal: `0b1010_10000001`

- Step 1: shift left <<

  ```
  int val1_sl = val1 << 8;
  ```

  Output in binary: `1010_00000000`

- Step 2: bitwise-or |

  ```
  int val3 = val1_sl | val2;
  ```

  Output in binary: `1010_10000001`

Outline

- Administrative
  - Information/Starting point
  - Submission and Groups
  - Submission Server and Tokens
- Project
  - Goal
  - Overview
  - Provided Code
  - Project Details:
    - Representation of Images
    - Part 1: Image Processing “Traitement d’images”
    - Part 2: Distance of images “Distance entre images”
    - Part 3: Localization “Localisation du motif”
    - Part 4: Cross-Correlation “Corrélation croisée”

Task 2: Distance of Images (1)

1. Compute mean absolute error between two pixels (between two integers representing RGB-values) Given pixels M and I, let M_c be the value of M for color c, then the mean absolute error is compute as

$$EA(M,I) = \frac{\sum\limits_{c \in C} |M_c - I_c|}{|C|}$$

DistanceBasedSearch.java

2. Compute mean absolute error between two images of same size.

| - |

DistanceBasedSearch.java

$$EAM(M,I) = \frac{1}{d} \sum_{(i,j) \in dim(P)} EA(M(i,j), I(i,j))$$

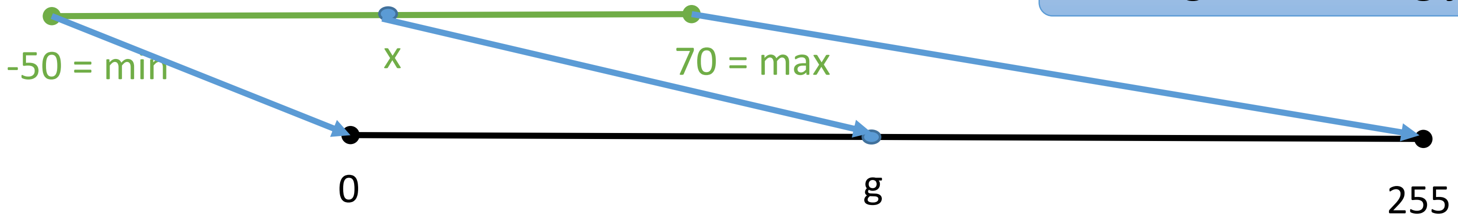
Task 2: Distance of Images (2)

3. Compute a matrix that stores for each possible position of the pattern on the base image the distance between the pattern and the covered part of the image. What is its size?

DistanceBasedSearch.java

4. Visualize the generated matrix: scale the values in the matrix to the interval [0-255].

ImageProcessing.java



$$g = \frac{255}{70+50}(x+50) = \frac{255}{max - min}(x - min)$$

Task 3: Localization

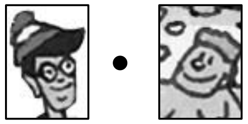
Collector.java

1. Find the row, column coordinates of the best element (biggest or smallest) for the given matrix
2. Find the row, column coordinate-pairs of the n best (biggest or smallest) elements of the given matrix

Task 4: Cross-Correlation

SimilarityBasedSearch.java

1. Computes the Normalized Cross Correlation of a gray-scale pattern M with part of a gray-scale image I



Inner product

Subtracting the mean

$$NCC(M,I,\underbrace{\boxed{r,c}}_{\text{Position of pattern on image}}) = \frac{\overbrace{\sum_{(i,j)\in dim(M)} [I(r+i,c+j) \boxed{-\bar{W}}] \times [M(i,j) \boxed{-\bar{M}}]}^{\text{Inner product}}}{\underbrace{\sqrt{\sum_{(i,j)\in dim(M)} [I(r+i,c+j)-\bar{W}]^2 \times \sum_{(i,j)\in dim(M)} [M(i,j)-\bar{M}]^2}}_{\text{Dividing by the standard deviation}}}$$

Position of pattern on image

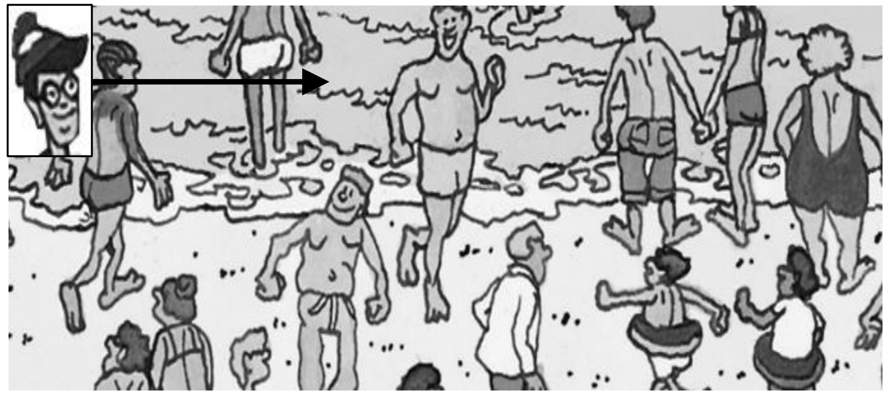
Dividing by the standard deviation

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**.

Task 4: Cross-Correlation

SimilarityBasedSearch.java

2. Compute a matrix that stores for each possible position of the pattern on the base image the Normalized Cross Correlation between the pattern and the covered part of the image.



Summary

- Administrative
  - Information/Starting point
  - Submission and Groups
  - Submission Server and Tokens
- Project
  - Goal
  - Overview
  - Provided Code
  - Project Details:
    - Representation of Images
    - Part 1: Image Processing “Traitement d’images”
    - Part 2: Distance of images “Distance entre images”
    - Part 3: Localization “Localisation du motif”
    - Part 4: Cross-Correlation “Corrélation croisée”