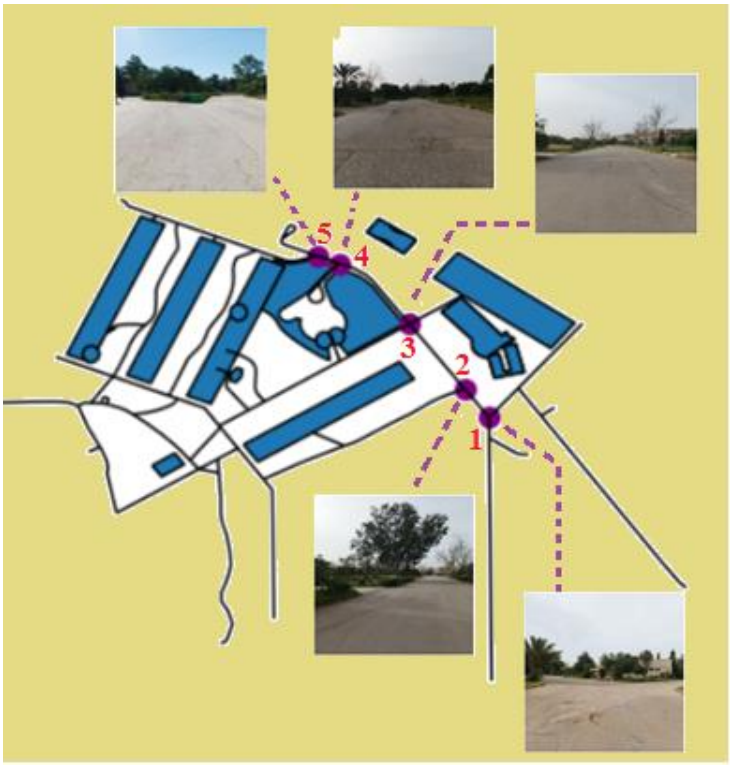
ISSN 2830-9820

Visual Computing MAGAZiNE

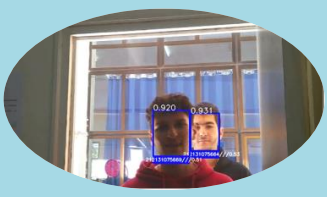
Vol. 2, Issue 1
2024



Mathematical Tools in Image Processing and Computer Vision

Visual Geo-Localization from images

An Embedded Intelligent System for Attendance Monitoring



Visual Computing Magazine

The Foreword

In this issue, we welcome Professor Mohammed HACHAMA from the National Higher School of Mathematics, who agreed to publish his presentation entitled "Mathematics for image processing and computer vision", delivered during the Doctoral Days of the Faculty of Computer Science of USTHB University.

This issue also includes two selected works carried out under my direction at the RIIMA laboratory. The first is carried out by Rania SAOUD, a Master's student, focused on visual geolocation from images using deep learning. The results obtained are encouraging to produce a general public product.

The second work is a collaborative effort of two undergraduate students: Abderraouf TOUZENE and Wassim ABED ABDELJALIL, focused on the development and deployment of an integrated smart attendance marking system using a Raspberry device dedicated to a smart university campus.

Chief Editor, Prof. Slimane LARABI
Computer Science Faculty
USTHB University
Algiers, Algeria

Visual Computing: A quarterly magazine. Volume 2, Issue 1, 2024
***ISSN:* 2830-9820**
Chief Editor: Prof. Slimane LARABI

Visual Computing Magazine

Mathematical Tools in Image Processing and Computer Vision

Prof. Mohammed HACHAMA, National Higher School of Mathematics, Algeria

1. Introduction

Image processing can be defined as the use of computers to process digital images through mathematical methods and algorithms to enhance an image quality or extract meaningful information.

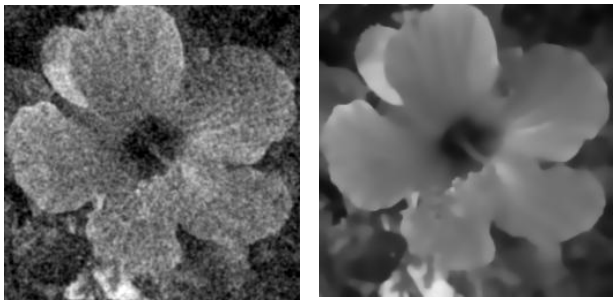
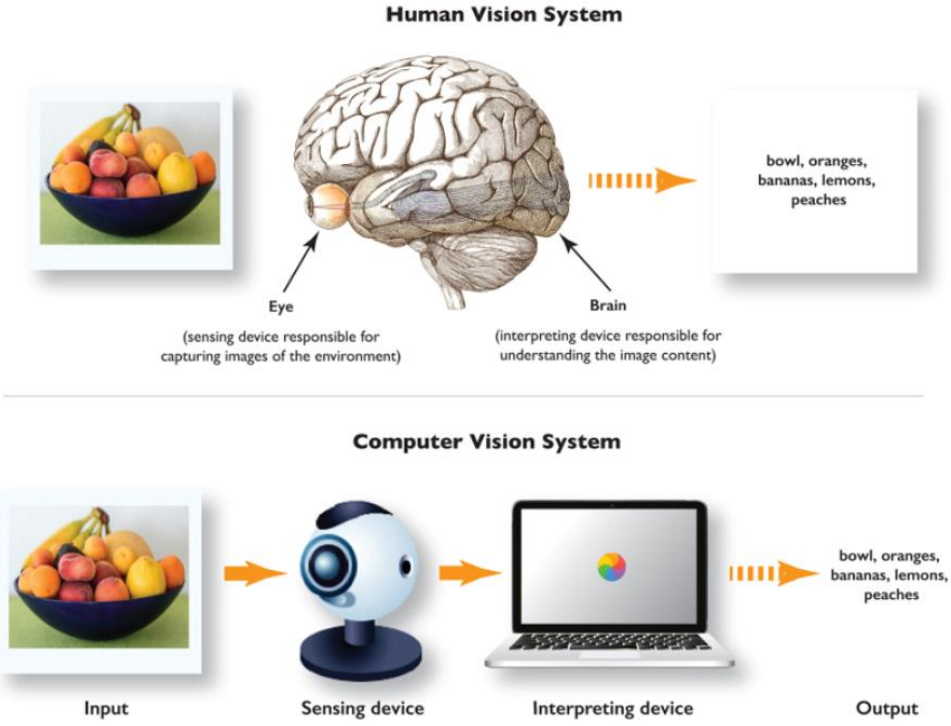


Figure 1: (Left) Raw image, (Right) Processed image

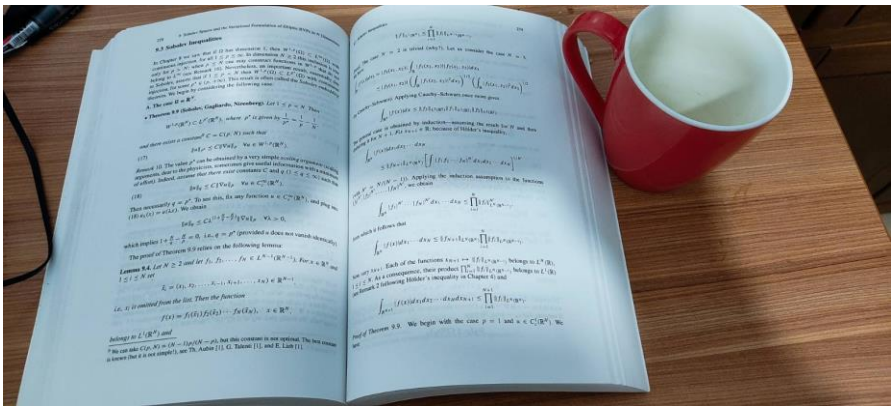
Computer vision aims to teach computers to see and interpret the world around them.



Mathematical Tools in Image Processing and Computer Vision

Prof. Mohammed HACHAMA, National Higher School of Mathematics, Algeria

Image processing and **computer vision** are domains of applications at the intersection of many disciplines such as mathematics, computer science, electronic, medicine. Nowadays, we use applications of image processing and computer vision on daily basis, such as Get real-time GPS navigation, traffic, and transit info (e.g. google map), diverse phone applications such as scanning.



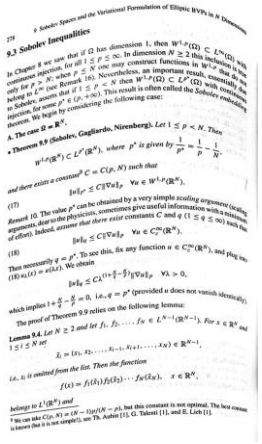
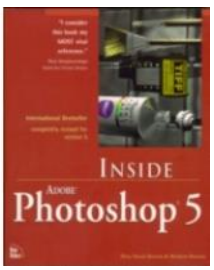


Figure 2: A scanned page of the book

2. Applications

There are many applications of image processing, such as:

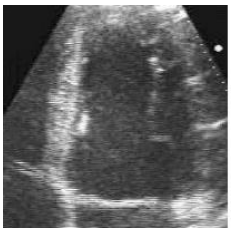
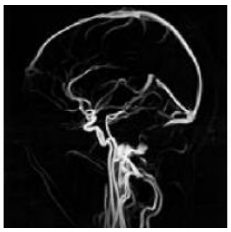
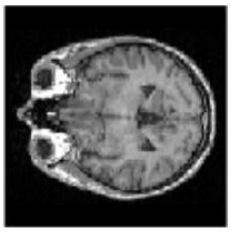
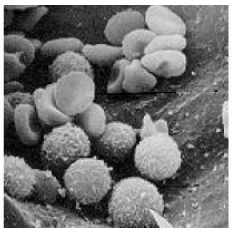
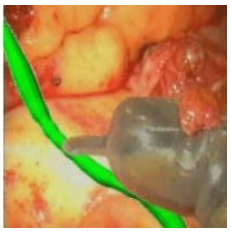
- Large public, especially smartphones applications



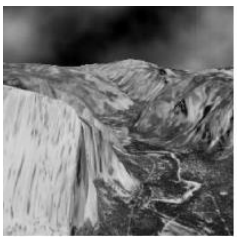
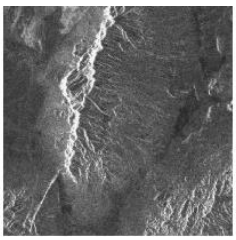
Mathematical Tools in Image Processing and Computer Vision

Prof. Mohammed HACHAMA, National Higher School of Mathematics, Algeria

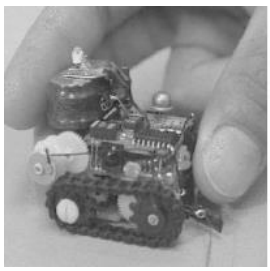
- Medical imaging used to automatic detection, diagnostic and computer aided decision

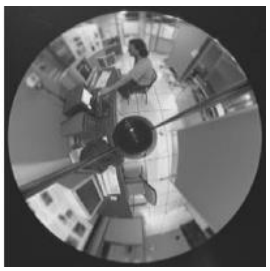






- Aerial and satellite images: for weather forecasting, vegetation estimation, fire spread estimation

- Robotics: robots, autonomous cars



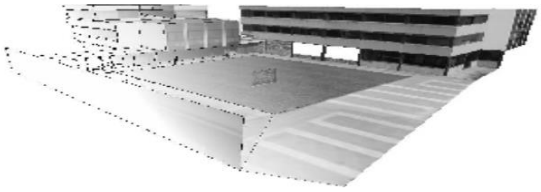
Mathematical Tools in Image Processing and Computer Vision

Prof. Mohammed HACHAMA, National Higher School of Mathematics, Algeria

- Security: especially, bio-metric identification (iris or fingerprint)




- Cinema: most of the movies scenes are created and edited using computers.

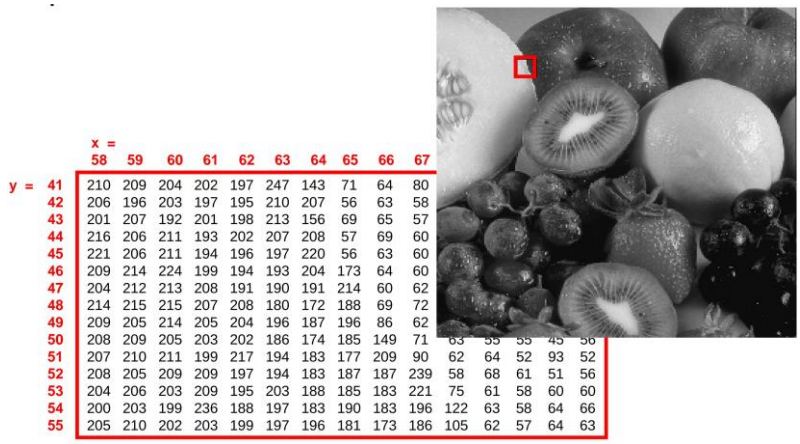
3. Mathematical Image Processing

Images can be represented as matrices. Each entry of the matrix represent some information such as the intensity, color, distance, heat, magnetic resonance

Images can also be represented as functions ($u : \Omega \subset R^2 \longrightarrow R$), which is more convenient in many cases and allows to use many tools of mathematical analysis, such as PDEs, variational calculus, functional analysis, … .

Mathematical Tools in Image Processing and Computer Vision

Prof. Mohammed HACHAMA, National Higher School of Mathematics, Algeria



	x = 58	59	60	61	62	63	64	65	66	67
y = 41	210	209	204	202	197	247	143	71	64	80
42	206	196	203	197	195	210	207	56	63	58
43	201	207	192	201	198	213	156	69	65	57
44	216	206	211	193	202	207	208	57	69	60
45	221	206	211	194	196	197	220	56	63	60
46	209	214	224	199	194	193	204	173	64	60
47	204	212	213	208	191	190	191	214	60	62
48	214	215	215	207	208	180	172	188	69	72
49	209	205	214	205	204	196	187	196	86	62
50	208	209	205	203	202	186	174	185	149	71
51	207	210	211	199	217	194	183	177	209	90
52	208	205	209	209	197	194	183	187	187	239
53	204	206	203	209	195	203	188	185	183	221
54	200	203	199	236	188	197	183	190	183	196
55	205	210	202	203	199	197	196	181	173	186

Image = Matrix, Pixel = indexes + intensities

Using these representations, images can be considered as mathematical objects that can be dealt with mathematical models and equations. One can do computations with images just as we do with numbers.

3.1 Partial Differential Equations

Filtering

Image filtering can be defined as any image transformation of the form: $\mathbf{v} = \mathbf{A}\,\mathbf{u}$, where $\mathbf{v}$ is the output image, $\mathbf{u}$ is the input and $\mathbf{A}$ is the transformation operator.

A first basic processing is linear (time-invariant) filtering. It consists in replacing the value of each pixel by a weighted sum of its neighbors, where the weights (called kernel) can define different processing.

Mathematical Tools in Image Processing and Computer Vision

Prof. Mohammed HACHAMA, National Higher School of Mathematics, Algeria

$F_1 = 1/9$

1	1	1
1	1	1
1	1	1

I' = (170+255+255+170+255+255+255+255+255)/9 = 236.111

255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255
255	255	85	85	255	255	255	255	255	255
255	255	85	85	255	255	255	255	255	255
255	255	85	85	255	255	255	255	255	255
255	255	85	170	170	170	170	170	255	255
255	255	85	170	170	170	170	170	255	255
255	255	255	255	255	255	255	255	255	255

	236	217	217	236	255	255	255	255	
	217	179	179	217	255	255	255	255	
	198	142	142	198	255	255	255	255	
	198	151	142	189	227	227	236	246	
	198	161	142	179	198	198	217	236	
	217	198	179	198	198	198	217	236	

Another very useful filter uses a Gaussian kernel:

$$F_g = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

The result is equivalent to a mathematical convolution with a Gaussian function:

$$u_\sigma = (g_\sigma \star u_0)(x) \quad \text{where} \quad g_\sigma(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{|x|^2}{2\sigma^2}\right)$$

$\sigma = 0$ $\sigma = 5$ $\sigma = 11$ $\sigma = 17$

Mathematical Tools in Image Processing and Computer Vision

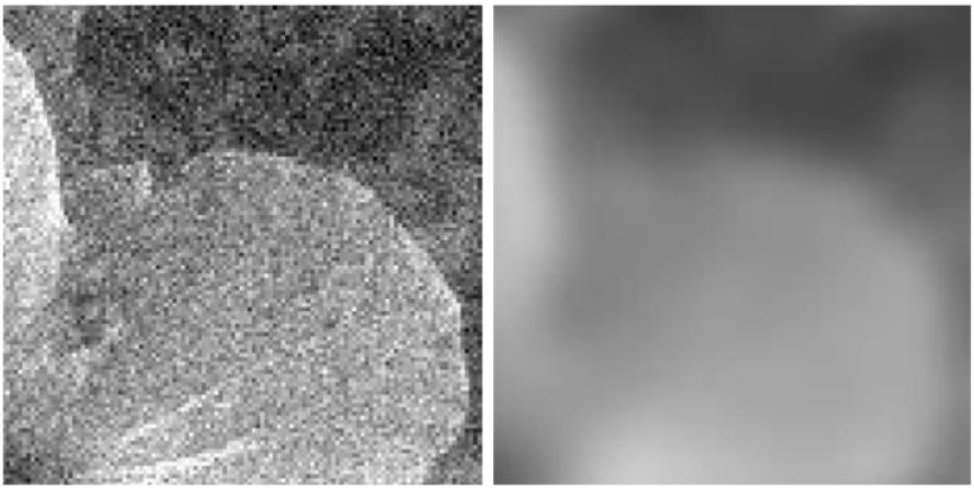
Prof. Mohammed HACHAMA, National Higher School of Mathematics, Algeria

Diffusion

The convolution with a Gaussian is also the exact solution of the heat equation on R^2:

$$\begin{cases} \dfrac{\partial u}{\partial t}(x,t) = \Delta u(x,t), & t \geq 0,\ x \in \mathbb{R}^2 \\ u(x,0) = u_0(x). \end{cases}$$

This equation performs a isotropic diffusion. Diffusion is the tendency to make each pixels similar to his neighborhood (neighborhood) in terms of intensities. Isotropy means that diffusion is applied everywhere, with ”the same strength”.



From the figure above, we can observe that the heat equation has one good effect when diffusion is applied inside the regions and a bad effect when applied on the boundaries, leading to a blurred image. Perona and Malik proposed an anisotropic PDE of the form Equation:

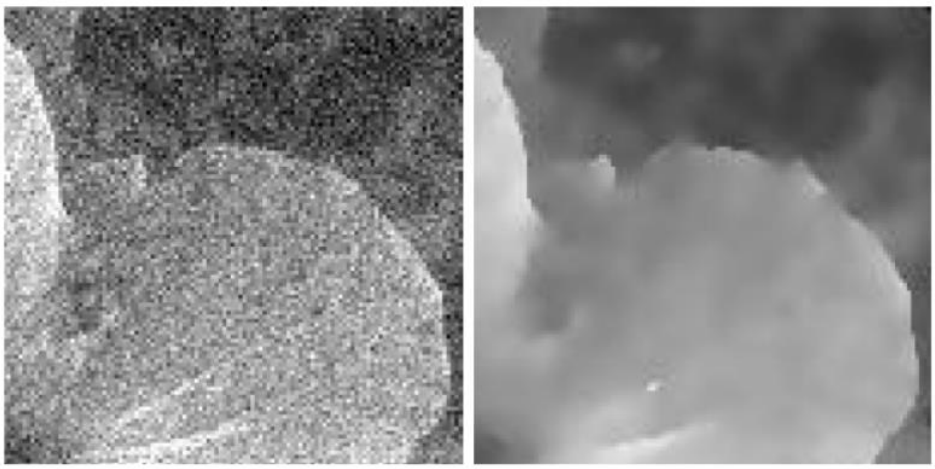
Mathematical Tools in Image Processing and Computer Vision

Prof. Mohammed HACHAMA, National Higher School of Mathematics, Algeria

$$\begin{cases} \frac{\partial u}{\partial t} & = & \nabla \cdot (c(|\nabla u|)\,\nabla u) & \Omega \times]0,T[\,, \\ u(x,0) & = & u_0(x) & \Omega, \\ u_n & = & 0 & \partial\Omega \times]0,T[\end{cases}$$

where $c(s) > 0$, non-increasing, $c(0) = 1$, and $\lim_{x \to +\infty} c(s) = 0$, e.g.,

- $c_1(s) = 1/(1 + s^2/\lambda^2)$,
- $c_2(s) = \exp\left(-s^2/(2\lambda^2)\right)$,
- Charbonnier diffusivity: $c_3(s) = 1/(\sqrt{1 + (s/k)^2})$.



In the same spirit, Weickert et al. proposed two PDEs that can be seen as modifications of the heat equation. They proposed to Take direction of the image gradients into account.

$$\frac{\partial u}{\partial t} = \nabla \cdot (D(\nabla u)\,\nabla u)$$

Mathematical Tools in Image Processing and Computer Vision

Prof. Mohammed HACHAMA, National Higher School of Mathematics, Algeria

where D represents a matrix-valued diffusion tensor that describes the smoothing directions and the corresponding diffusivities. Let μ1 and μ2 the eigenvalues of D. The idea is to modify these eigenvalues to obtain new ones λ1, λ2 for specific applications.

- Edge-enhancing anisotropic diffusion: $\frac{\partial u}{\partial t} = \nabla \cdot (D(J(\nabla u))\nabla u)$

$$\begin{aligned} \lambda_1 &= \begin{cases} 1 & \mu_1 = 0 \\ 1 - \exp\left(\frac{-\alpha^2}{\mu_1^4}\right) & else \end{cases} \\ \lambda_2 &= 1 \end{aligned}$$

- Coherence-enhancing aniso. Diffusion : $\frac{\partial u}{\partial t} = \nabla \cdot (D(J(\nabla u))\nabla u)$

$$\begin{aligned} \lambda_1 &= \alpha \\ \lambda_2 &= \begin{cases} \alpha & \mu_1 = \mu_2 \\ \alpha + (1-\alpha)\exp\left(\frac{-1}{(\mu_1-\mu_2)^2}\right) & else \end{cases} \end{aligned}$$

Mathematical Tools in Image Processing and Computer Vision

Prof. Mohammed HACHAMA, National Higher School of Mathematics, Algeria

Deconvolution

Another variant of the heat equation can be defined to perform a deconvolution, i.e, remove the blur from an input image. A naive idea would be to "reverse" the heat equation:

$$\begin{cases} \dfrac{\partial u}{\partial t}(x,t) = -\Delta u(x,t), \quad 0 \leq t \leq T,\ x \in \mathbb{R}^2 \\ u(x,T) = u_0(x). \end{cases}$$

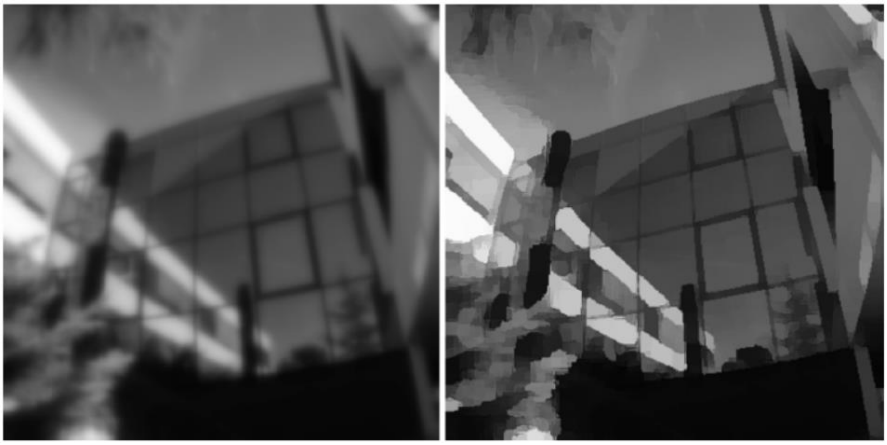
such equation turned to be numerically unstable. Osher-Rudin proposed another variant, the "shock filter", defined as:

$$\frac{\partial u}{\partial t} = -sign(\Delta u(x,t))\,|\nabla u(x,t)|.$$

This equation gives good deconvolution results.

Mathematical Tools in Image Processing and Computer Vision

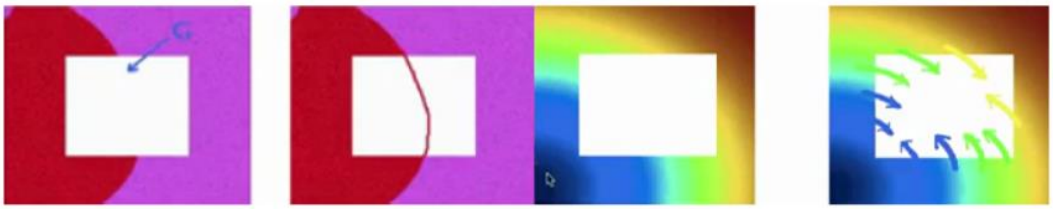
Prof. Mohammed HACHAMA, National Higher School of Mathematics, Algeria



Inpainting

Inpinting is the process of restoration where some pixels are missing in the input image.

On idea is to use a PDE which allows to ”transport” some information from the region of known information to the missing pixels. This can be expressed as a constant information in some direction.



Mathematical Tools in Image Processing and Computer Vision

Prof. Mohammed HACHAMA, National Higher School of Mathematics, Algeria

The transport equation can be written as follows :

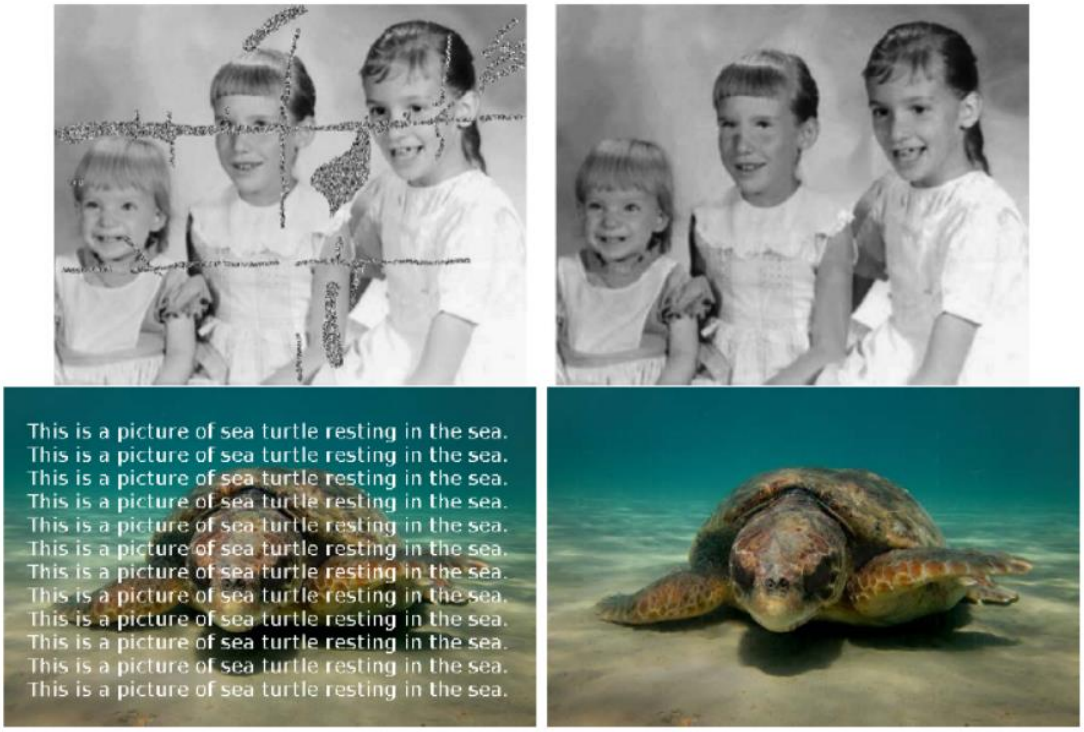
- Transport the known image structures along the level lines.

$$\nabla \underbrace{\Delta u}_{\text{Info.}} . \underbrace{\nabla^{\perp} u}_{\text{Direction}} = 0$$

- Evolution

$$u_t = \nabla \Delta u . \nabla^{\perp} u$$

These equations performs very well in image inpainting (see Figures below).



Mathematical Tools in Image Processing and Computer Vision

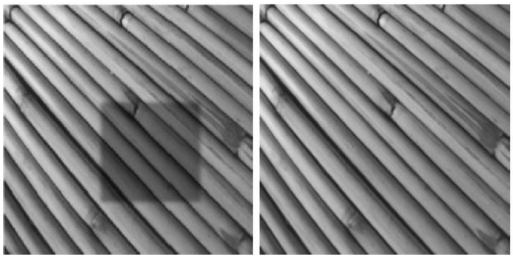
Prof. Mohammed HACHAMA, National Higher School of Mathematics, Algeria

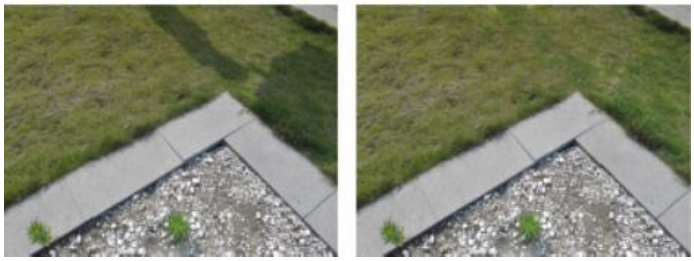
Shadow removal

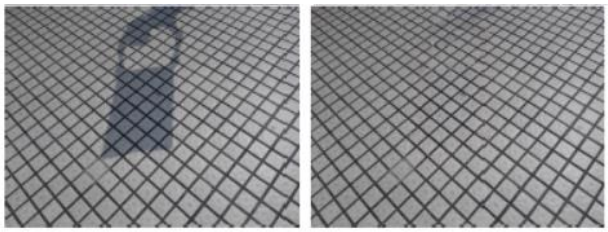
Another application of PDE is shadow removal. This is a very important application that can be used a pre-processing for further applications.
One PDE used to perform that is the following :

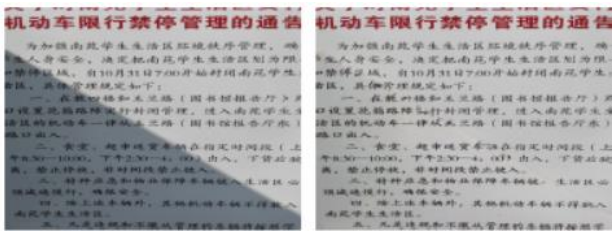
$$\partial_t u\,(t,x) = \mathrm{div}_{NL}\left(v(\mathbf{x})\nabla_{NL}\left(\frac{u(t,\cdot)}{v}\right)(\mathbf{x},\mathbf{y})\right) + \lambda_1\chi_{sb}(\mathbf{x})\,\mathrm{div}(\mathbf{W}(\mathbf{x})\nabla u(t,\mathbf{x})) + \lambda_2\chi_{out}(\mathbf{x})\,(f(\mathbf{x})-u(t,\mathbf{x})).$$

where ∇NL is a nonlocal gradient, W is a matrix estimating the directions, χ_{sb} and χ_{out} are characteristics functions of the shadow boundary and the shadow-free region.









Mathematical Tools in Image Processing and Computer Vision

Prof. Mohammed HACHAMA, National Higher School of Mathematics, Algeria

Image fusion

Background selected region foreground blending

Image fusion is another important application using PDEs. To solve this problem, one can use the following PDE:

$$\partial_t u\,(t,x) = \mathrm{div}_{NL}\left(v(x)\nabla_{NL}\left(\frac{u(t,.)}{v}\right)(x,y)\right) + \lambda\alpha(x)(f(x)-u(t,x)).$$

where v is a "rough solution" of the fusion problem and α is an interpolating function defined as follows:

$$\alpha(x) = \begin{cases} 1, & \text{if } x \in \Omega_f, \\ G(x), & \text{if } x \in \Omega_{sb}, \\ 0, & \text{if } x \in \Omega_b. \end{cases}$$

Blending rough solution and a PDE solution

Visual Computing Magazine

Visual Geo-Localization from images

Rania SAOUD, USTHB University, Algeria

Abstract
This paper presents a visual geo-localization system capable of determining the geographic locations of places (buildings and road intersections) from images without relying on GPS data. Our approach integrates two methods: Scale-Invariant Feature Transform (SIFT) for place recognition and deep learning using the VGG16 model for classifying road junctions. The most effective techniques have been integrated into an offline mobile application, enhancing accessibility for users requiring reliable location information in GPS-denied environments.

1. Introduction

Geo-localization involves determining the precise location of an image or video in the real world, typically using latitude and longitude coordinates obtained through technologies such as GPS, Wi-Fi, and IP address tracking. Algorithms process this data to pinpoint exact coordinates[1][2]. Geo-localization is important for organizing and analyzing large volumes of imagery data, as demonstrated by systems like the US Geological Survey (USGS), which classify and locate satellite and drone images to streamline data collection and analysis.

Visual geo-localization addresses these issues by identifying the geographic position of an image using visual cues within the image itself, rather than relying on external metadata like GPS tags[2]. This technique uses attributes such as building facades, points of interest, and geographical information to compare the query image with a database of geo-referenced images[1]. Based on Sift descriptors and recognition of road intersection types (see figure 1), the goal of this work is to provide accurate location predictions in situations where conventional methods, such as GPS, are ineffective, inaccurate, or unavailable.

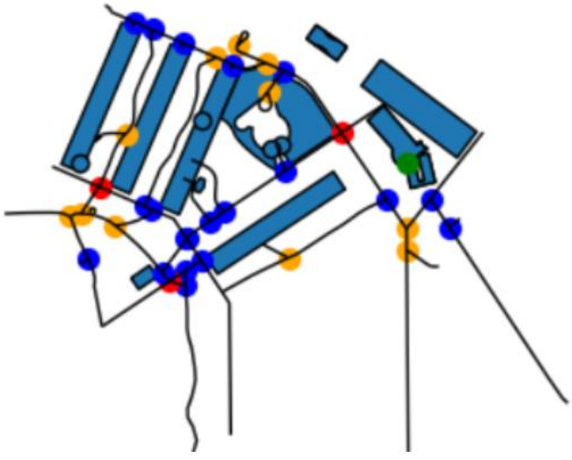


Figure 1. Example of a part of map indicating many types of road junctions

Visual Geo-Localization from images

Rania SAOUD, USTHB University, Algeria

2. Proposed System

2.1. Place Recognition using SIFT descriptor

In this approach, we focus on place recognition by comparing input images against a dataset of panoramic images using SIFT descriptors. First, we create the dataset by extracting frames from videos, which are preferred over still images due to their continuous and comprehensive environmental capture. We then apply the SIFT algorithm to selected images to match their features against the panoramic dataset using the FLANN matcher. This step identifies the most similar place for each image. Finally, a voting process determines the most likely place based on the results from all analyzed images.

1) Dataset Creation :

The initial step in our dataset creation involves extracting frames from video recordings. To construct panoramic images, we stitch multiple frames together using algorithms that align overlapping areas based on shared features. This process helps overcome challenges such as changes in lighting and camera movement, and we enhance the accuracy of the stitch through techniques like color normalization and feature alignment.

After stitching, we crop unnecessary black edges from the panoramic images to focus on relevant content, thereby optimizing the data for further processing. Using the SIFT technique, we then extract key features from the images that remain consistent despite changes in scale or rotation, making it easier to match images from different scenes. Additionally, we use APIs like Google Places to gather detailed metadata about specific locations, enriching our dataset with precise geographical coordinates. These coordinates are integrated using Geographic Information Systems (GIS) for detailed mapping and spatial analysis, enhancing the practical application of our geolocation tasks. Figure 2 represents a panoramic dataset taken in our university.

Figure 2. Some computed panoramic images of our university campus

Visual Geo-Localization from images

Rania SAOUD, USTHB University, Algeria

2) Image Classification and Filtering using Detectron2 :

Detectron2, a tool created by Facebook AI, performs panoptic segmentation on the images provided by the user. The model uses Panoptic FPN R-101 from Detectron2's model zoo.
Each image is first adjusted to the RGB color format, which is necessary for processing with Detectron2. The model then segments the images, identifying different elements like roads and pavements. It calculates how much of each image is covered by roads and pavements. A threshold is set so that only images where roads and pavements make up less than 40% of the image move on to the next phase. This ensures that only images with significant non-road elements are chosen for place matching using SIFT. For instance, Figure 3 displays examples of images selected for further processing.

3) Place Matching using SIFT :

We take the images classified in the previous step for SIFT matching and generate SIFT descriptors for each image. We use the FLANN-based matcher for its efficiency in handling large datasets. This is done by checking if the distance of the nearest match is less than 75% of the distance of the second nearest match, thus eliminating less likely matches. Each query image contributes a 'vote' to the place it matches best based on the count of good matches, so the place receiving the highest number of votes across all queries is determined to be the most similar place. Finally, we generate a visual representation using Folium by placing a green marker on a map at the coordinates corresponding to the most voted place, providing an intuitive geographical context to the results. Figure 4 is an example of the most similar place for the mathematics faculty building images.

Figure 3. Example of selected images for place matching using Detectron2

Visual Geo-Localization from images

Rania SAOUD, USTHB University, Algeria

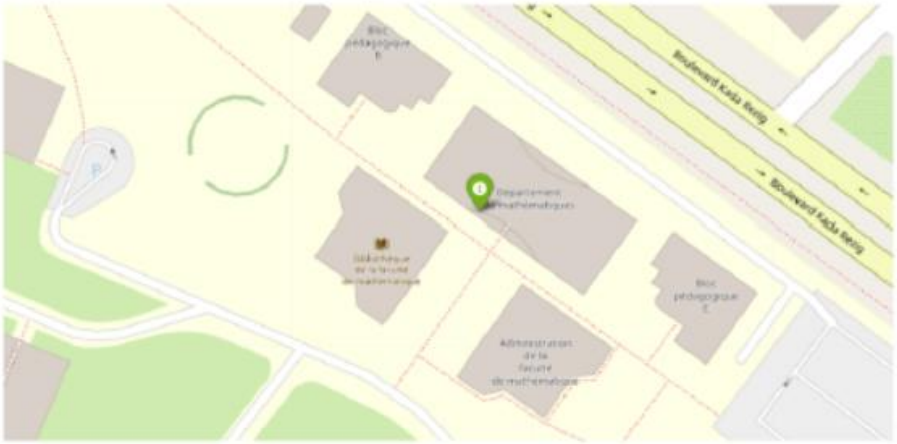


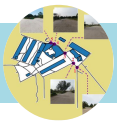
Figure 4. Visual representation of the most similar place

2.2. Road junction Identification using deep learning

The second approach is to identify road junctions using a deep learning method with the VGG16 model, pre-trained on a large database of images. We start by collecting diverse images of roads, including features like T-junctions, X-junctions, Y-junctions, and roundabouts.
These images are labeled to highlight specific road features. The VGG16 model is then fine-tuned to focus on road junctions, ensuring it performs well with new, unseen images. After training, the model is tested with new images to evaluate its capability in identifying and classifying different road features.

Model Preparation and Training:
- Preprocessing: Images were resized to 224x224 pixels and normalized to suit the input requirements of the VGG16 model.
- Class Weight Calculation: To address dataset imbalances, class weights were calculated and applied during training to ensure equitable learning across less frequent classes.
- Model Architecture Modifications: The base VGG16 model was customized by removing the top layer and adding dense layers with ReLU activation and dropout layers to enhance feature learning capabilities.

Model Training and Validation:
- Training Set Preparation: The dataset was split into training, validation, and test sets, with augmentation techniques such as random flipping and brightness adjustment applied to enhance generalization.
- Training Process: Initially, only the top layers of the model were trained. Techniques like batch normalization and dropout were used to improve training stability and prevent over-fitting.



Visual Geo-Localization from images

Rania SAOUD, USTHB University, Algeria

Fine-Tuning and Evaluation:
- Fine-Tuning: After the initial training, deeper layers of the VGG16 were gradually unfrozen and the model was fine-tuned with a reduced learning rate to refine its ability to classify junction types accurately.
- Model Evaluation: The fine-tuned model was tested against a separate set of images to assess its performance and ensure it could reliably classify road features in varied real-world conditions.
Our deep learning strategy harnesses the capabilities of the pretrained VGG16 model, which we fine-tune to classify various road junctions and features.

2.3. Integration and Map Matching

The methodology for integrating geographic data with a graph-based map system and the subsequent application for route mapping and validation is done as follow:

1) **Creation of Graph-based Road Map Dataset**

Initially, we define the geographic area of interest using JOSM, a robust tool for editing OpenStreetMap data. This tool enables us to tailor the map data precisely by adding or removing roads, buildings, and intersections, thus facilitating the map-matching process.
Then, we convert detailed road data into a structured graph. We use GeoJSON files to represent roads as line strings, which aids in constructing a graph where intersections are nodes and road segments are edges, enhancing our analysis of road connectivity. We classify junction types by identifying nodes where roads intersect, with different configurations indicating various junction types like T-junctions, X-junctions or Y-junctions (see Figure 5).
Integrating the data into a unified graph-based map involves consistency checks to ensure data accuracy and representation. The validation occurs through systematic verification of node adjacency's and junction classifications, ensuring the integrity of the geographic data within our application.

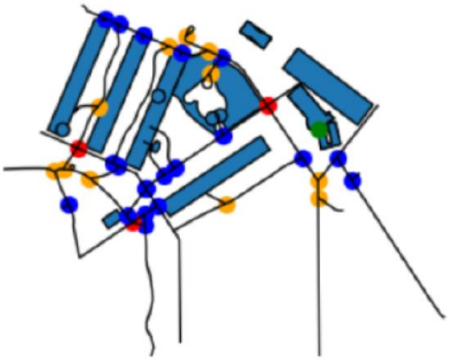


Figure 5. Junction types: X (red), Y (yellow), T (blue), Crossroad (red)

Visual Geo-Localization from images

Rania SAOUD, USTHB University, Algeria

2) Sequence Matching with Graph-Based Map

In this phase, we use the graph-based road map to identify potential routes by matching a sequence of junction types to the graph data. Starting with intersections close to a given location, we calculate distances using the Haversine formula to maintain focus on relevant intersections. We then apply a depth-first search algorithm to trace paths that match the defined sequence of junction types. This method not only aids in route planning but also enhances our navigation system's reliability by ensuring the paths adhere to expected road patterns.

Figure 6 illustrates a sequence of real images taken from intersections 1 to 5. Each image represents a critical point in the route, showcasing different types of road junctions encountered along the way. The pathfinding algorithm processes these images to determine the most probable route on the map. By using the intersection types and their sequences, the algorithm identifies the path through the road network. The resultant path is depicted in the map shown in Figure 5.

3) Validation and Correction Mechanism

Once a route is identified, we conduct a detailed validation of each junction of the sequence to confirm alignment with the expected path. Discrepancies are addressed by exploring alternative routes that provide a better match. This process is important for maintaining the accuracy and reliability of the navigation system, ensuring that the route suggestions are practical and reflect real-world conditions.

Figure 6: The images represent the road intersections encountered along the route. These include various junctions such as T-junctions, Y-junctions, and crossroads. The images provide a visual representation of the real-world path. The path computed by our algorithm is indicated by purple dots which correspond to the real path images.

Visual Geo-Localization from images

Rania SAOUD, USTHB University, Algeria

3. Experimental results

In this section, we present the results of our experiments conducted to evaluate the performance of our visual Geo-localization system. We will detail the performance metrics for each approach, compare the effectiveness of traditional image processing and deep learning methods, and discuss the results of our mobile application implementation.

3.1. Dataset Preparation and Annotation

We captured 282 images of road junctions using a smartphone, focusing on T-junctions, Y-junctions, X-junctions, and roundabouts. These images were taken from various distances and angles, ensuring a diverse dataset, a selection of images is displayed in Figure 7.
Using the VGG Image Annotator (VIA), we labeled the images to highlight specific road features. Accurate labeling is important for the model's performance, as shown in Figure 8.

3.2. Data Splitting and Balancing

The data was split into training (70%), validation (15%), and test (15%) sets. To address class imbalances, we used SMOTE (Synthetic Minority Over-sampling Technique) to generate synthetic samples.

Figure 7: The collected images.

Figure 8: Example of annotated image.

Visual Geo-Localization from images

Rania SAOUD, USTHB University, Algeria

Data Augmentation

To enhance the model's generalization, we used Keras' ImageDataGenerator for data augmentation. This technique artificially creates variations of our existing images by applying random transformations such as shifting, rotating, flipping, and altering brightness. These transformations increase the diversity of the dataset, helping the model to generalize better to new, unseen images. Additionally, SMOTE was used to create synthetic samples for the minority classes [3]. SMOTE works by generating new samples along the line segments joining existing minority class samples. This further balances the dataset and ensures that all classes are equally represented during training. Combining ImageDataGenerator and SMOTE significantly increased the number of training images. While the exact number of augmented images is not directly counted, the process resulted in more than 20,000 image variations over the training epochs, greatly enhancing the dataset's diversity.

3.3. Training and Fine-Tuning of the VGG16 Model

We used the Adam optimizer and categorical cross-entropy loss function for training the VGG16 model. The accuracy metric was used to measure the model's performance during training. The initial training was conducted over 50 epochs with a batch size of 32. For fine-tuning, we unfroze the last eight layers of the VGG16 model, training for an additional 30 epochs with a reduced learning rate to improve accuracy. We used callbacks like ReduceLROnPlateau and EarlyStopping to enhance training efficiency and prevent overfitting.

3.4. Results

We conducted tests where users uploaded images of buildings and road intersections. The app processed these images to determine the user's location, prioritizing intersection images for better accuracy. Figure 9 shows an example of three query images with a correct classification. Table 1 shows the values of precision, recall and F1-measure.

Figure 9: The first three images, well classified with respectively X, T and Y junction. The fourth image is misclassified, the correct type of junction is "X"

Visual Geo-Localization from images

Rania SAOUD, USTHB University, Algeria

By addressing the imbalance in the dataset, the model's performance on the test set improved by 3%. Below, the table 02 illustrates the changes in loss and accuracy.

	Precision	Recall	F1-score
Roundabouts	1.00	0.67	0.8
T-junction	1.00	1.00	1.00
X-junction	1.00	1.00	1.00
Y-junction	0.95	1.00	0.97
Accuracy			0.98

Table 1: Performance metrics of our fine-tuned VGG16 model

	Model Before Class Weights	Model After Class Weights
Training Loss	0.04	0.0294
Training Accuracy	0.97	0.9846
Test Loss	0.53	0.0959
Test Accuracy	0.94	0.9767

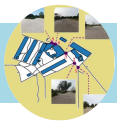
Table 2: Training and test loss and validation accuracy's before and after addressing the imbalance in the dataset using SMOTE.

Conclusion

The experimental results demonstrate the effectiveness of our proposed Geo-localization method using a deep learning approach. The proposed system which is implemented on mobile successfully integrates these methods, providing a practical tool for users in GPS-denied environments.
Future work will focus on expanding the dataset, further improving model accuracy, and adapting the system for augmented reality (AR) and indoor application.

References

[1] Zamir, A.R., 2014. Visual Geo-localization and Location-aware Image Understanding (Doctoral dissertation, University of Central Florida Orlando, Florida).
[2] Brejcha, J., Cad´ık, M., 2017. State-of-the-art in visual geo-localization. Pattern Analysis and Applications. 1, 2, 20
[3] Vehicle-borne laser mapping system (VLMS) for 3DGIS [17] Medium article, Synthetic Minority Over-Sampling Technique (SMOTE): Empowering AI through Imbalanced Data Handling



Visual Computing Magazine

An Embedded Intelligent System for Attendance Monitoring

TOUZENE Abderraouf, ABED Abdeljalil Wassim, USTHB University, Algeria

Abstract

In this paper, we propose an intelligent embedded system for monitoring class attendance and sending the attendance list to a remote computer. The proposed system consists of two parts : an embedded device (Raspberry with PI camera) for facial recognition and a web application for attendance management. The proposed solution take into account the different challenges: the limited resources of the Raspberry Pi, the need to adapt the facial recognition model and achieving acceptable performance using images provided by the Raspberry Pi camera.

Figure 1: Output of our system

1. Introduction

Managing attendance in educational institutions represents a major logistical challenge. Traditionally, manual attendance taking is not only time-consuming but also prone to human errors and potential fraud. To overcome these limitations, facial recognition has emerged as a promising technology to automate this process while ensuring increased reliability. However, implementing facial recognition systems on embedded devices like the Raspberry Pi presents significant challenges. The Raspberry Pi, with its limited computational power and memory resources, requires specific optimizations to efficiently run deep learning models that are typically resource-intensive. Furthermore, deployment in real-world environments, such as classrooms, imposes additional constraints such as variability in lighting conditions and the need for processing.

In this paper we propose an embedded system for attendance monitoring based on facial recognition. The proposed system is designed to capture images of participants, process these images to detect and recognize faces, and record attendance data via a web application *(see figure 1)*. The primary goal is to provide a practical and reliable solution for student identification in the classroom while addressing the specific constraints of resource-limited environments. It aims to demonstrate the feasibility and effectiveness of such an approach in an educational setting, providing a basis for future improvements and potential extensions.

This paper is organized as follows: section 2 presents the stages implemented in this work. In section 3 experimental results obtained are presented are discussed. Finally we conclude the paper with some remarks.

An Embedded Intelligent System for Attendance Monitoring

TOUZENE Abderraouf, ABED Abdeljalil Wassim, USTHB University, Algeria

2. Proposed System

The system architecture includes a facial recognition system, which operate in network mode, thereby enabling more reliable transfer and management of attendance data. This data is transferred to a database hosted on a server connected to the same network. In the absence of a network or server, the system will operate in hotspot mode, where it will create its own local network, allowing direct connection with the desktop application on the user's computer.
Developers can interact with the system via SSH (Secure Shell) and VNC (Virtual Network Computing) protocols for maintenance and deployment. Administrators and end users can access the system through a user-friendly interface, either via a web application or a desktop application, depending on the chosen connection method. Figure 2 illustrates a schematic representation of this architecture.

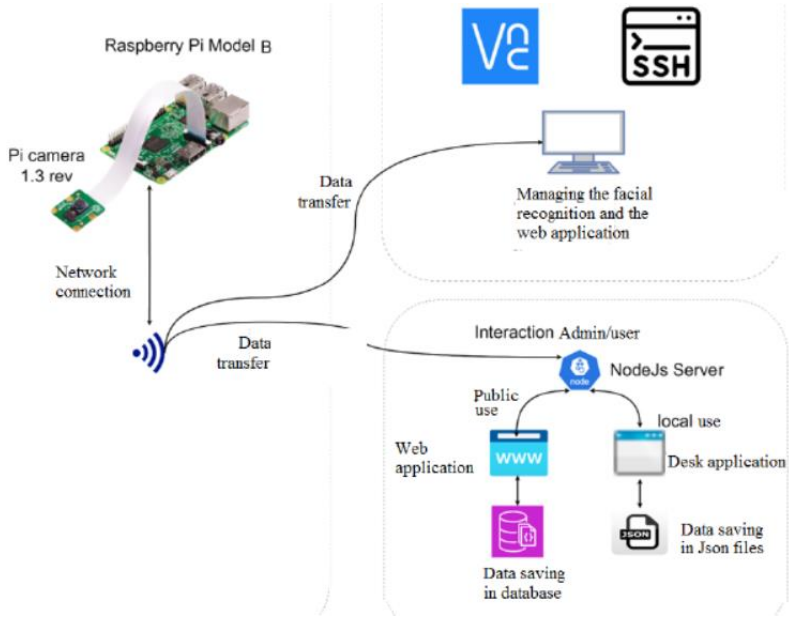


Figure 2: Overall system architecture

2.1 Reference Image Storage

To add a student's image to our database, we first apply a sharpness filter to enhance the image quality. This step improves the capture and representation of relevant facial features. Typically, the images used come from identity photos or student badges, which are often low in detail and slightly blurred.
Next, we use a facial representation model to extract a feature vector from the image. Once this encoding is obtained, it will be stored in our database, where it will serve as a reference for comparison with other images for student identification.

An Embedded Intelligent System for Attendance Monitoring

TOUZENE Abderraouf, ABED Abdeljalil Wassim, USTHB University, Algeria

2.2. Facial Recognition System

The embedded facial recognition system we have developed follows the following steps:
The system starts by capturing a video using the Pi Camera rev 1.3 module of the Raspberry Pi, which will be processed in the subsequent steps of our system.

Detection and tracking
The first step of our system is to extract the coordinates of the faces as well as the key points from the video. It outputs the coordinates of the bounding boxes and the key points for each detected face.
An object tracking algorithm has been implemented in our system to address two essential objectives. Firstly, to reduce the number of requests transmitted to the attendance monitoring system: by tracking the movements of detected faces, we can generate one request per face rather than sending a request for each new detection, thereby lightening the system's load.
Secondly, the tracking algorithm also helps minimize the number of false positives. Instead of simply detecting a face and automatically assigning an identification to each new appearance, the system keeps track of identifications already made for a given face, thus avoiding the generation of repeated false identifications for the same person at different times. This approach, therefore, contributes to strengthening the system's accuracy while ensuring more efficient data and resource management.
To meet our performance constraints, we have opted to use the centroid-based object tracking algorithm [16].

Alignment and Cropping:
In this step, we calculate and then apply a geometric transformation (scaling, rotation, and translation) that aligns the facial landmarks (extracted from the detection step) of the source image to a standardized position. It produces a new image as output in which the facial landmarks are aligned and cropped, with the aim of increasing the accuracy of facial recognition. Researchers at Google have stated that the application of an alignment algorithm increased the accuracy of their facial recognition model from 98.87% to 99.63% [5]. Figure 3 illustrates an example of alignment.

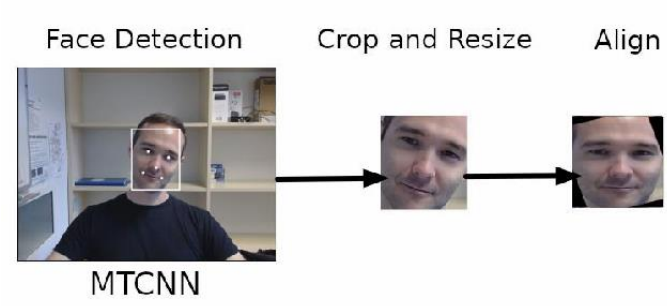


Figure 3: Example of alignment and cropping of a face [17]

An Embedded Intelligent System for Attendance Monitoring

TOUZENE Abderraouf, ABED Abdeljalil Wassim, USTHB University, Algeria

Representation:

A deep learning model is used to extract facial features to obtain a vector representation of the face. Figure 4 shows an example of a vector representation of a face.

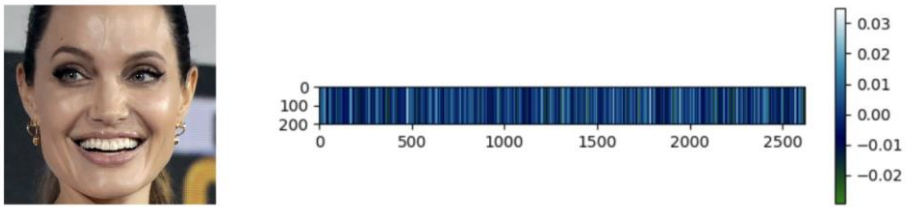


Figure 4: Example of face representation [9]

Computing Distances:
After the representation step, we proceed to calculate the distances between the extracted facial features and the features of faces stored in the database. The minimum distance is then compared to a pre-defined threshold experimentally to determine whether the face is recognized or not.

3. Experimental results

To deploy the described system on a Raspberry Pi, we need to consider the constraints imposed by the limited resources of this embedded system. Therefore, we followed the following steps:

3.1. Implementation details

- We first adopted an approach based on the use of threads to effectively separate video input management from the facial recognition process. This approach allowed optimal allocation of available resources, thus ensuring smooth processing of video streams without any frame loss despite hardware constraints.
- In order to minimize processing load while maintaining acceptable inference times and without compromising the system's accuracy, we decided to reduce the resolution of captured videos.
- ONNX Runtime is designed to provide high performance for executing deep learning models on various hardware platforms. By using techniques such as graph optimization, model quantization, and efficient thread management, ONNX Runtime aims to reduce inference time and improve computational efficiency. These optimizations are particularly important for deployments on devices like the Raspberry Pi, where resources are limited, and performance needs to be maximized.
- To make the use of our system simpler, dedicated services have been set up on the operating system to manage:

TOUZENE Abderraouf, ABED Abdeljalil Wassim, USTHB University, Algeria

• Network connection Startup: Connection startup can be done either via a local network, Wi-Fi, or a wired connection (Ethernet), where the Raspberry Pi is connected to other devices through the network, or in hotspot mode where it is directly connected with the host machine via Wi-Fi.
• Startup of the Facial Recognition System: A service has been created to ensure the automatic launch of the presence system scripts during the startup of the embedded system.
• Startup of the Configuration Interface: We have set up a configuration interface allowing users to adjust system parameters according to their specific needs to meet all scenarios, as well as configure the connection and ensure data transfer from the embedded system to the presence management system.

3.2. Attendance Management

For classroom attendance management, we developed two complementary applications using the MERN stack (MongoDB, Express.js, React, Node.js) for the web application and a file-based system for the desktop application.
The web application, built with React, offers a responsive and modern user interface for online attendance management. Teachers and administrators can view real-time attendance, generate reports, and manage student information. Data is centralized in a MongoDB database, providing flexible and efficient document management. The backend of the application, developed with Express.js and Node.js, handles user requests, authentication, and communication with the database, ensuring the security and integrity of information. This setup allows for real-time management and a smooth, secure user experience.
The desktop application, developed using Electron, is designed to work independently without requiring a constant internet connection. This application uses React for a consistent user interface with the web application. Unlike the web application, it stores data locally as files, allowing for local processing and data management without relying on a centralized database. Users can import attendance data captured by the facial recognition system, analyze this data locally, and export it for later use or synchronization with the web application. This approach is particularly useful for environments with limited connectivity, ensuring that teachers can manage and analyze attendance flexibly. By integrating these two applications, we provide a comprehensive solution for classroom attendance management, combining the benefits of centralized online management with the flexibility of local processing depending on network availability.

3.3. Conducted experiments and Obtained results

The dataset we used to test the performance of our system consists of 77 images. These images are photos faculty student. For the choice of detection and recognition models, we compared and tested several models. Initially, we chose the DeepFace module [9] which we evaluated using different configurations, particularly on two sets of data (77 and 7 student images). We conducted tests on five separate videos to assess the model's performance in different scenarios:
Scenario 1: features a student without any accessory covering part of their face, **Scenario 2**: features a single student wearing a cap, **Scenario 3**: features a single student wearing a hoodie, **Scenario 4**: features a single student whose facial feature vector is not in the database.

An Embedded Intelligent System for Attendance Monitoring

TOUZENE Abderraouf, ABED Abdeljalil Wassim, USTHB University, Algeria

Tables 1 and 2 present a comparison of the performance of some models implemented by DeepFace [9] across all the aforementioned scenarios.

Table 1: Comparison of the performance of models implemented by DeepFace with 7 student images.

Recognition	Detection	Arc Face [14]	Facenet 512	Facenet [11]	Dlib	VGG-Face [12]
yolov8 [8]	cosine	82.91%	52.99%	80.34%	11.97%	82.05%
	euclidean	82.05%	82.91%	82.05%	70.09%	82.05%
yunet [7]	cosine	5.13%	3.42%	2.56%	5.13%	5.13%
	euclidean	2.56%	5.98%	80.34%	5.13%	5.13%
Fast MTCNN	cosine	0.00%	65.81%	76.92%	72.41%	82.05%
	euclidean	0.00%	82.05%	80.34%	75.21%	82.05%
Dlib	cosine	82.05%	85%	76.92%	3.42%	35.90%
	euclidean	48.72%	5.98%	80.34%	5.13%	49.57%
SSD [3]	cosine	1.71%	85%	0.00%	0.00%	85%
	euclidean	0.00%	1.71%	0.00%	0.00%	0.85%
MTCNN [4]	cosine	82.05%	85%	9.40%	11.11%	17.09%
	euclidean	19.66%	8.55%	81.20%	14.53%	17.09%
Mediapipe [6]	cosine	17.95%	1.71%	20.51%	17.95%	18.80%
	euclidean	16.24%	80.34%	22.22%	17.95%	18.80%

Increasing the number of vectors clearly impacted the performance of the models. Therefore, we decided to try other models, namely Yunet [7] and SFace [15], since increasing the number of students from 7 to 77 did not impact the recognition rate as shown in Table 3.

3.4. Robustness of the system to Camera pose and Light conditions

We began by building a database comprising the facial characteristic vectors of 77 students, using their identity photos. These characteristic vectors were used as references for the identification. The experiments took place in two different practical work rooms with two groups of 6 and 9 students participated in this experiment. We conducted tests on three scenarios to evaluate the system's performance under real conditions.

An Embedded Intelligent System for Attendance Monitoring

TOUZENE Abderraouf, ABED Abdeljalil Wassim, USTHB University, Algeria

1. First Case: Optimal Viewing Angle with Poor Lighting Conditions. In this first experimental case, the lighting conditions were poor, resulting in dark faces of the subjects in the captured images. Despite these difficulties, the system detected the five entering students, demonstrating its robustness and ability to function effectively even in light conditions. Figure 5 shows some images of the recognized faces.
2. Second Case: Optimal Viewing Angle with Acceptable Lighting Conditions In this second case, the shooting angle was optimal, and the lighting conditions were acceptable. Under these favorable conditions, the system achieved accurate detection and successfully identified the six present individuals.
3. Third Case: Unfavorable Viewing Angle with Acceptable Lighting Conditions In this third case, although the lighting conditions were better than in the first case, the viewing angle was not optimal, making detection more difficult for the system. Despite this, the system was able to detect 6 from the 9 entering students, highlighting the importance of the viewing angle in the recognition process. Figure 5 shows the images some recognized faces.

Table 2: Comparison of the performance of models implemented by DeepFace with 77 student images.

Recognition / Detection		Arc Face [14]	Facenet 512	Facenet [11]	Dlib	VGG-Face [12]
yolov8 [8]	cosine	5,13%	0,00%	0,00%	2,56%	1,71%
	euclidean	1,71%	85%	0,00%	2,56%	1,71%
yunet [7]	cosine	4,27%	0,00%	0,85%	4,27%	1,71%
	euclidean	85%	85%	0,00%	5,13%	1,71%
Fast MTCNN	cosine	4,31%	0,00%	0,00%	3,45%	1,72%
	euclidean	2,59%	1,71%	0,00%	10,34%	1,72%
Dlib	cosine	85%	0,00%	0,00%	0,00%	2,56%
	euclidean	85%	85%	0,00%	1,71%	2,56%
SSD [3]	cosine	1,71%	0,00%	0,00%	0,00%	0,85%
	euclidean	0,00%	85%	0,00%	0,00%	0,85%
MTCNN [4]	cosine	2,56%	0,00%	0,00%	4,27%	1,71%
	euclidean	1,71%	85%	0,00%	4,27%	1,71%
Mediapipe [6]	cosine	17,09%	0,85%	17,09%	17,09%	17,09%
	euclidean	16,24%	17,09%	17,09%	17,09%	17,09%

An Embedded Intelligent System for Attendance Monitoring

TOUZENE Abderraouf, ABED Abdeljalil Wassim, USTHB University, Algeria

Conclusion

In this paper we presented a facial recognition-based attendance monitoring system. Initially, we focused our efforts on deploying and optimizing pretrained models on the embedded system, emphasizing their efficient integration and adaptation to specific hardware constraints. This enabled us to achieve satisfactory facial recognition performance under limited resource conditions.
Subsequently, we designed and developed a user-friendly web application for managing the attendance system. The interaction between the embedded system and the web application facilitated real-time attendance management, making data access and system administration easier.
We also developed a desktop application to provide an alternative in case of unavailability of a computer network.
Future work will focus on using more advanced optimization techniques to optimize resource management by the operating system and improving the quality of images acquired by the system, either by using a more powerful camera module or by developing an optimized super-resolution model suitable for embedded systems. These enhancements would contribute to the overall robustness and efficiency of the attendance monitoring system, making it more reliable and effective in various scenarios.

Scenarios	Correct Recognition	Incorrect Recognition	Unknown	Detected Faces	ACC	FAR	FRR
Scenario 1	14	0	3	17	82%	0%	0.18%
Scenario 2	3	0	9	12	25%	0%	75%
Scenario 3	2	0	10	12	17%	0%	83%
Scenario 4	0	0	24	24	100%	0%	0%

Table 3: Results of tests of the Yunet and SFace models with 77 and 7 student images where (ACC):accuracy, (FAR): false acceptance rate , and (FRR): false rejection rate.

An Embedded Intelligent System for Attendance Monitoring

TOUZENE Abderraouf, ABED Abdeljalil Wassim, USTHB University, Algeria

Figure 5. Some images from the first and the third scenario

References

[1] A. Sharifara, M. S. Mohd Rahim, and Y. Anisi, "A general review of human face detection including a study of neural networks and haar feature-based cascade classifier in face detection," in 2014 International Symposium on Biometrics and Security Technologies (ISBAST), (Kuala Lumpur, Malaysia), pp. 73–78, 2014.
[2] S. G. Joshi and R. Vig, "Histograms of orientation gradient investigation for static hand gestures," in International Conference on Computing, Communication & Automation, (Greater Noida, India), pp. 1100–1103, 2015.
[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," UNC Chapel Hill, 2016.

An Embedded Intelligent System for Attendance Monitoring

TOUZENE Abderraouf, ABED Abdeljalil Wassim, USTHB University, Algeria

[4] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Mtcnn: Face detection alignment," IEEE Signal Processing Letters, 2016.
[5] J. Deng, J. Guo, Y. Zhou, J. Yu, I. Kotsia, and S. Zafeiriou, "Retinaface: Single-stage dense face localisation in the wild," Imperial College London, 2019.
[6] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, "Mediapipe: A framework for building perception pipelines," Google Research, 2020.
[7] W.Wu, H. Peng, and S. Yu, "Yunet: A tiny millisecond-level face detector," Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China, vol. 20, no. 5, pp. 656–665, 2023.
[8] D. Reis, J. Kupec, J. Hong, and A. Daoudi, "Real-time flying object detection with yolov8," Georgia Institute of Technology, 2023.
[9] S. I. Serengil and A. Ozpinar, "Lightface: A hybrid deep face recognition framework," in 2020 Innovations in Intelligent Systems and Applications Conference (ASYU), (Istanbul, Turkey), pp. 1–5, 2020.
[10] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human level performance in face verification," in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 1701–1708, 2014.
[11] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," arXiv preprint arXiv:1503.03832, 2015.
[12] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," Visual Geometry Group, Department of Engineering Science, University of Oxford, 2015.
[13] G. B. Huang, M. Mattar, H. Lee, and E. Learned-Miller, "Learning to align from scratch," in NIPS, 2012.
[14] J. Deng, J. Guo, J. Yang, N. Xue, I. Kotsia, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," arXiv preprint arXiv:1801.07698, 2015.
[15] Y. Zhong, W. Deng, J. Hu, D. Zhao, X. Li, and D. Wen, "Sface: Sigmoid-constrained hypersphere loss for robust face recognition," 2021.
[16] J. Nascimento, A. Abrantes, and J. Marques, "An algorithm for centroid-based tracking of moving objects," in 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258), vol. 6, pp. 3305–3308 vol.6, 1999.
[17] M. L. Canellas, C. Alvarez Casado, L. Nguyen, and M. B. Lopez, "Improving depression estimation from facial videos with face alignment, training optimization and scheduling," 2022.

Visual Computing Magazine

Vol 2, Issue 1, 2024

Content