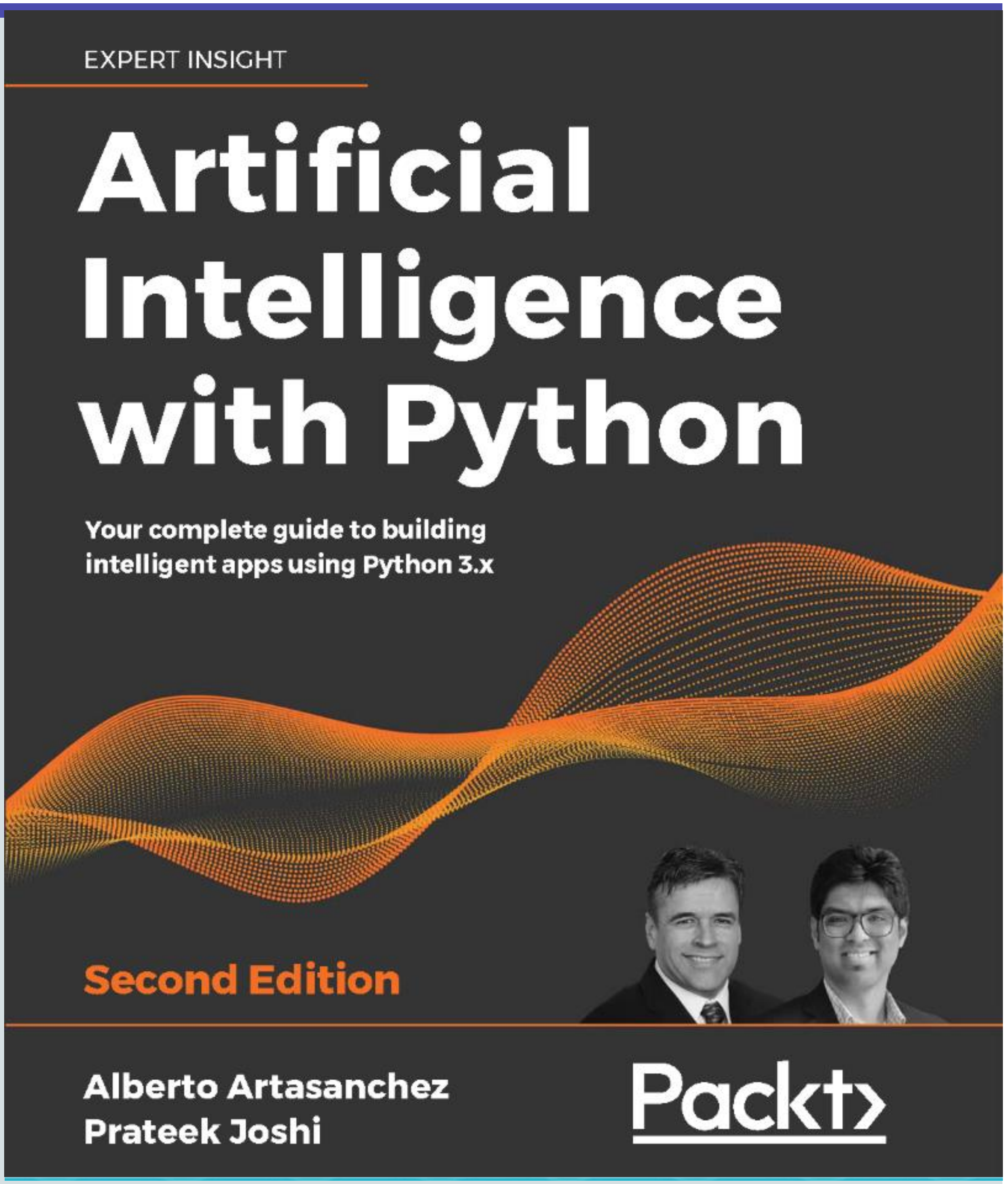
University of South-Eastern Norway
School of Business

Lecture 7: Computer Vision

Sinuo Wu
Course: AI for Business Applications (AI3000)



EXPERT INSIGHT

Artificial Intelligence with Python

Your complete guide to building intelligent apps using Python 3.x

Second Edition

Alberto Artasanchez
Prateek Joshi

Packt

University of South-Eastern Norway
School of Business

Do it before we start:

Download Data From Canvas – AI3000 - Files – Day7 Practice

Sinuo Wu
Course: AI for Business Applications (AI3000)

Exam Info

Innlev. dato	FLOW- type	Emnekode	Emne	Type	Timer/ Utlev	Rom	Ant stud på campus
Mandag 20/11	FLOWlock	AI3000R-1	Artificial Intelligence for Business Applications	S	4 timer	**?**	191

Reflection

- More math and go deeper to algorithms?

https://www.amazon.com/Machine-Learning-Algorithms-reference-algorithms/dp/1785889621

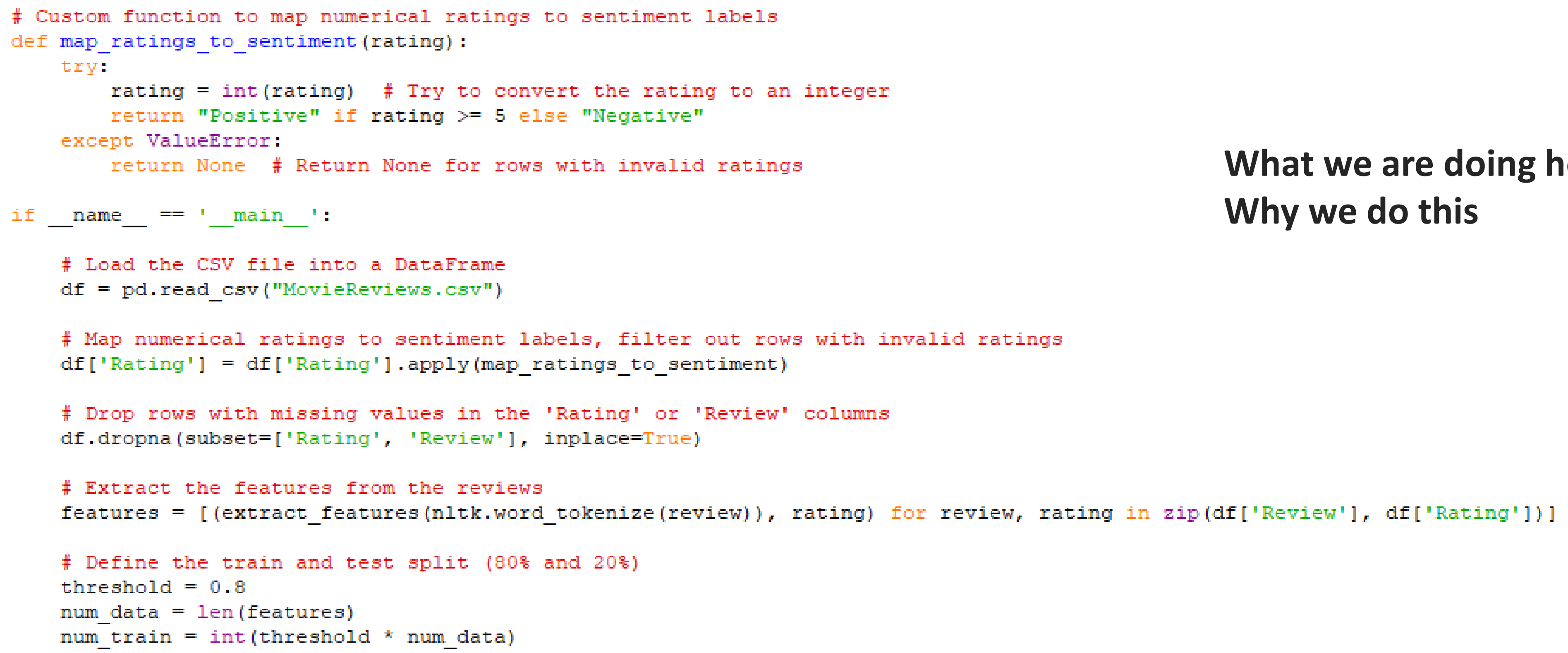
- Focus on coding? – Purpose of our course – Solve practical problems

- Clarify the goal of each exercise – Very good suggestion

- More Practical cases – If you save the model and use it.

Reflection

1. Explain the coding
2. How this model can be applied



```python
# Custom function to map numerical ratings to sentiment labels
def map_ratings_to_sentiment(rating):
    try:
        rating = int(rating)  # Try to convert the rating to an integer
        return "Positive" if rating >= 5 else "Negative"
    except ValueError:
        return None  # Return None for rows with invalid ratings

if __name__ == '__main__':

    # Load the CSV file into a DataFrame
    df = pd.read_csv("MovieReviews.csv")

    # Map numerical ratings to sentiment labels, filter out rows with invalid ratings
    df['Rating'] = df['Rating'].apply(map_ratings_to_sentiment)

    # Drop rows with missing values in the 'Rating' or 'Review' columns
    df.dropna(subset=['Rating', 'Review'], inplace=True)

    # Extract the features from the reviews
    features = [(extract_features(nltk.word_tokenize(review)), rating) for review, rating in zip(df['Review'], df['Rating'])]

    # Define the train and test split (80% and 20%)
    threshold = 0.8
    num_data = len(features)
    num_train = int(threshold * num_data)
```

What we are doing here
Why we do this

Data here is just for model training! You apply the model for new data!

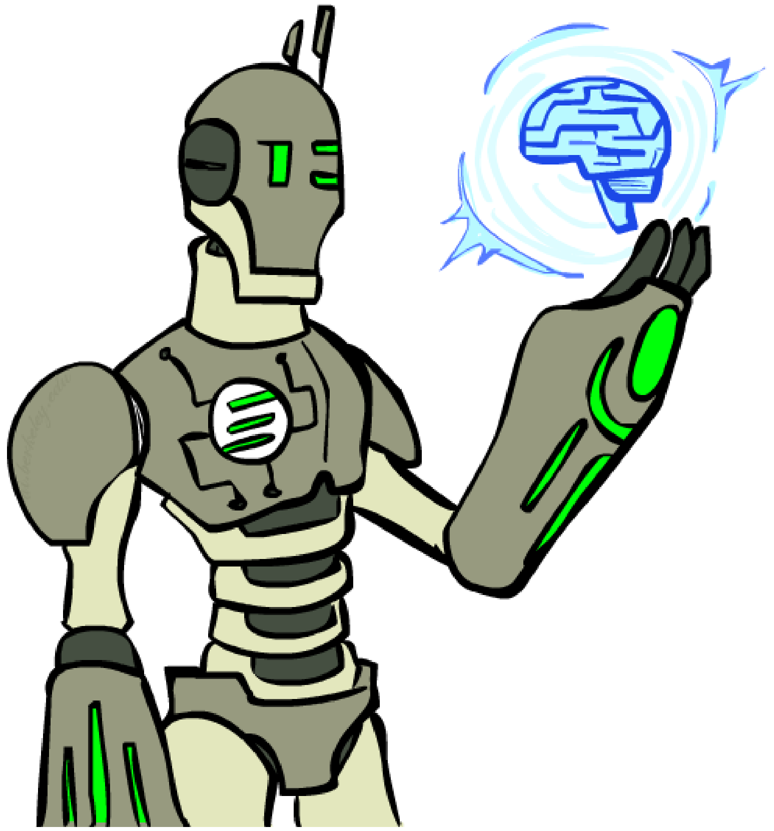
University of South-Eastern Norway
School of Business

Quiz

- Enter room number or Scan the QR code

Today

- Intro to Computer Vision
- Applications of Computer vision
- Image Processing
- Image Feature Extraction



Computer Vision

Computer vision

- Computer vision is a multidisciplinary field that enables computers to interpret and understand the visual world through digital images or videos.
- Its primary goal is to replicate the human visual system's ability to extract meaningful information from visual data.
- Computer vision aims to teach machines to "see" and comprehend the visual content of images and videos.

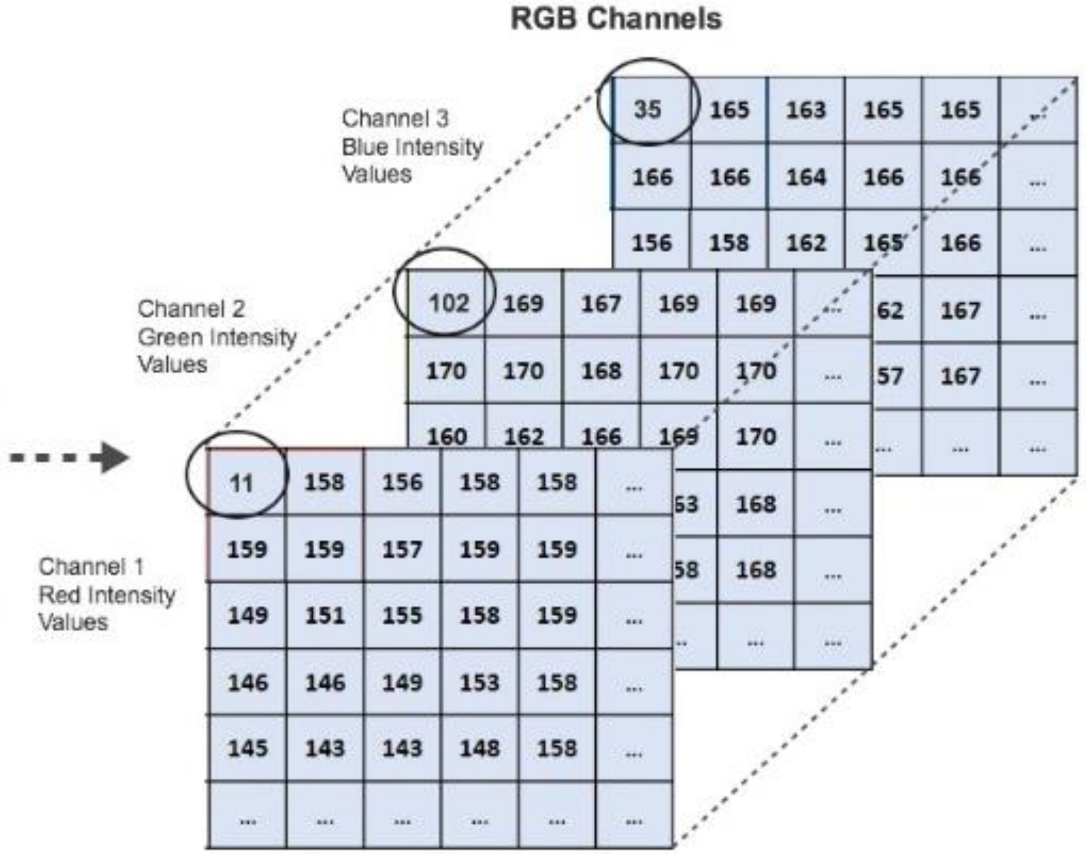
Computer vision

- **Image Processing:** The manipulation and enhancement of images to improve their quality or extract useful information.
- **Feature Extraction:** Identifying and extracting relevant patterns, shapes, or objects from images.
- **Object Detection and Recognition:** Locating and classifying objects or specific features within images or video frames.
- **Image Segmentation:** Dividing an image into meaningful regions or segments based on shared characteristics.
- **Motion Analysis:** Analyzing the movement and changes in visual content over time, often used in video processing.

Every image tells a story

- Goal of computer vision: perceive the “story” behind the picture.
- Compute properties of the world
  – 3D shape
  – Color
  – Names of people or objects
  – What happened?

Human eyes VS Computer



Can computers match human perception?

- **Yes and no (mainly no)**
  - Computers can be better at “easy” things
  - Humans are better at “hard” things

- **But huge progress**
  - Accelerating in the last ten years due to deep learning
  - What is considered “hard” keeps changing

University of South-Eastern Norway
School of Business

Challenges

Viewpoint variation

illumination

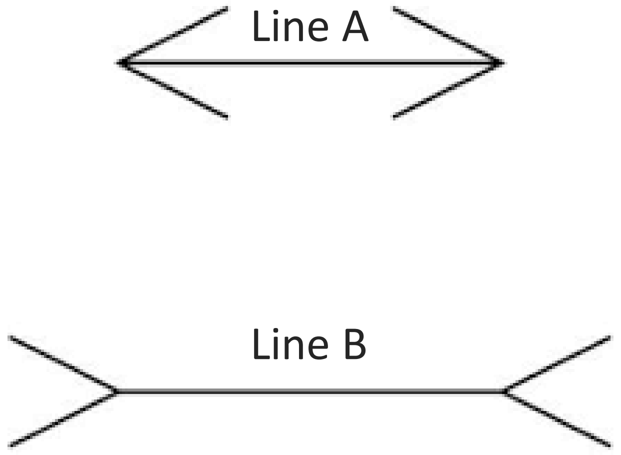
Scale

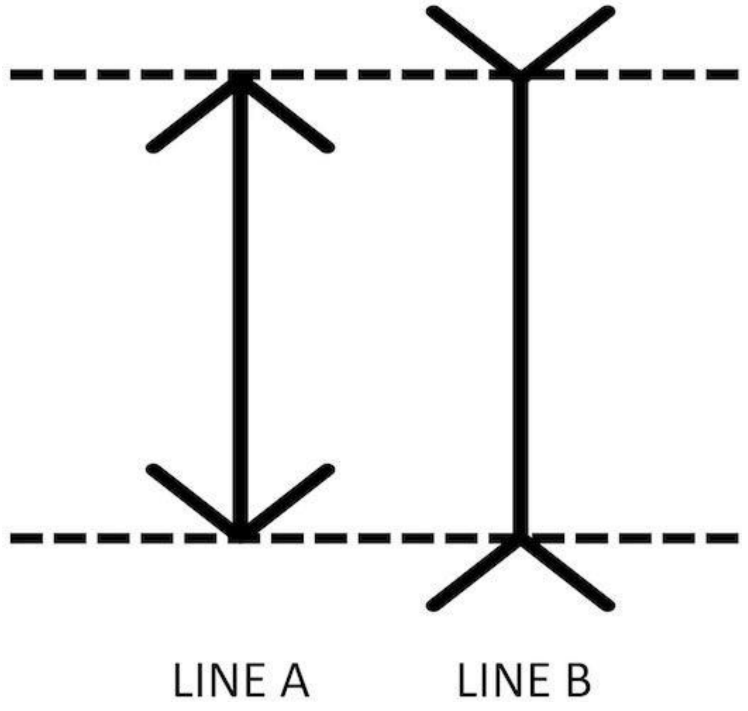
Shape

Ambiguity

Interference

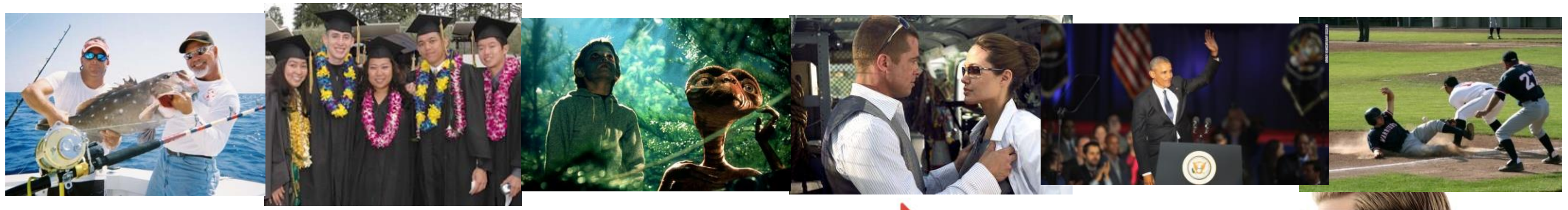
Challenges





Why study computer vision?

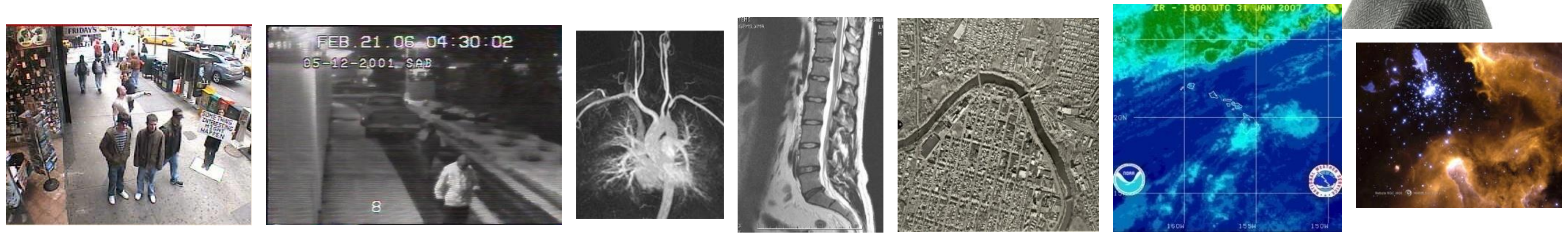
- Billions of images/videos captured per day
- Huge number of applications



flickr

Google Photos

YouTube Broadcast Yourself™

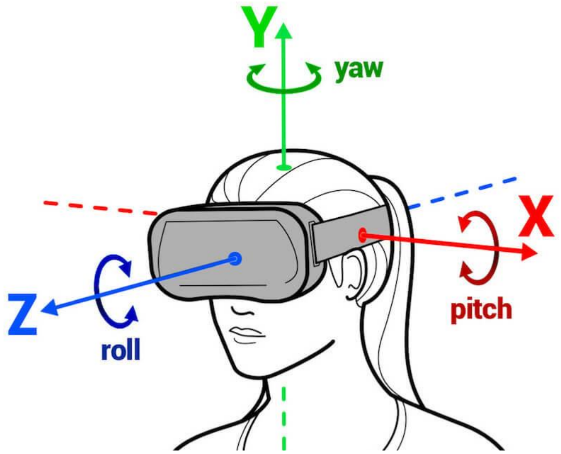


Object detection

Waymo

EasyMile

Object tracking

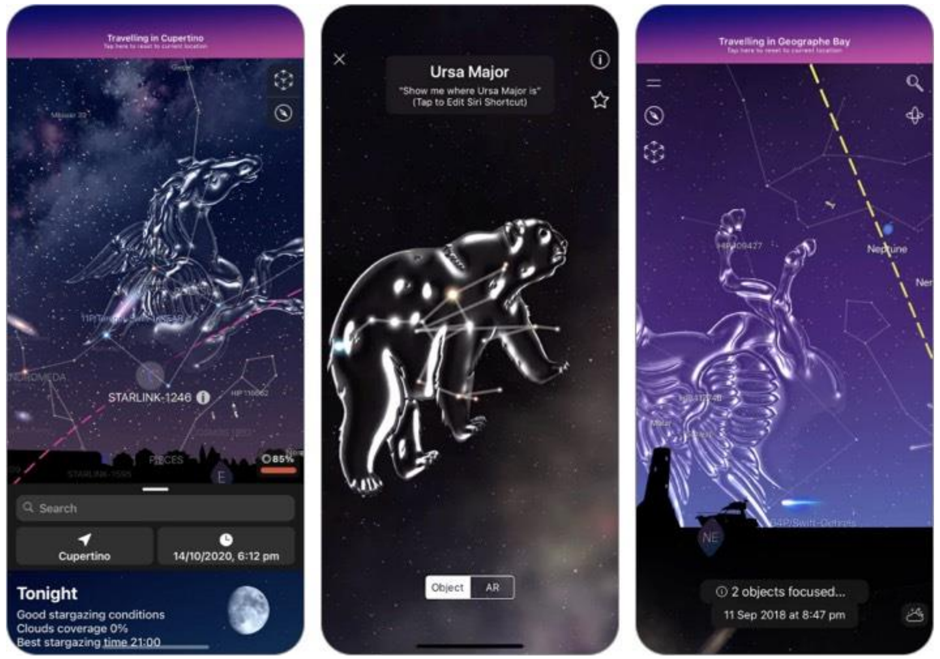


6DoF head tracking

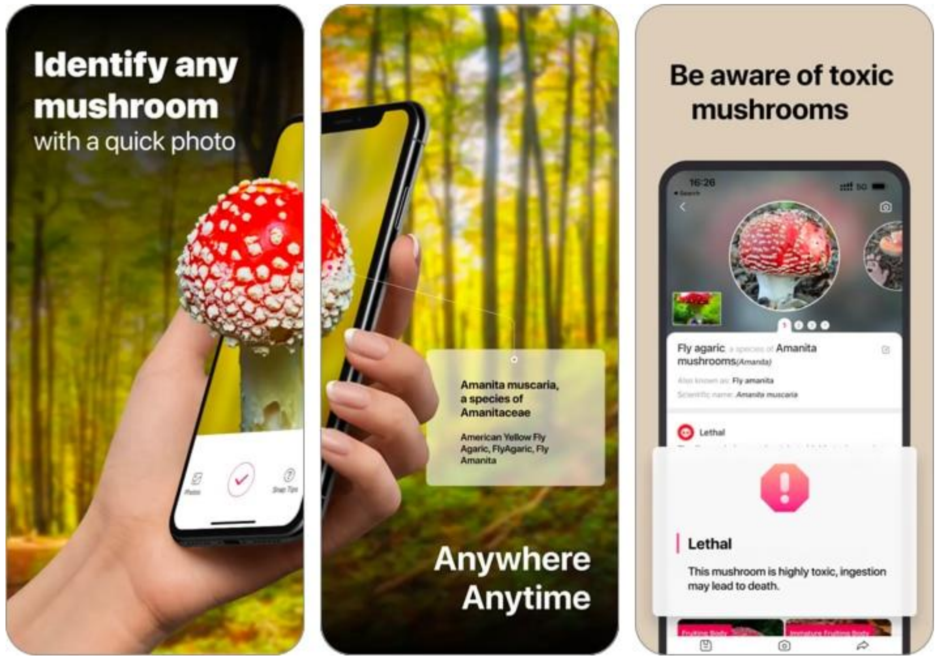
Hand & body tracking

3D-360 video capture

Object identification

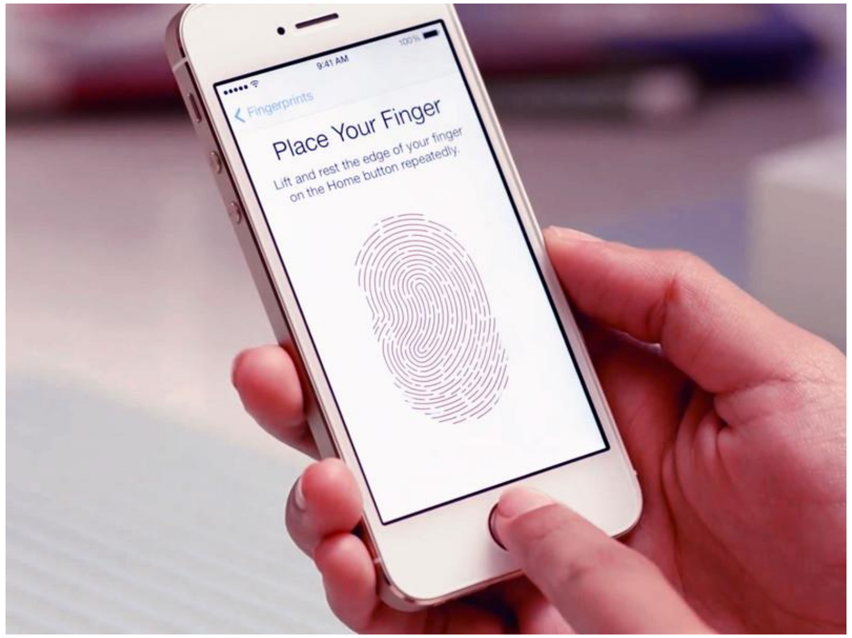


Stargazing


Identify any mushroom
with a quick photo
Amanita muscaria, a species of Amanitaceae
American Yellow Fly Agaric, FlyAgaric, Fly Amanita
Anywhere Anytime
Be aware of toxic mushrooms
Fly agaric
Lethal
This mushroom is highly toxic, ingestion may lead to death.

Mushroom identification

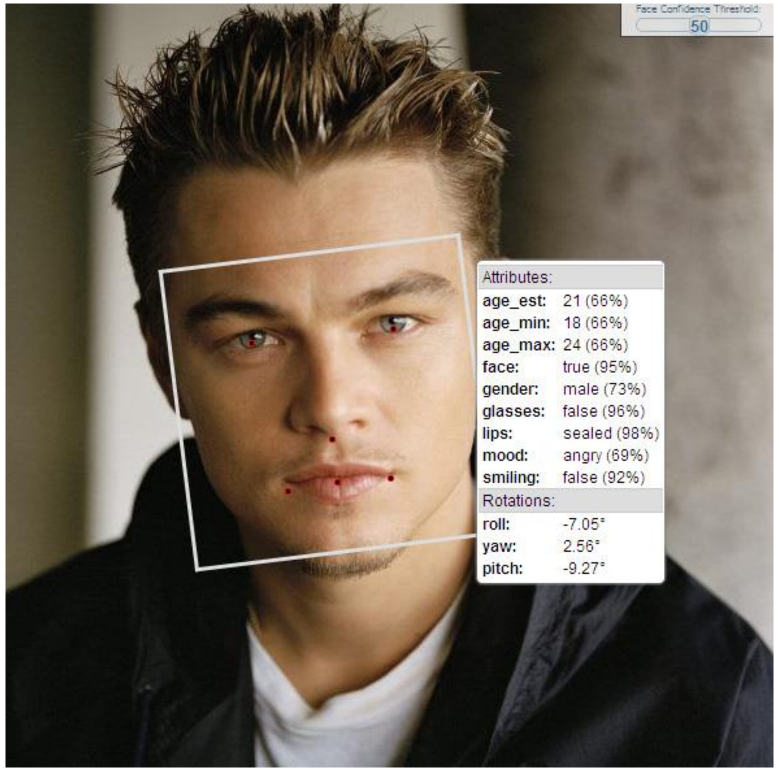
Object verification


Fingerprints
Place Your Finger
Lift and rest the edge of your finger on the Home button repeatedly.

Fingerprint scanners on many new smartphones and other devices

Face unlock on Phone
See also http://www.sensiblevisio.com/

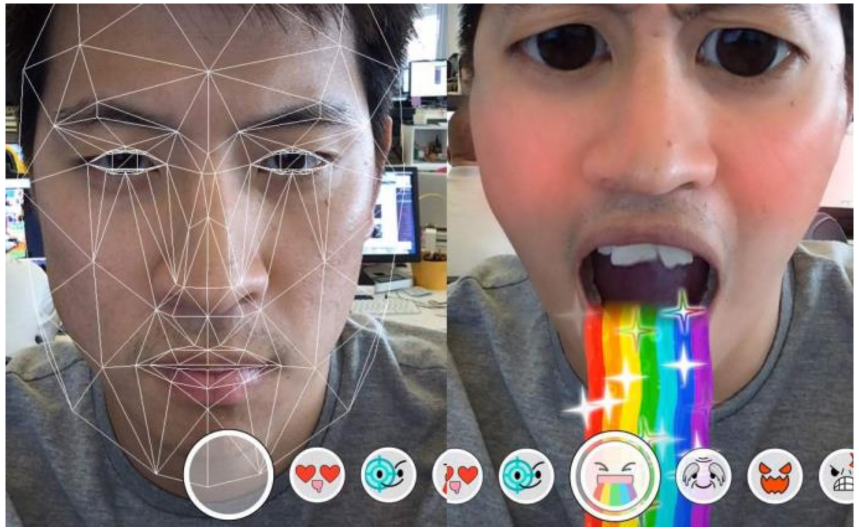
Face detection and analysis


Face Confidence Threshold: 50

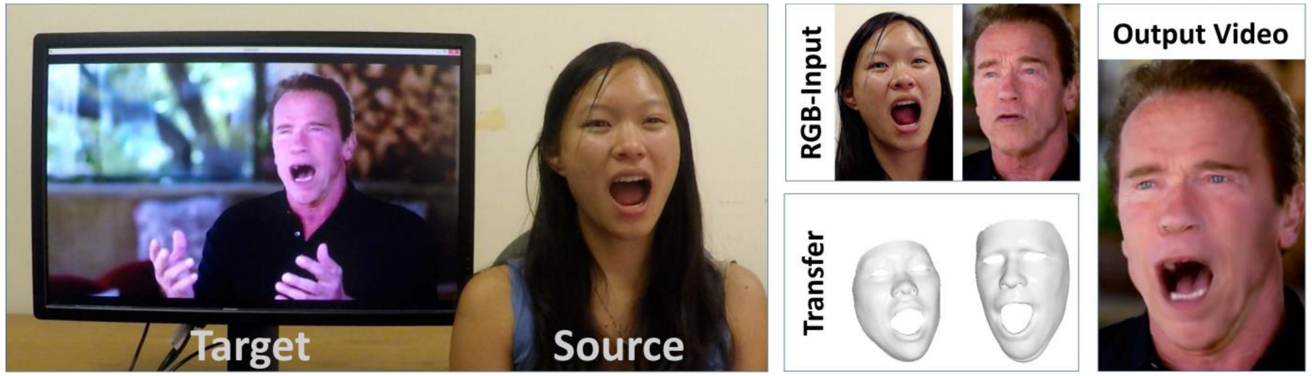
```
Attributes:
age_est:   21 (66%)
age_min:   18 (66%)
age_max:   24 (66%)
face:      true (95%)
gender:    male (73%)
glasses:   false (96%)
lips:      sealed (98%)
mood:      angry (69%)
smiling:   false (92%)
Rotations:
roll:      -7.05°
yaw:       2.56°
pitch:     -9.27°
```

- Nearly all cameras detect faces in real time

3D face tracking w/ consumer cameras



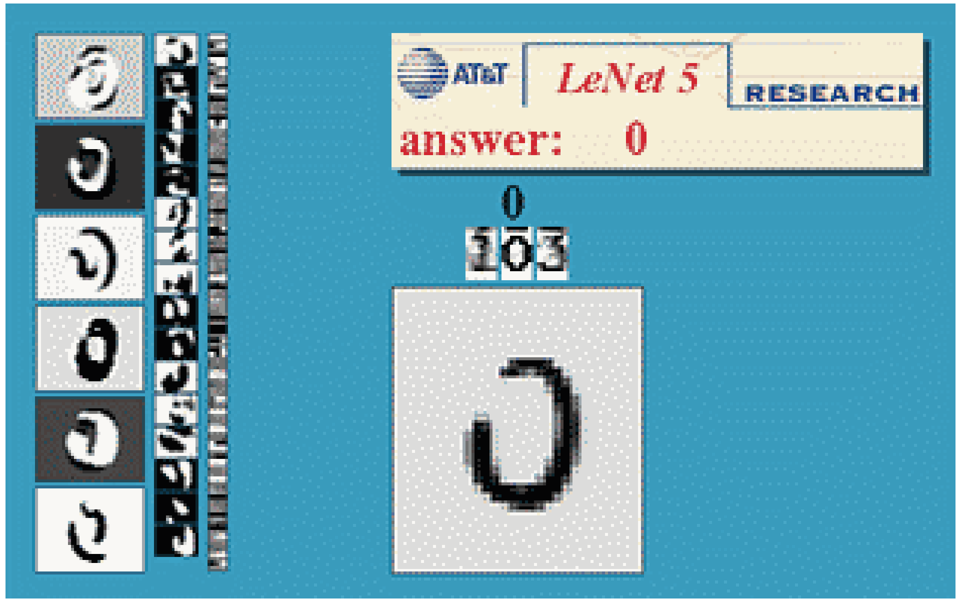
Snapchat Lenses



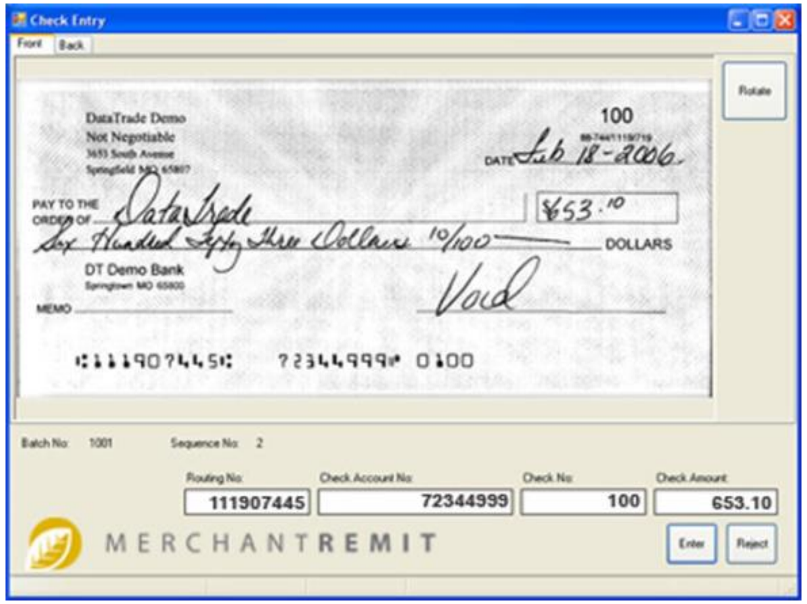
Face2Face system (Thies et al.)

Optical character recognition (OCR)

- If you have a scanner, it probably came with OCR software



Digit recognition, AT&T labs (1990's)
http://yann.lecun.com/exdb/lenet/


Check Entry

Front | Back

DataTrade Demo
Not Negotiable
3655 South Avenue
Springfield MO 65807

100

DATE Feb 18-2006

PAY TO THE ORDER OF DataTrade

$653.10

Six Hundred Fifty Three Dollars 10/100 DOLLARS

DT Demo Bank
Springtown MO 65800

MEMO

Void

⑆111907445⑆ 72344999⑈ 0100

Batch No: 1001 Sequence No: 2

Routing No:	Check Account No:	Check No:	Check Amount:
111907445	72344999	100	653.10

Automatic check processing

Robotics

NASA’s Mars Curiosity Rover
https://en.wikipedia.org/wiki/Curiosity_(rover)

Amazon Picking Challenge
http://www.robocup2016.org/en/events/amazon-picking-challenge/

Current state of the art

- You just saw many examples of current systems.
  - Many of these are less than 10 years old

- Computer vision is an active research area, and rapidly changing
  - Many new apps in the next 5 years
  - Deep learning powering many modern applications

- Many startups across a dizzying array of areas
  - Deep learning, robotics, autonomous vehicles, medical imaging, construction, inspection, VR/AR, …

Evolution

- **Early Days (1960s-1970s):** Back then, people started making computer programs to work with pictures. But these early programs couldn't handle complicated images very well.
- **Machine Learning (1980s-1990s):** Machine learning techniques began to be applied. This led to the development of algorithms for tasks such as **object recognition and image classification**.
- **Deep Learning (2000s):** In the 2000s, deep learning techniques emerged, which use neural networks to learn patterns in images and videos. These algorithms have significantly improved **the accuracy of computer vision** tasks such as object detection and facial recognition.
- **Real-Time (Recent Years):** Today, computers can do all of this in real-time. This means they can do it super fast, which is useful for things like self-driving cars, drones, and robots.

Have a Break!

Image Processing

Image processing

- **Noise Reduction:** Images often contain noise, which is unwanted random variations in pixel values. Noise reduction techniques, such as filtering, are used to improve image quality. *Does not always mean clearer!
- **Image Enhancement:** Enhancement techniques can be applied to improve the visibility of certain features in an image, such as contrast stretching or histogram equalization.
- **Image Scaling and Resizing:** Resizing images can be done to fit them into specific dimensions or reduce file size. Interpolation methods are used to resample pixel values when resizing.

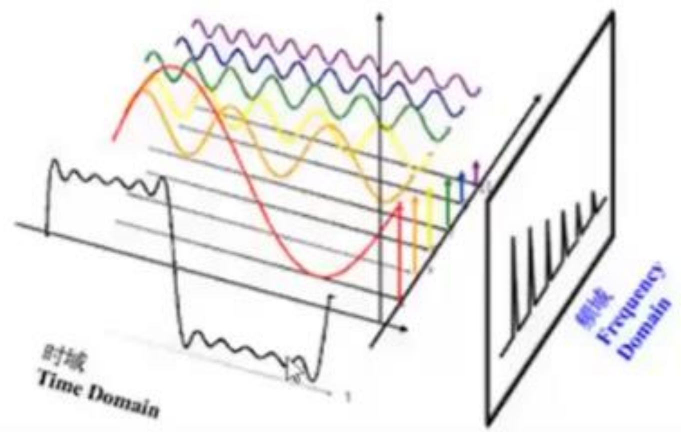
Library

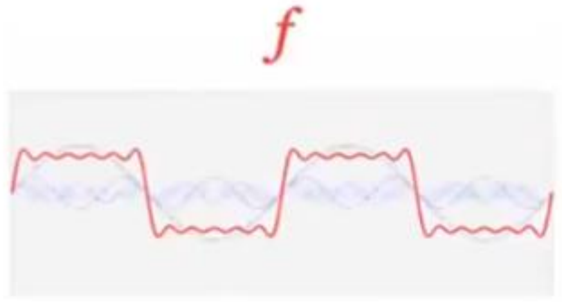
- OpenCV (OpenSource Computer Vision Library) is a versatile, open-source software library for computer vision and machine learning tasks.
- OpenCV is used in diverse applications, including image and video analysis, object detection, feature extraction, 3D vision, augmented reality, facial recognition, and more.
- Offers a rich set of functions and algorithms for image and video processing, enhancing tasks like filtering, resizing, and enhancement.
- Pip install opencv-python

Mathematical basis

- Gaussian Blur and Fourier Transform

$$A\sin(\omega x) + B\cos(\omega x)$$





$$a_n\cos(nx) + b_n\sin(nx)$$

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots\ldots$$

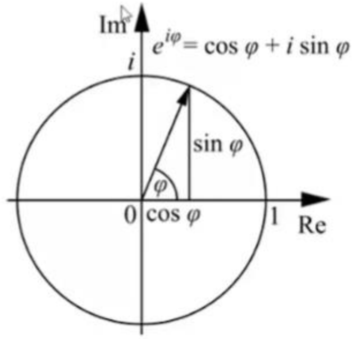
$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!}\ldots$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!}\ldots.$$

欧拉公式

$$e^{ix} = \cos x + i\sin x$$

$$e^{\pi i} + 1 = 0$$



https://www.amazon.com/Machine-Learning-Algorithms-reference-algorithms/dp/1785889621

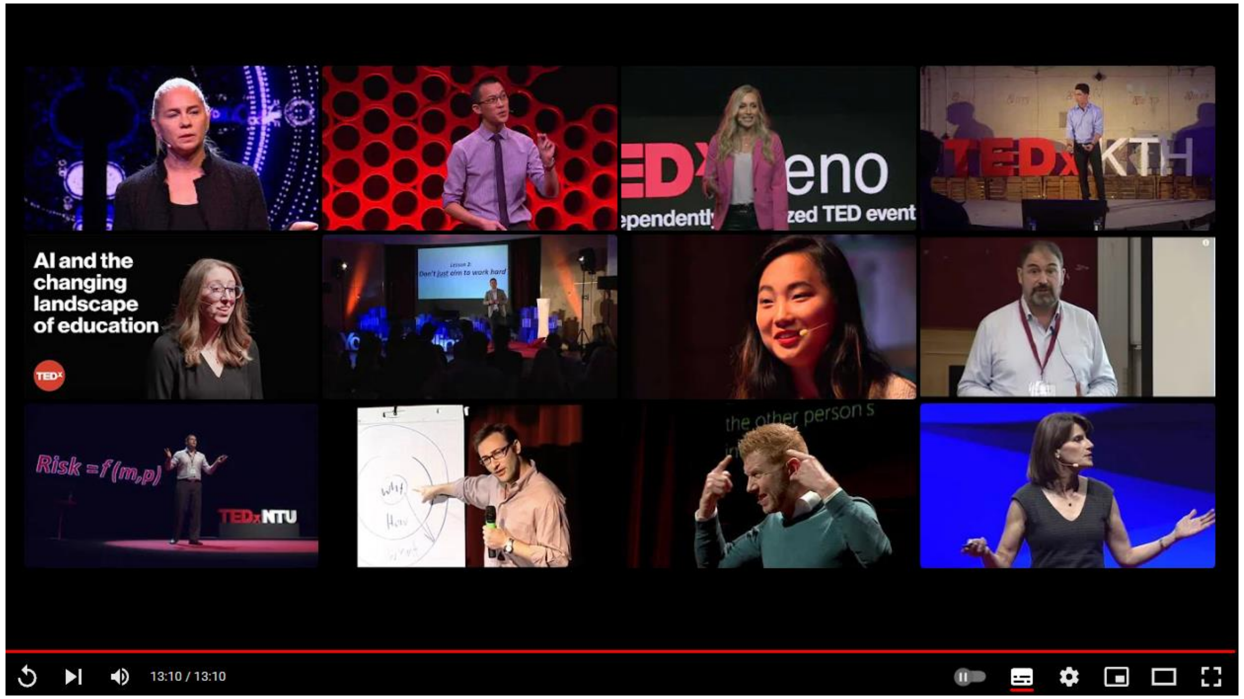
Noise Reduction

- **Try it**

```
import cv2
import numpy as np

# Load the noisy image
noisy_image = cv2.imread('noisy_image.jpg')

# Apply Gaussian blur for noise reduction
kernel_size = (5, 5)
blurred_image = cv2.GaussianBlur(noisy_image, kernel_size, 0)

# Save the denoised image
cv2.imwrite('denoised_image.jpg', blurred_image)

# Display the original and denoised images
cv2.imshow('Original Image', noisy_image)
cv2.imshow('Denoised Image', blurred_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Note: Karnel is the dimensions of a small matrix (also called a kernel or filter) that is used to perform various operations, such as blurring, sharpening, or edge detection, on an image.

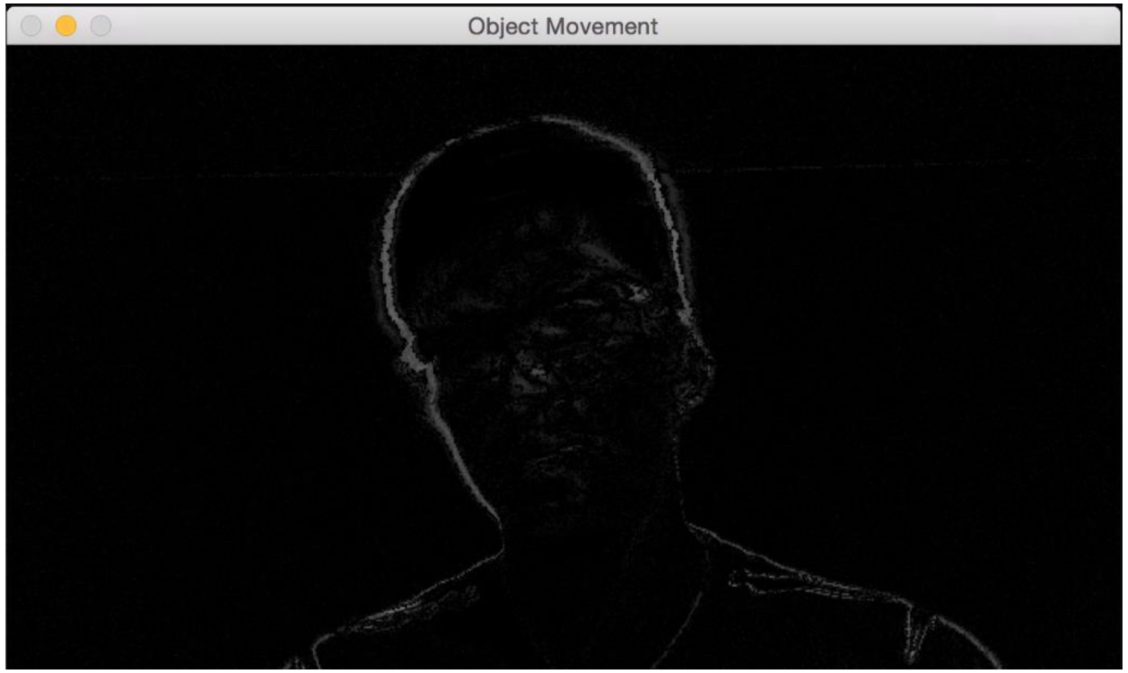
Inspiration

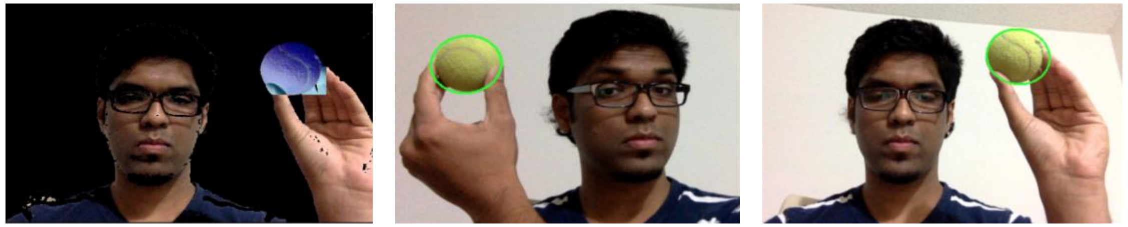


https://www.youtube.com/watch?v=40riCqvRoMs

University of South-Eastern Norway
School of Business

Object tracking





```
import cv2

# Compute the frame differences
def frame_diff(prev_frame, cur_frame, next_frame):
    # Difference between the current frame and the next frame
    diff_frames_1 = cv2.absdiff(next_frame, cur_frame)

    # Difference between the current frame and the previous frame
    diff_frames_2 = cv2.absdiff(cur_frame, prev_frame)

    return cv2.bitwise_and(diff_frames_1, diff_frames_2)

# Define a function to get the current frame from the webcam
def get_frame(cap, scaling_factor):
    # Read the current frame from the video capture object
    _, frame = cap.read()

    # Resize the image
    frame = cv2.resize(frame, None, fx=scaling_factor,
            fy=scaling_factor, interpolation=cv2.INTER_AREA)

    # Convert to grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)

    return gray

if __name__=='__main__':
    # Define the video capture object
    cap = cv2.VideoCapture(0)

    # Define the scaling factor for the images
    scaling_factor = 0.5

    # Grab the current frame
    prev_frame = get_frame(cap, scaling_factor)

    # Grab the next frame
    cur_frame = get_frame(cap, scaling_factor)

    # Grab the frame after that
    next_frame = get_frame(cap, scaling_factor)

    # Keep reading the frames from the webcam
    # until the user hits the 'Esc' key
    while True:
        cv2.imshow('Object Movement', frame_diff(prev_frame,
                cur_frame, next_frame))

        prev_frame = cur_frame
        cur_frame = next_frame
        next_frame = get_frame(cap, scaling_factor)

        key = cv2.waitKey(10)
        if key == 27:
            break

    # Close all the windows
    cv2.destroyAllWindows()
```

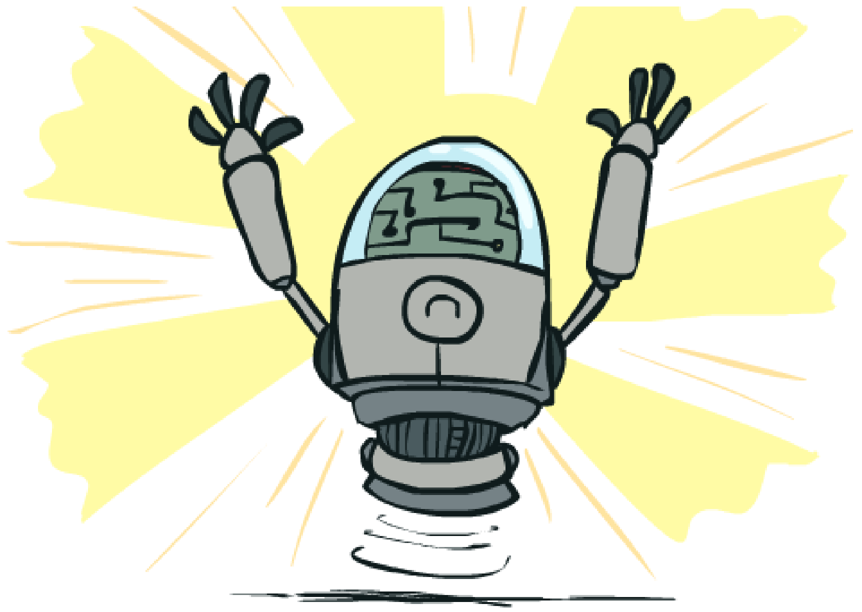
Note: **Frame difference** is one of the simplest techniques that can be applied to identify the moving objects

CAMShift Algorithm

1. **Initialization**: Begin by selecting an initial region of interest (ROI) around the object you want to track. This region is typically defined using a bounding box.
2. **Histogram Creation**: Create a color histogram based on the color information within the selected ROI. This histogram represents the object's color distribution.
3. **Back projection**: Calculate a back projection of the histogram. In this step, each pixel in the entire image is assigned a value indicating how closely it matches the colors in the histogram. This helps locate the object in subsequent frames.
4. **Mean Shift Tracking**: Apply the Mean Shift algorithm to iteratively find the peak (mode) of the back projection. This involves shifting the region towards the peak's location based on the similarity of pixel colors to the histogram.
5. **Adaptive Sizing and Orientation**: CAMShift continuously adapts the size and orientation of the tracking window based on the object's characteristics. It computes an oriented bounding box to enclose the object, considering changes in scale and orientation.
6. **Repeating Tracking**: Repeatedly perform the Mean Shift tracking and adaptive sizing and orientation steps for each frame in a video sequence to continually update the object's position, size, and orientation.
7. **Termination**: The tracking process continues until the end of the video sequence or until the user decides to stop it.

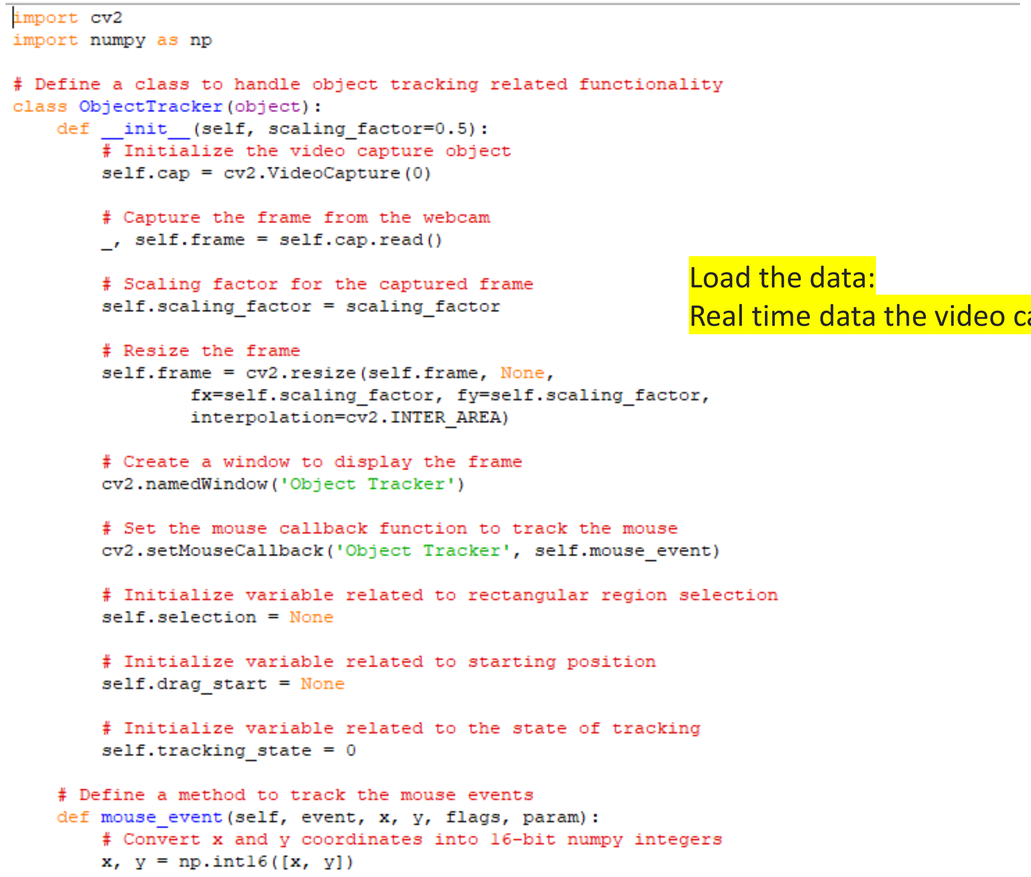
Practice

Goal: Successfully run the coding and track the object



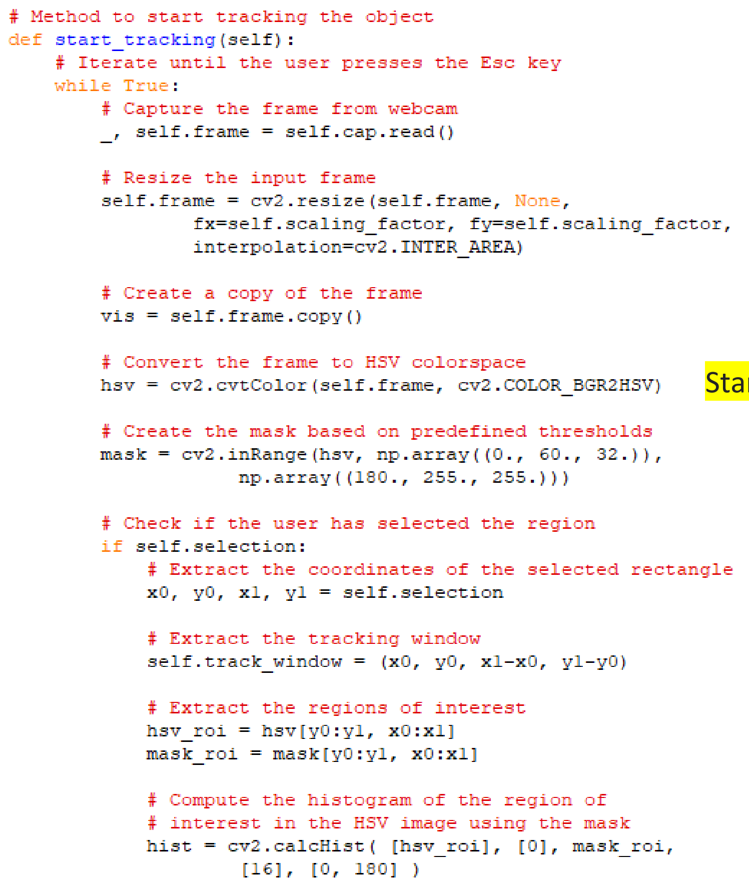
University of South-Eastern Norway
School of Business

Coding



```python
import cv2
import numpy as np

# Define a class to handle object tracking related functionality
class ObjectTracker(object):
    def __init__(self, scaling_factor=0.5):
        # Initialize the video capture object
        self.cap = cv2.VideoCapture(0)

        # Capture the frame from the webcam
        _, self.frame = self.cap.read()

        # Scaling factor for the captured frame
        self.scaling_factor = scaling_factor

        # Resize the frame
        self.frame = cv2.resize(self.frame, None,
                fx=self.scaling_factor, fy=self.scaling_factor,
                interpolation=cv2.INTER_AREA)

        # Create a window to display the frame
        cv2.namedWindow('Object Tracker')

        # Set the mouse callback function to track the mouse
        cv2.setMouseCallback('Object Tracker', self.mouse_event)

        # Initialize variable related to rectangular region selection
        self.selection = None

        # Initialize variable related to starting position
        self.drag_start = None

        # Initialize variable related to the state of tracking
        self.tracking_state = 0

    # Define a method to track the mouse events
    def mouse_event(self, event, x, y, flags, param):
        # Convert x and y coordinates into 16-bit numpy integers
        x, y = np.int16([x, y])
```

Load the data:
Real time data the video captured



```python
# Method to start tracking the object
def start_tracking(self):
    # Iterate until the user presses the Esc key
    while True:
        # Capture the frame from webcam
        _, self.frame = self.cap.read()

        # Resize the input frame
        self.frame = cv2.resize(self.frame, None,
                fx=self.scaling_factor, fy=self.scaling_factor,
                interpolation=cv2.INTER_AREA)

        # Create a copy of the frame
        vis = self.frame.copy()

        # Convert the frame to HSV colorspace
        hsv = cv2.cvtColor(self.frame, cv2.COLOR_BGR2HSV)

        # Create the mask based on predefined thresholds
        mask = cv2.inRange(hsv, np.array((0., 60., 32.)),
                    np.array((180., 255., 255.)))

        # Check if the user has selected the region
        if self.selection:
            # Extract the coordinates of the selected rectangle
            x0, y0, x1, y1 = self.selection

            # Extract the tracking window
            self.track_window = (x0, y0, x1-x0, y1-y0)

            # Extract the regions of interest
            hsv_roi = hsv[y0:y1, x0:x1]
            mask_roi = mask[y0:y1, x0:x1]

            # Compute the histogram of the region of
            # interest in the HSV image using the mask
            hist = cv2.calcHist( [hsv_roi], [0], mask_roi,
                    [16], [0, 180] )
```

Standardization

It allows the user to select a region of interest (ROI) by clicking and dragging the mouse

Have a Break!

Image Feature Extraction

Feature Extraction

- Feature extraction is a critical step in computer vision, where the goal is to identify relevant and informative characteristics within an image.

- These features serve as a basis for further analysis, such as object detection, image classification, or image segmentation.

- Feature extraction methods can be broadly categorized into handcrafted feature extraction and learned feature extraction using deep learning techniques.

Features in Image

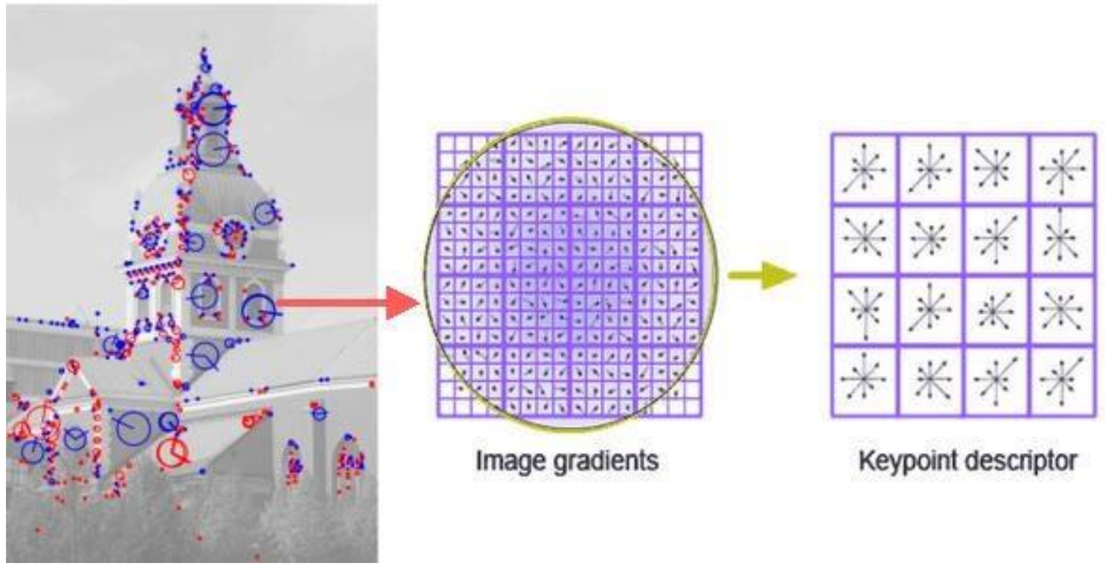
- Features

Color features

Geometric features: Edge, Comer, Blob

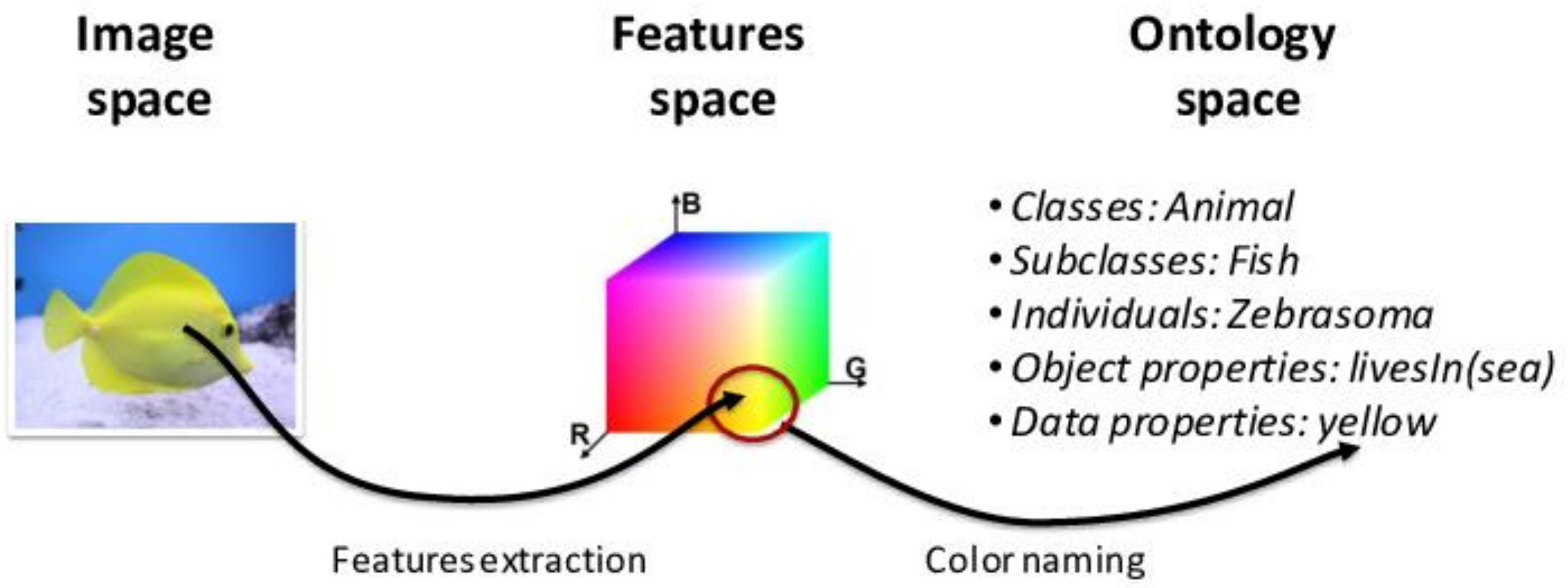
Texture features

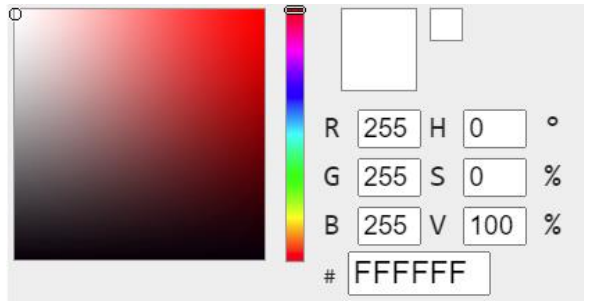
Widely applied algorithms

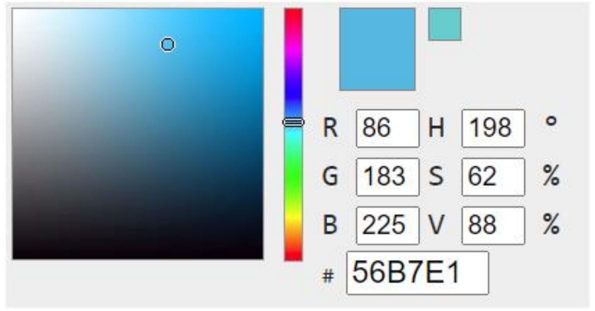


https://medium.com/machine-learning-world/feature-extraction-and-similar-image-search-with-opencv-for-newbies-3c59796bf7704

Color Features







Conigliaro, D., Ferrario, R., Hudelot, C., & Porello, D. (2017). Integrating Computer Vision Algorithms and Ontologies for Spectator Crowd Behavior Analysis. In *Group and Crowd Behavior for Computer Vision*. Elsevier. https://doi.org/10.1016/B978-0-12-809276-7.00016-3

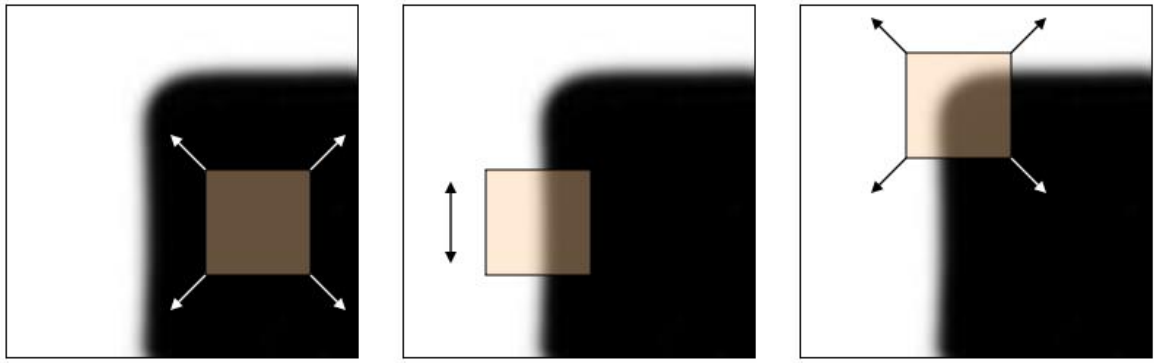
Geometric Features

1.Edge: Represents abrupt changes in intensity in an image, often seen as lines or curves. Used for object detection and shape analysis.

2.Corner: A point where intensity changes occur in multiple directions, making them good landmarks for tasks like feature matching and image stitching.

3.Blob: A region with similar intensity in an image, used for tasks like image segmentation and texture analysis.

https://cs.nyu.edu/~fergus/teaching/vision_2012/3_Corners_Blobs_Descriptors.pdf



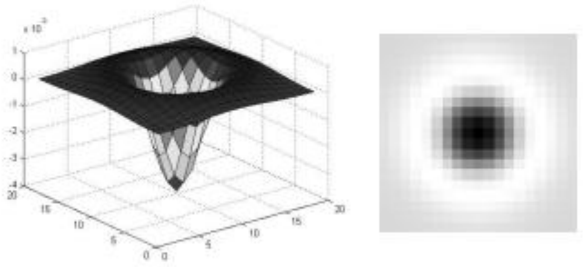
“Flat” region:
No change in all directions

“Edge” region:
No change along edge direction

“Corner” region:
Significant change in all directions

In computer vision, the detection and analysis of these features play a crucial role in **extracting meaningful information from images.** Various algorithms and techniques are employed to detect edges, corners, and blobs, and these features can be used as building blocks for more complex image processing and analysis tasks. Depending on the specific application, one or more of these features may be more suitable for capturing the desired information within an image

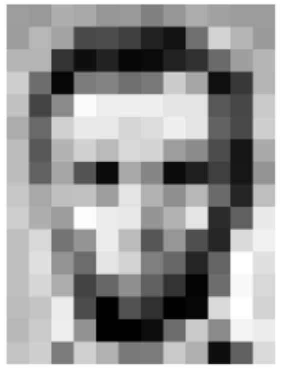
Blob

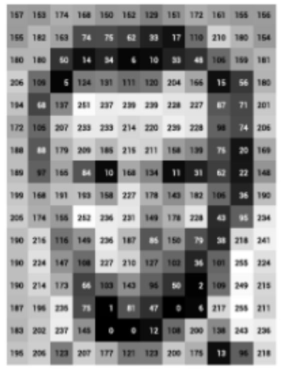


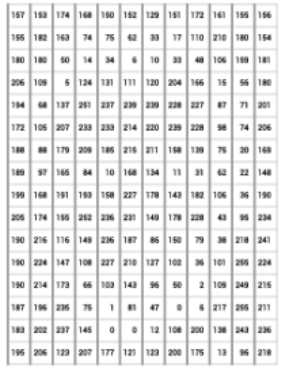
Edge

Pixel Features

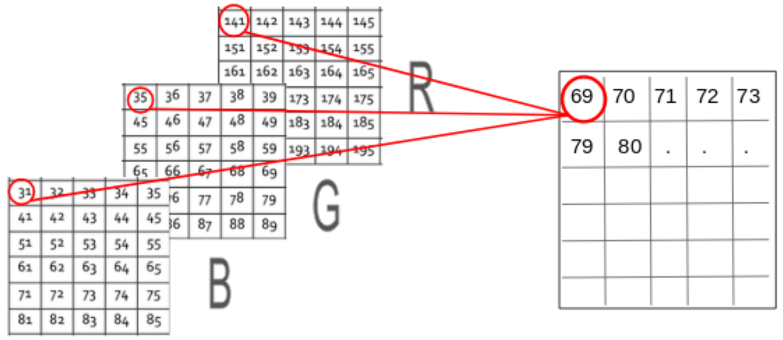
- Pixel features refer to characteristics or attributes associated with individual pixels in an image.
- Pixel features are typically represented as **numerical values**.
- Pixel features are the result **of image transformation**, this process can be achieved by algorithms.
- Each pixel in an image is associated with a set of numerical attributes or characteristics that describe various properties of that pixel, such as its color intensity, position, and so on.
- For example, in grayscale images, the pixel's intensity is represented by a single numerical value ranging from 0 (black) to 255 (white).







157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218



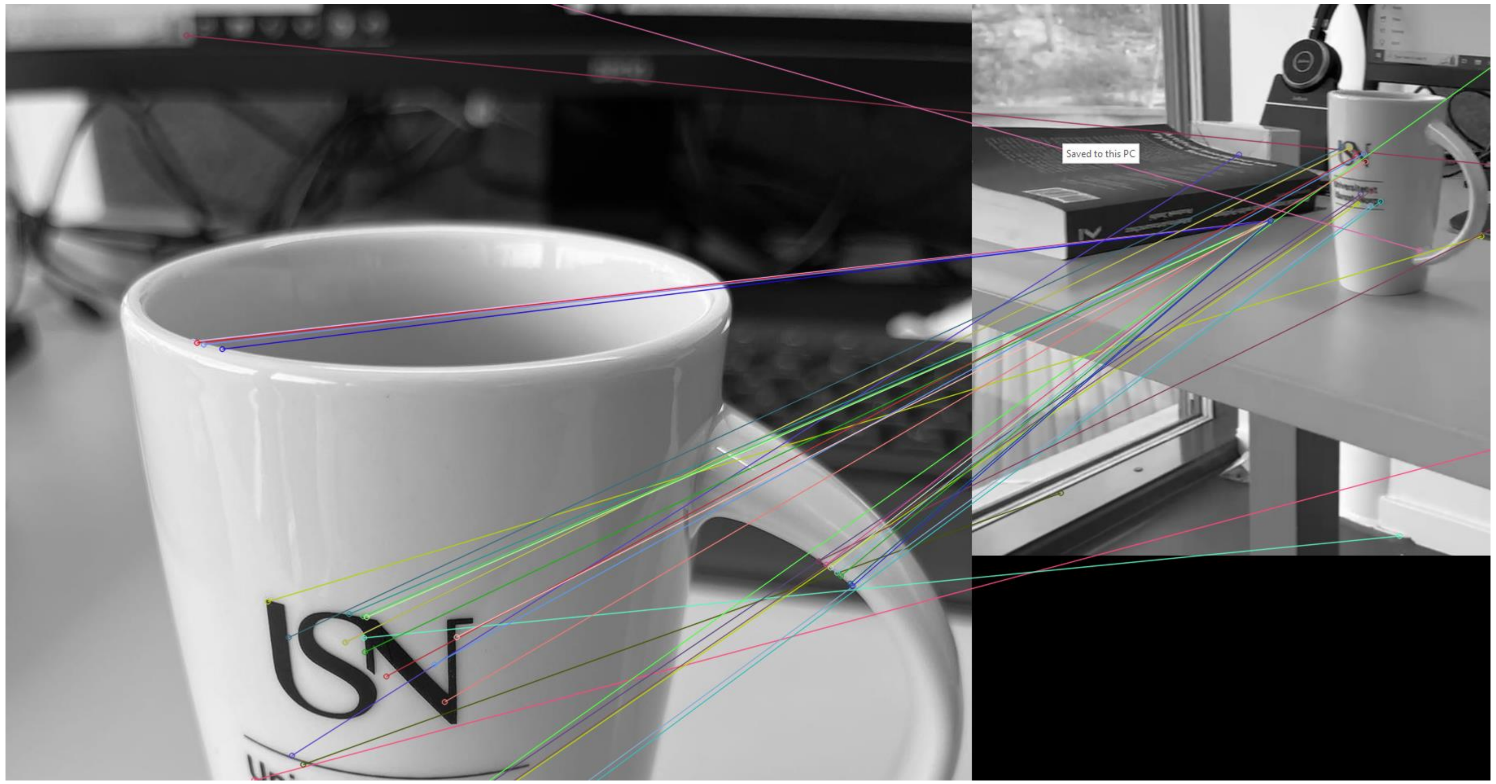
Algorithms

1. **SIFT (Scale-Invariant Feature Transform):**
    - **Description:** SIFT is an algorithm that extracts key points and their descriptors from an image. It is known for its scale and rotation invariance, making it robust to changes in an object's size and orientation.
    - **Key Points:** SIFT detects distinctive local features, such as corners and blobs, and computes descriptors that capture information about the surrounding region's texture and gradient.

2. **SURF (Speeded-Up Robust Features):**
    - **Description:** SURF is another algorithm for detecting and describing key points in images. It is designed to be faster than SIFT while maintaining good performance.
    - **Key Points:** SURF uses a combination of box filters and Haar-like wavelet responses to identify interesting points and generate descriptors.

3. **ORB (Oriented FAST and Rotated BRIEF):**
    - **Description:** ORB is a fast and efficient algorithm for feature detection and description. It is designed to be computationally lightweight and suitable for real-time applications.
    - **Key Points:** ORB combines the FAST (Features from Accelerated Segment Test) corner detector with the BRIEF (Binary Robust Independent Elementary Features) descriptor to extract and match key points.

Case: Detect an object from a video



Simple version of object detection

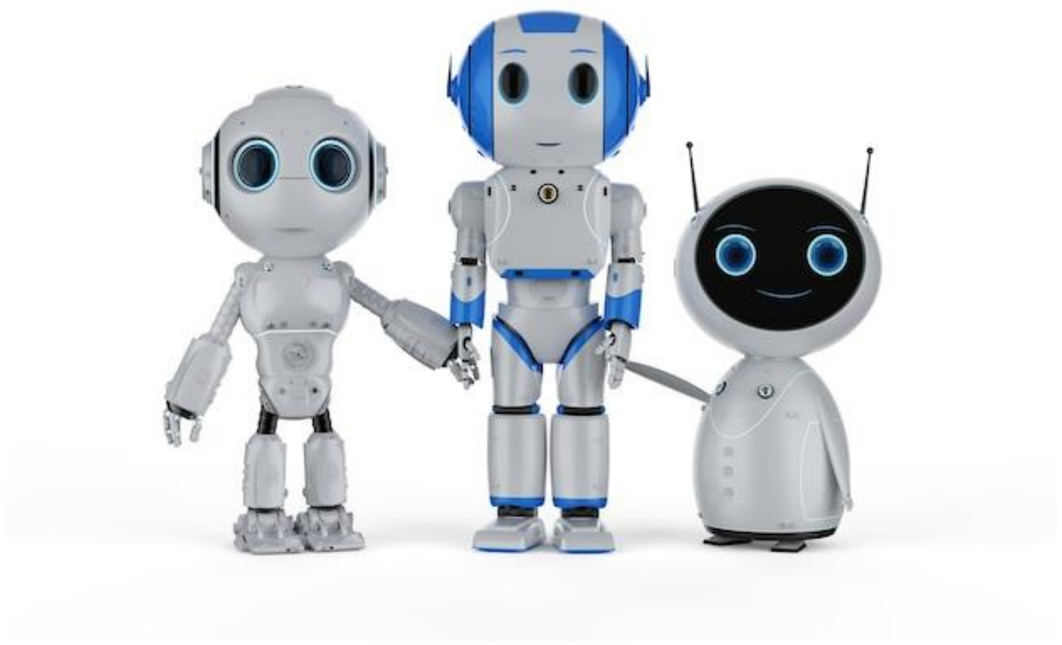
University of South-Eastern Norway
School of Business

Have a Break!

University of South-Eastern Norway

School of Business

Group Up!



Work on your assignment!

Guidance

https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e

THANKS

University of
South-Eastern Norway
School of Business