Homework Spring 2025
Applied Computer Vision (CMU-Africa)

Manyara Bonface Baraka - mbaraka

February 14, 2025

Question 1: Counting Objects - Inventory Management Systems

In this section, we will be implementing this naive method.

1. Load an image using OpenCV.
2. Convert the image to grayscale for easier processing.
3. Apply Gaussian blur to reduce noise and improve edge detection.
4. Use Canny Edge Detection to extract contours.
5. Find contours in the image and count them as individual objects.

Tasks

1. Implement the naive object counting algorithm above.
2. Experiment with different parameters of the filters you use and various image processing techniques and report how they help improve your counting algorithm.

- **For image 1:**

- Regardless of the change of parameters the count was still working correctly counting them to 7

For Image 2:

Experimentaion

- **Using Canny** For the image on 1b that had 13 objects it **counted 13 objects**
- **Use of Sobel** After tuning parameters it **counted as 14 objects**
- Use of the different Morphological operations
- Using the default parameters above it gave out 16 counts.
- Reducing the edge to 50, 100 it gave 10 objects
- Leaving at 50 and changing the Gaussian blur to 9, 9 gave the correct count.
- Gaussian blur only uses odd numbers.
- Sobel performed poorly.

Insights

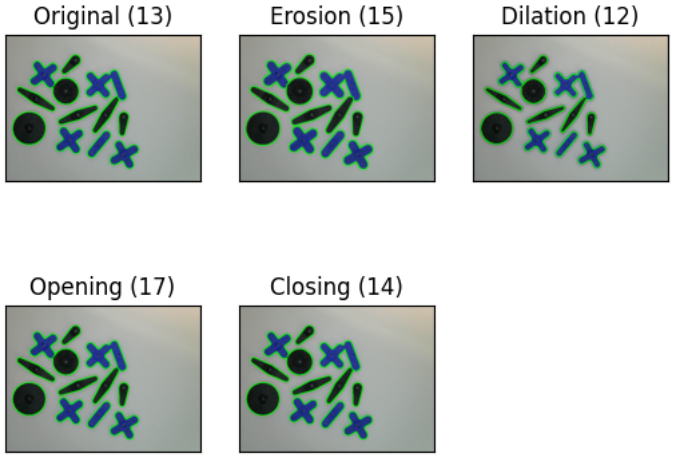
Why different counts for different parameters?

- **Gaussina blur and Edge detection**
  - **Gausian blur** This helps in noise reduction, the more blurring the less sensitive the algorithm becomes to the small variations that can lead to fewer or more edge being detected. **Canny edge detection** this relies on two threshold that enables in determining the strong and weak edges. When the threshold is too low it may lead to the detection of many weak edges that will lead to over counting, when its too high it may also miss the subtle edges hence leading to undercounting.

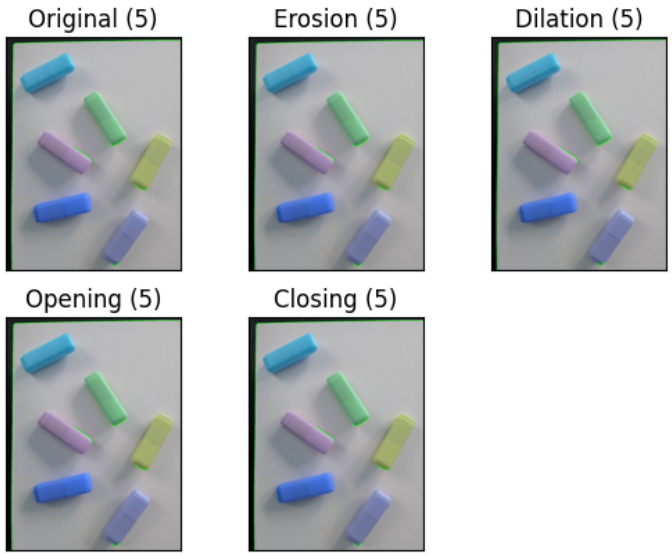
  Why odd numbers

- This is because the Gaussian kernels are symmetric and hence they must have a center pixel.
  Why Sobel performed poorly

- Sobel detects edges based on the gradient of the pixel intensity but doesn't apply any post-processing steps such as suppression. Therefore the result is a less precise edge detection mostly in images that are noisy or touching objects.



3. Real-world experimentation: Put some items on a table with a plain background, take a picture using your smartphone, and count the number of items.



4. Report on the difficulties faced by the naive counting algorithm in real-world examples and how it could be improved.

   - **Object touching each other**: When objects are touching or overlapping g the edge detection often fails to differentiate between them, the contours merge and the algorithm mistakenly counts the multiple objects as a single entity.

- **Color similarity and texture** The picture I took had almost the same color and texture that tended to blend with the background when it was converted to grey the algorithm struggled to detect the edges/

5. Suggest other real-world scenarios where counting objects could be helpful.

   - Trafficc monitoring - counting the number of vehicles on the roads
   - Biomedical imaging - counting number of cells in a microscope
   - Agriculture - counting the number of fruits or plants for yield estimation

Question 2: Finding lines on a chessboard and straightening the image - Robot-Assisted Chess Training

1. Convert the image to grayscale (of course after reading the image).
2. Apply Gaussian Blur and contrast enhancement using CLAHE.
3. Use Canny Edge Detection to identify edges in the image (experiment with Sobel).
4. Apply the Hough Transform to detect lines in the image.
5. Store and analyze angles of detected lines to determine the dominant orientations.
6. Apply image rotation to correct the perspective.

Tasks

1. Implement the straightening algorithm above.
2. Experiment with different parameters of the filters you use and various image processing techniques and report how they help improve your straightening algorithm.

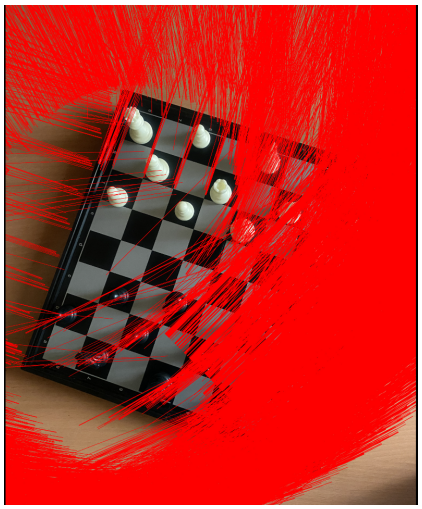


Figure 1: Threshold of 150

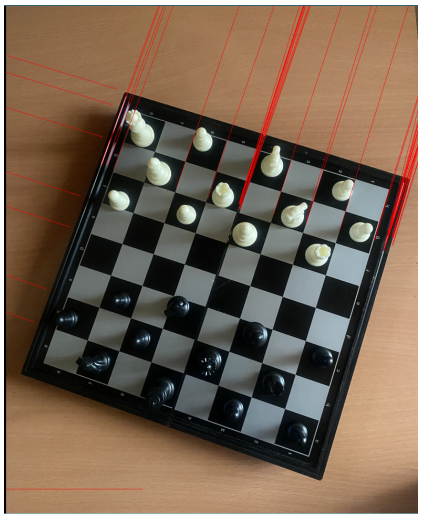


Figure 2: Threshold of 600

Threshold

- Threshold in the transformation helps in detecting lines that are considered valid, it acts as a sensitivity setting to filter out weak or less significant lines based on how many edge points align along the possible lines.
- It also defines the confidence or minimum number of votes that are needed for a line to be considered a valid detection.
  - **Low threshold [150 in my case**] more lines are detected including the weaker lines and the false positive.
  - **Higher [600 in my case**] the prominent and well-defined lines are detected however some important lines might be missed.

Canny threshold

- Mostly used to identify the regions with strong intensity gradients that often correspond to object boundaries, textures, or structural features in an image. It had both the lower threshold and the upper threshold.
- **Upper threshold:** pixels with intensity gradients above this value are considered strong edges and retained
- **Lowere threshold:** pixels that have intensity gradients below this value are discarded as noise.

Hough Transform

- Is technique used to detect shapes in images mostly lines and circles. it works even in noisy datasets.

Morphological Operations

- **Dilation**: it expands the white regions in a binary or grayscale image, it works by placing a structure element that is the kernel over the image and setting the pixel to white if at least one of its neighbors is white.

  Why Dilation
  - Enahnces thin or broken edges
  - Strengthen the line's presence
  - Connects the nearby edge fragments

  Dilation Parameters
  - **Kernel size:** this defines the shape and set size of the neighborhood over which the dilation is applied a larger kernel results in a more aggressive expansion of edges that may lead to the unwanted merging of close lines

- **Number of iterations :** More iteration expand the edges which might be important if the edges are weak, however, if the iterations are excessive it can distort the structure and merge the edges that should be separate.

- **Erosion**: This is the opposite of dilation it shrinks the white regions by removing the pixels on object boundaries it also checks whether all the pixels under the stated kernel are white if not the central pixel is removed.

  Why Erosion after dilation

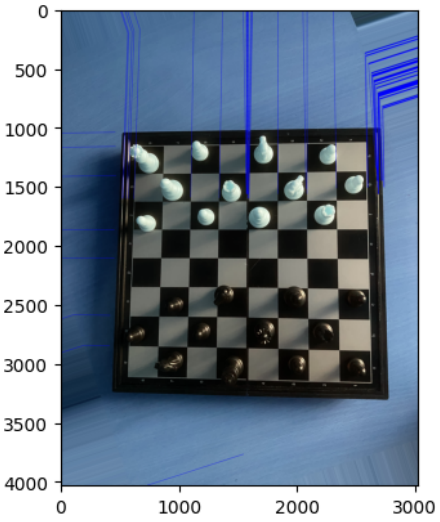
  - Removes small noise artifacts
  - Restores the original thickness of the edges
  - Keeps only the most prominent edges

  Erosion Parameter

  - just like dilatation kernel and iteration parameters perform the same only that in erosion the kernel choice affects how aggressively erosion removes the pixel

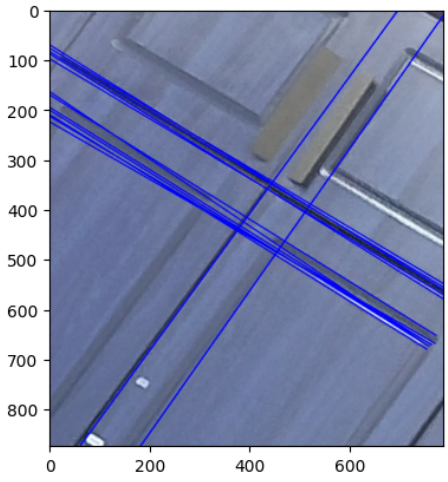
Rotation

- This step was to align the chessboard properly for it to appear upright and not tilted or skewed.

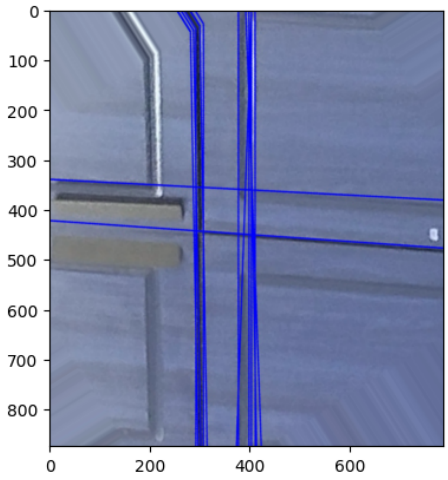


3. Real-world experimentation: Find an object with lines like a square, rotate it and use your implementation to automatically straighten it.

Line Detection



Rotation



4. Report on the difficulties faced by the naive straightening algorithm in the real-world example and how it could be improved.

Challenges

- Tried with an image a lot of noise it could not detect and gave false lines.

- In low light conditions, the naive algorithm had false line detection
- Struggles to identify curved areas

Improvement

- **Adaptive Thresholding for Better Edge Detection**
  - The naive algorithm works best with uniform lighting but it struggles with different varying illumination such as shadows or reflection. Applying adaptive thresholding will dynamically adjust the threshold values for edge detection based on the image intensity hence making it more robust.
- **Use perspective Transform instead of simple rotation**
  - In the naive algorithm rotation is corrected by estimating the dominant angle of the detected lines, as much as it worked well it is limited to only in-plane rotation therefore if the image has a perspective distortion, the simple rotation won't be fully correct.
  - Use of perspective transformation instead of rotating the image based on the detected angles, it maps four key corner points to the image of the known square region.
- **Use Deep learning**
  - The Hough transform is sensitive to noise and it requires fine-tuning of parameters such as the threshold, resolution, and the minimum length, this sometimes causes inconsistency detection in different images. Use of**Deep Hough Transform** will use CNNs to detect the lines more accurately by learning the patterns.
  - Instead of detecting lines based on the pixel intensity DHT learns the features that are presented in the real world lines additionally, it does better generalization in position and lighting.

5. Suggest other real-world scenarios where aligning an image can be helpful.
   - **Medical imaging** - this is applied when correcting the orientation of an X-ray or MRI scan
   - **Robotic visions:** enable robots to recognize structures in different environments set up
   - **Optical character recognition** - enable alignment of documents for better text recognition.

Question 3: Image Compression with Fourier Transform

Tasks

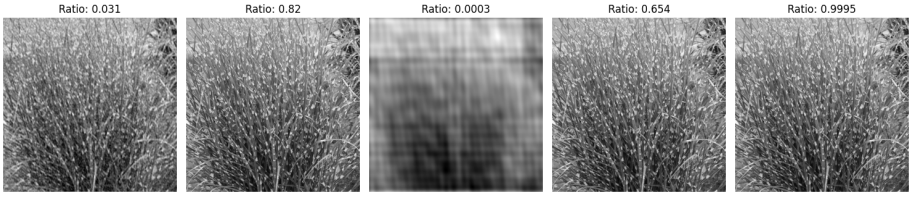
1. Convert the image to grayscale (after reading the image start with a low-frequency image).
2. Compute the Fast Fourier Transform (FFT) to analyze frequency components.
3. Apply a threshold to retain significant frequency components while discarding others.
4. Apply inverse Fourier Transform to reconstruct a compressed image.
5. Using a high-frequency image, compress it and compare the compression performance with a low-frequency image.

High frequency Compression Data

Compression Ratio	**Original Size (KB)**	**Compressed Size (KB)**	**Compression Achieved**
0.01	9866.90	1120.05	88.65%
0.82	9866.90	2932.91	70.28%
0.0003	9866.90	597.71	93.94%
0.654	9866.90	2891.28	70.70%
0.9995	9866.90	2959.05	70.01%

Compression Ratio	Original Size (KB)	Compressed Size (KB)	Compression Achieved
0.031	194.89	134.73	30.87%
0.82	194.89	204.65	-5.01%
0.0003	194.89	35.18	81.95%
0.654	194.89	204.34	-4.85%
0.9995	194.89	204.73	-5.05%

Low frequency image



6. Find a high-resolution satellite image online, compress it using your algorithm and report on its performance.

Compression Ratio	Original Size (KB)	Compressed Size (KB)	Compression Achieved
1×10^{-5}	2301.19	892.32	61.22%
1×10^{-4}	2301.19	1017.91	55.77%
1×10^{-3}	2301.19	1227.24	46.67%
1×10^{-2}	2301.19	2002.05	13.00%
1	2301.19	3395.47	-47.55%

- **How does Fourier work with compression?**
  - Fourier transforms an operation that decomposes a signal that is an image into sine and cosine waves of different frequencies. This beneficial because:

    - Images have both low-frequency (components with smooth variations like the sky) and high-frequency components(sharp edges of the image).
    - Most high-frequency components in an image contribute minimal to human perception, hence removing them can significantly reduce the data size without any loss in visual quality.
  - **The compression process:**
    - Convert image to the frequency domain using FFT np.fft.fft2(): This computes the 2D fourie transform and converts the spatial domain image into its frequency domain representation
    - Removing od the less significant frequency components
    - Perform an inverse FFT to reconstruct the compressed image.

- **Impact of the compression ratio**
  - Ration of 1: Retains all frequencies
  - Ration of 0.1: 90% of the frequency removes that is the high compression some loss of quality is seen
  - Ration 0.01: 99% of frequencies removed there is significant blurring.

Why observe some negative compression?
- It mostly happened on the low-frequency image this could be because the image has large smooth regions that are already efficiently stored hence the naive method performs worse. This indicates that the compressed image has already stored enough data.
- Too much compression results in blurring and loss of details whereas too little compression does not significantly reduce the storage size.

Best practice: images with critical details like medical images its is best to retain a high frequency of the image to avoid information loss.

Question 4: Data Augmentation

Tasks

1. Complete the function in the starter notebook which accepts an image and a type of augmentation to perform. Use the parameters described in the comment section of each augmentation.

2. Despite data augmentation being good, not all transformations are valid, and it is therefore important to consider the problem you are trying to solve. Considering the problem of facial recognition, state if the transforms given in the starter notebook will be valid or not.

Transformation Explained and relevance to face recognition

- **Resize [Nearest Neighbor and Cubic interpolation]**
  - Nearest neighbor uses the nearest neighbor for interpolation where the closes pixels values are copied to the new location

$$x' = \text{round}\left(x \times \frac{W}{w}\right), \quad y' = \text{round}\left(y \times \frac{H}{h}\right)$$

- Cubic uses cubic spline interploation that smooths out pixel values using the following cubic function:

$$f(x) = ax^3 + bx^2 + cx + d$$

Benefits of resize

- **Neearest neigbor** it fast although it produces blocky artifacts
- **Cubic** has smooth results but its expensive computationally.

Usable in Facial recognition

- **Valid Transformation** it is required to standardize the image input size
- **Issues** it might distort some key facial features that may lead to inaccurate recognition

- **Flipping [Horizonatal & Vertical]**
  - (img, 0) Flips the image vertically
  - (img, 1) Flips the image horizontally

  Flipping this image is same as applying the following transformation matrix: **Vertical Flip**

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & h \\ 0 & 0 & 1 \end{bmatrix}$$

  Horizontal Flip

$$M = \begin{bmatrix} -1 & 0 & w \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  Benefits of Flipping

  - Horizontal flip helps in recognizing faces under different orientations
  - Vertiacl flipping in face recognition can't be useful since human face is never upside down

  Usable in Facial recognition

  - **Horizontal Flop Valid**
  - **Vertical Flip Invalid**

- **Blurring [Gaussian]** - uses gaussian filter to smooth the image with the given formula that is applied for each image pixel:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

  where σ controls the blur intensity.

  Benefits of Blurring

- Enables the model to ignore minor noises and focus on essential facial features
- Reduces the overfitting by forcing the model to generalize

Usable in Facial recognition
Valid Blurring enables in handling images that have low resolution or with a noisy background.

- **Rotation** - just from the name it rotates an image in a given angle in this case 30^o around the center, Affine transformation I applied to perform rotation:.

$$M = \begin{bmatrix} \cos\theta & \sin\theta & t_x \\ -\sin\theta & \cos\theta & t_y \end{bmatrix}$$

where $\theta = 30°$, and t_x, t_y adjust for image center rotation.

Benefits of Rotation

- Help in recognizing tilted images at a given angle
- improves robustness in the real-world scenario

Usable in Facial recognition
Valid: helps in recognizing faces in different angles.

- **Shearing [X-axis Y-axis]** - Applies the shear factor on the image vecotors.

Shearing on X-Axis

$$M = \begin{bmatrix} 1 & k & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Shearing on Y-Axis

$$M = \begin{bmatrix} 1 & 0 & 0 \\ k & 1 & 0 \end{bmatrix}$$

Benefits of Flipping
It is useful for object detection but not ideal for faces

Usable in Facial recognition
Not Valid: Shearing might distort facial properties hence reducing the recognition accuracy

Summary of the transformation validation

3. Add any two transforms not included in the starter code that are relevant to a face recognition model.

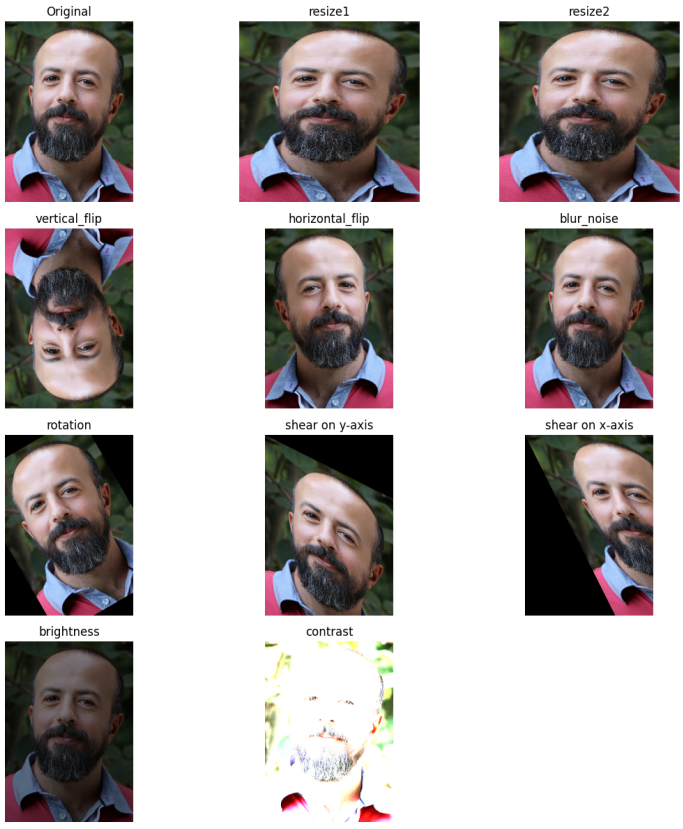
Transformation	**Validity**	**Reason**
Nearest Neighbor Resize	Yes (not always)	May distort facial features
Cubic Interpolation Resize	Yes	Maintains facial structure
Vertical Flip	No	Faces aren't naturally upside-down
Horizontal Flip	Yes	Helps recognize left/right variations
Gaussian Blur	Yes	Improves noise tolerance
Rotation (30°)	Yes	Useful for tilted faces
Shearing (X/Y Axis)	No	Distorts facial proportions

- **Brightness** - this enables to simulate different lighting condition. Each pixel is modified by:

  $$I' = \alpha I + \beta$$

  where α scales brightness and β shifts intensity

  Benfit: Helps face recognition in different light set up
  Valid Ensures the model to be robust across different light variations.
- **Contrast**: Helps the model to handle high and low contrast images.


Original
resize1
resize2
vertical_flip
horizontal_flip
blur_noise
rotation
shear on y-axis
shear on x-axis
brightness
contrast