Notes by Izam Mohammed

OpenCV

Computer vision *→ A process in which we can understand the image and videos and can be manipulated*

- *Basic*
  - *Read*
    - *Image file → imread(path, flag)*
    - *Video file*
    - *Camera image*
    - *Camera video*
  - *Show → imshow(name, image)*
  - *Save → imwrite(path, image)*
- *Noise*

  Random variations in brightness or color can be present in an image.
  - *Gaussian Noise*

    Random distribution of pixel values, followed by Gaussian normal distribution
  - *Salt-and-Pepper Noise*

    Randomly occurring bright and dark pixels
  - *Speckle Noise*

    Caused by variations in brightness or color due to random fluctuations in the image.
  - *Quantization Noise*

    The image is represented with a limited number of intensity levels
- *Geometrical shapes*
  - *Circle*
  - *Rectangle → cv2.rectangle(img, start_point, end_point, color,thickness)*
  - *line*
- *Channels*
  - *Split → cv2.split(img)*
  - *Merge*
  - *Resize → cv2.resize(img, (new_width, new_height))*
- *Color spaces*

  cv2.cvtColor(image, color code)

  - *cv2.COLOR_BGR2RGB*
  - *cv2.COLOR_BGR2HSV*
  - *cv2.COLOR_BGR2LAB → A perceptually uniform color space*
  - *cv2.COLOR_BGR2GRAY*
- *HSV color space*

  - Hue, Saturation, Value
  - Hue → type of color, range (0, 179)
  - Saturation → intensity of color, range (0 to 255)
  - Value → Brightness of color , range(0 to 255)
    Use cases
  - Color filtering → Color-based object detection
  - Image thresholding → segmenting images

- Bitwise operators
  - AND → pixel-wise and operation of 2 images, useful in masking. The result is non-zero only both values are non-zero
  - OR → useful in combining images. The result is nonzero if at least one value is non-zero
  - NOT → perform on a single image. Invert the pixel values
  - XOR → useful to highlight differences between 2 images. Result non-zero only if both values are different
- Simple thresholding
  - Advantages
  - Disadvantages
  - Thresholding is the way to convert an image to a binary image.
  - _, binary_image = cv2.threshold(img, threshold value, max, type: cv2.thresh_binary)
- Adaptive image thresholding
  - The image will be divided into various regions and the thresholding
  - Threshold = cv2.adaptiveThresholding(img, max_val, adaptive method, binarization, size of neighbour, constant subtract)
  - Thresh = cv2.adaptiveThreshoding(img, 255, cv2.adaptive_thresh_mean_c, cv2.thresh_binary, 11, 2)
- Bind trackbars
  - GUI component that allow interactively change a parameter
    code

    ```
    def update_val(x):
            Blah blah
    cv2.createTrackbar(“threshod”, “img window”, 0, 255, update_val)
    cv2.setTrackbarPos(“threshold”, “img window”, 128)
    ```
- Smoothing
  - Advantages
  - Disadvantages
  - Average blurring
    - Each pixel in image is the average of its neighbours
    - cv2.blur(original_img, (ksize, ksize))
  - Gaussian blur
    - Reduce the high frequency noises
    - cv2.GaussianBlur(img, (ksize, ksize), sigma)

    - Median Blurring
        - Replace each cell with the median value of its neighbours
        - cv2.medianBlur(original_size, ksize)
    - Bilateral filtering
        - Preserves edge while smoothing the image
    - Smoothing in coloured images → split, blur, merge
- Morphological transformations

    Operations based on the shape of the image, are only applicable in the binary image

    - Use cases
        - Noise Reduction
        - Object detection and segmentation
        - Shape and size adjustment
        - Boundary extraction
    - Erosion
        - Expands the white space
          kernel = np.ones((5, 5), np.uint8)
          erosion = cv2.erode(image, kernel, iterations=1)
    - Dilation
        - Expand the dark part in an image
          dilation = cv2.dilate(image, kernel, iterations=1)
    - Opening
        - Erosion followed by dilation
          cv2.morphologyEx(image, cv2.MORPH_OPEN, kernel)
    - Closing
        - Dilation followed by erosion
          cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)
- Image gradient
    - Rate of change of intensity in different directions
    - Types
        - Sobel operator
            - Computes the gradient using convolution with sobel kernels in both horizontal and vertical direction
            - cv2.Sobel(image, ddepth, dx, dy, ksize)
- Edge detection
    - Identifying the boundaries in an image
    - Use cases
        - Object detection
        - Image segmentation
        - Feature extraction
- Canny edge detection
    - Multistage algorithm that combines gradient calculation, non-maximum suppression, and edge tracking by hysteresis
    - cv2.Canny(img, lower_thresh, upper_thresh)

  - *Algorithm*
    - *Image Smoothing → grayscale, gaussian filter*
    - *Finding Gradients → Sobel x and y combining*
    - *Non-Max Suppression → in each kernel, if the pixel is local maximum, keep it otherwise remove it.*
    - *Double Thresholding → lower val, upper val*
    - *Edge tracking by hysteresis*
- *Mouse events*
  - *Listen to the events in the mouse*
  - *cv2.setMouseCallback("image", mouse_callback)*
def mouse_callback(event, x, y, flags, param)
- *Image pyramids*
  - *Gaussian image pyramid*
  - *Laplacian pyramid*
  - *Application*
    - *Image compression*

      Reduce the resolution
    - *Multi-scale processing*

      Perform operations on different scales
    - *Image blending*

      Combine 2 images seamlessly by blending them at different resolutions
- *Contours*
  - *It is the boundaries of an object in an image*
  - *Finding the contours*
    - *contours, _ = cv2.findContours(binary_img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)*
  - *Draw contours*
    - *cv2.drawContours(img, contours, -1, (0, 255, 0), 2)*
  - *Contour properties*

    ```
    for contour in contours:
        Area = cv2.contourArea(contour)
        Perimeter = cv2.arcLength(contour, True)
    ```
- *Template matching*
  - *Used to find a sub-image a template in an image*
  - *res = cv2.matchTemplate(img, template_img, method=cv2.CCOEFF_NORMED)*

    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)