# EN3160 – Assignment 02

Feature Detection and Image Alignment

**220701X – Savindu Wickramasinghe**
Department of Electronic and Telecommunication Engineering
University of Moratuwa

## Contents



Repository: https://github.com/Savidilsh/Fitting-and-Alignment

## 1 Blob Detection using LoG

We used the Laplacian of Gaussian (LoG) to locate round sunflower heads in the image. The photo was converted to grayscale, blurred with Gaussians at several scales, and the Laplacian operator was computed. Peaks in this scale-space were taken as blob centers.

extbfParameters: $\sigma_{min} = 3$, $\sigma_{max} = 12$, $N_\sigma = 30$, threshold $= 0.3$. Only the lower 60% of the frame was processed to ignore the sky.

The method found the main sunflower heads while ignoring small, noisy spots. The results show LoG works well for detecting roughly circular features over a range of sizes.

```python
def detect_sunflower_centers(img_path, out_path='figures/q1_sunflower.png'):
    img = cv2.imread(img_path, cv2.IMREAD_REDUCED_COLOR_4)
    if img is None:
        raise FileNotFoundError(f"Image {img_path} not found")
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    inv = cv2.bitwise_not(gray)

    blobs = blob_log(inv, min_sigma=3, max_sigma=12, num_sigma=30, threshold=0.3)

    h = gray.shape[0]
    # Filter to lower part of image (field area)
    blobs2 = [b for b in blobs if b[0] > h * 0.4]
    blobs2.sort(key=lambda b: b[2], reverse=True)
    top = blobs2[:10]
```

(a) LoG implementation code

(b) Detected sunflower blobs

Figure 1: Laplacian of Gaussian blob detection results

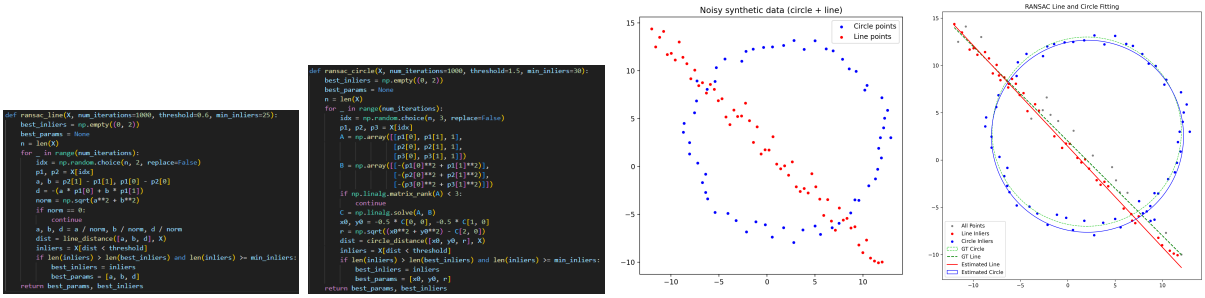
## 2 RANSAC Line and Circle Fitting

We created synthetic data that contains points from a line and a circle, both corrupted by Gaussian noise. RANSAC was used to find a robust line fit first, then the remaining points were used to fit a circle.

The final estimates were:

$$extLine: a = -0.730,\ b = -0.684,\ d = 0.992 \quad extCircle: x_0 = 2.083,\ y_0 = 2.514,\ r = 10.147$$

RANSAC removed outliers reliably and produced stable model parameters despite the noise.




(a) Line fitting code (b) Circle fitting code (c) Noisy synthetic data (d) Final RANSAC fit

Figure 2: RANSAC-based line and circle model fitting

When the circle is fitted first, its inliers can overlap with line points, leading to unstable results. Hence, estimating the line first gives better segmentation between models.

## 3 Homography Warping and Blending

A $3 \times 3$ homography $\mathbf{H}$ was found from four corner correspondences on the destination image. We used this matrix to warp a source image so it fits the target plane with correct perspective.

The homography used (source $\rightarrow$ destination) was:

$$\mathbf{H} = \begin{bmatrix} 1.311561 & -0.138957 & 453.104039 \\ 0.341954 & 0.722508 & 88.078729 \\ 0.000241 & -0.000078 & 1.000000 \end{bmatrix}$$

Examples:

- Car advertisement: the ad image is warped to match the billboard area.
- Dog on TV: the dog image is warped to fit the TV screen region.

The warped outputs look natural; small edge artifacts come from interpolation.

```python
def warp_image(dest_img_path, src_img_path, output_filename):
    global points
    points = [] # Reset points for each new run

    dest_img = cv2.imread(dest_img_path)
    src_img = cv2.imread(src_img_path)

    if dest_img is None or src_img is None:
        print(f"Error: Could not load images. Check paths: {dest_img_path}, {src_img_path}")
        return

    dest_clone = dest_img.copy()

    cv2.namedWindow("Select 4 Points")
    cv2.setMouseCallback("Select 4 Points", select_points_callback, {'image': dest_clone})
    print(f"Click 4 points on the destination image ('{dest_img_path}') clockwise.")

    while len(points) < 4:
        cv2.imshow("Select 4 Points", dest_clone)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            cv2.destroyAllWindows()
            return

    if len(points) == 4:
        h, w = src_img.shape[:2]
        src_pts = np.array([[0, 0], [w-1, 0], [w-1, h-1], [0, h-1]], dtype=np.float32)
```

(a) Homography warp code

(b) Car advertisement

(c) Dog on TV display

Figure 3: Perspective warping and blending results

—

## 4 Image Stitching using SIFT and RANSAC

We stitched two overlapping `graf` images (img1 and img5) using SIFT keypoints and RANSAC. Keypoints were matched with Lowe's ratio test (0.75) to keep good correspondences. A homography was estimated from those matches and used to warp one image into the other.

The stitch produced a single panorama with good alignment and minimal ghosting in overlapping areas.

Figure 4: SIFT keypoints, filtered matches, and final stitched panorama

The combination of SIFT and RANSAC proved robust for feature-based alignment. SIFT ensured repeatable keypoints under scale and rotation, while RANSAC eliminated false matches to produce accurate homography estimation.

—

# 5 Conclusion

This assignment shows practical methods for finding features, fitting models, and aligning images. Summary:

- LoG found circular features in the sunflower image across scales.
- RANSAC produced robust fits for a line and a circle in noisy data.
- Homography warping allowed realistic placement of images onto planar surfaces.
- SIFT plus RANSAC gave reliable matches for stitching a panorama.

The results meet the assignment goals for EN3160 Assignment 02.