# Assignment #3
# 3D Scene Reconstruction via Smartphone Camera Triangulation

## 1 Introduction

3D scene reconstruction, a cornerstone of computer vision, enables applications like augmented reality, robotics, and 3D modeling. This assignment guides you in implementing a computer vision pipeline to reconstruct a 3D scene from images captured with a smartphone camera. By applying triangulation to estimate 3D points from multiple viewpoints, you will explore multi-view geometry, feature detection, and visualization. The goal is to develop a functional system that demonstrates these principles while addressing challenges such as noisy data and varying lighting conditions.

You have maximum freedom in terms of methods to use to arrive to your goal, with only occasional suggestions provided, and a handful of restrictions. We expect you to make this work by using all online and offline resources at your disposal.

## 2 Assignment Details

### 2.1 Task

Capture 2 images of a static, textured object or scene (e.g., a detailed toy, a cluttered desk, or a small indoor setup) using a smartphone camera from varied angles. The scene must have sufficient visual features (e.g., avoid plain surfaces) for robust feature detection. Using these images, reconstruct a 3D point cloud via triangulation and visualize the results. The pipeline should handle real-world data challenges and produce a coherent 3D representation.

### 2.2 Steps

1. **Image Capture**:

   - Take 2 high-resolution images from different perspectives, ensuring at least 60% overlap between images.
   - Minimize camera shake and maintain consistent, diffuse lighting to avoid shadows or reflections.
   - Select a scene with distinct features (e.g., edges, patterns) for reliable keypoint detection.

2. **Feature Detection and Matching**:

   - Use OpenCV (Python) to detect and match keypoints across image pairs with a feature detector like SIFT (scale/rotation-invariant), ORB (fast), or SURF (balanced).
   - Apply a descriptor-based matcher (e.g., FLANN or brute-force) to detect matching keypoints.

- Visualize matched keypoints to verify correctness and address mismatches (by removing them) due to repetitive patterns or low texture.

3. **Camera Calibration**:
   - Estimate your camera's intrinsic parameters using a checkerboard pattern and OpenCV's calibration functions.
4. **Fundamental Matrix Estimation**:
   - Compute the fundamental matrix between image pairs via the 8-point algorithm using the previously detected matched keypoints and optionally RANSAC to handle outliers.
   - Validate the matrix by visualizing epipolar lines and checking consistency.
5. **Triangulation**:
   - Use the fundamental matrix to match arbitrary points between the two images.
   - And then triangulate matched points to reconstruct their 3D versions using linear or iterative methods of your choice.
6. **Visualization**:
   - Plot the calculated 3D points using Matplotlib for scatter plots.
   - Enhance visualizations by coloring points based on their image intensity.

### 2.3 Deliverables

1. **Report (2–3 pages)**:
   - Describe the methodology (image capture, feature detection, triangulation).
   - Discuss challenges (e.g., poor lighting, insufficient keypoints) and solutions.
   - Present results, including sample images, matched keypoints, 3D point cloud visualizations.
   - Compare two feature detectors (e.g., SIFT vs. ORB) for keypoint count, matching accuracy, and reconstruction quality.
2. **Code Submission**:
   - Submit well-documented Python code with clear comments for each pipeline step.
   - Include a README with dependencies and instructions for running the code.
3. **Discussion**:
   - Analyze reconstruction accuracy, identifying error sources and propose improvements.
   - Discuss trade-offs between computational complexity and reconstruction quality.

### 2.4 Evaluation Criteria

- **Correct Implementation (40%)**:
  - Accurate feature matching and fundamental matrix estimation (20%).
  - Correct triangulation and pose estimation, yielding a coherent 3D point cloud (20%).
- **Quality of Reconstruction and Visualization (30%)**:
  - Clear, outlier-free 3D point cloud (15%).
  - Informative visualizations, including camera poses or colored points (15%).
- **Report Clarity and Depth (20%)**:
  - Well-structured report with clear methodology and error analysis (10%).
  - Insightful discussion of challenges, solutions, and improvements (10%).
- **Code Organization and Documentation (10%)**:
  - Clean, modular code with comprehensive comments and a clear README (10%).