COMS 4733 Homework 3

Jaisel Singh

November 13, 2024

1 Camera Calibration

1.1 Constructing the Intrinsic Matrix

The intrinsic camera matrix K encodes the internal parameters of a camera that define how 3D points in the camera coordinate frame are projected onto the 2D image plane. I implemented the `get_camera_intrinsics` function to construct this 3×3 matrix from the focal lengths (f_x, f_y) and principal point coordinates (c_x, c_y).

The intrinsic matrix has the following structure:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where f_x and f_y represent the focal lengths in pixel units (accounting for sensor size and lens properties), and (c_x, c_y) represents the principal point—the intersection of the optical axis with the image plane. This matrix transforms 3D points in the camera frame to homogeneous 2D pixel coordinates through the projection equation $\lambda \mathbf{p} = K\mathbf{P}_c$, where $\mathbf{P}_c = [X, Y, Z]^T$ is a 3D point and $\lambda = Z$ is the depth.

Image Downsampling Question: When downsampling an image by half (from $H \times W$ to $H/2 \times W/2$ using `image[::2, ::2, :]`), the camera parameters must be adjusted to maintain equivalent imaging geometry. The new parameters should be:

$$f_x^{new} = f_x/2 \tag{2}$$

$$f_y^{new} = f_y/2 \tag{3}$$

$$c_x^{new} = c_x/2 \tag{4}$$

$$c_y^{new} = c_y/2 \tag{5}$$

All intrinsic parameters scale by the same factor as the image resolution because downsampling changes the effective pixel size. When we skip every other pixel, each new pixel represents twice the physical area, so focal lengths (measured in pixels) are halved. Similarly, the principal point's pixel coordinates are halved since the origin shifts by the same downsampling factor.

1.2 Charuco Board Detection

Camera calibration requires determining the extrinsic parameters—the rotation R and translation t that relate the camera frame to a world coordinate frame. I used OpenCV's Charuco board detection to estimate these parameters for both the front and wrist cameras.

Implementation:

1. Detected Charuco markers in each calibration image using `cv2.aruco.CharucoDetector`
2. Refined corner positions for sub-pixel accuracy
3. Computed pose using `cv2.aruco.estimatePoseCharucoBoard`, which returns rotation vector (rvec) and translation vector (tvec)
4. Converted rvec to rotation matrix using Rodrigues transformation
5. Constructed the 4×4 extrinsic transformation matrix:

$$T_{world}^{cam} = \begin{bmatrix} R & t \\ \mathbf{0}^T & 1 \end{bmatrix}$$

Results: The algorithm successfully detected markers in both camera views with sufficient coverage for reliable pose estimation. Visualization of the detected board shows coordinate axes properly aligned with the Charuco pattern, confirming accurate calibration. The computed extrinsics were saved to `data/calibration/extrinsics_result.json` for use in subsequent point cloud processing.

World Frame Orientation Question: Examining the calibration visualizations and extrinsic transformation matrices, the z-axis is parallel to the vertical direction (orthogonal to the table surface). Looking at the translation vector components in the extrinsics, the camera center's position along the z-axis is **positive**, indicating the camera is positioned above the table/calibration board. This makes physical sense as cameras must be elevated to capture a top-down or angled view of the workspace. The red, green, and blue axes in the pose visualization confirm this coordinate system orientation. This is based on Looking at the tvec values (which represent world origin in camera frame), if 'tvec[2]' (z-component) is positive, it means:

- The world origin is in front of the camera (positive z in camera frame)
- This implies the camera is positioned at negative z in the world frame

2 Point Cloud Visualization

2.1 Inverse Projection Function

The `depth_to_camera_frame_point_cloud` function reconstructs 3D geometry from 2D depth measurements by reversing the camera projection model.

Method: Given a depth image where each pixel (u, v) has an associated depth value d, the corresponding 3D point in the camera frame is computed as:

$$x = (u - c_x) \cdot d / f_x \tag{6}$$
$$y = (v - c_y) \cdot d / f_y \tag{7}$$
$$z = d \tag{8}$$

This inverse perspective projection uses the depth value directly as the z-coordinate (distance along the optical axis), while x and y are computed by "unprojecting" the pixel coordinates through the camera intrinsics. The terms $(u - c_x)$ and $(v - c_y)$ center the pixel coordinates relative to the principal point, and division by the focal lengths converts from pixels to metric units in the camera frame. The function operates vectorially over all pixels using meshgrid operations, efficiently generating the complete point cloud in a single pass.

2.2 Camera to World Transformation Function

The `transform_camera_to_world` function performs a critical coordinate transformation, mapping points from the camera's local reference frame into a shared world coordinate system.

Approach: The extrinsic matrix $T_{world}^{cam} = [R \mid t]$ represents the transformation from world to camera coordinates. To transform in the opposite direction (camera to world), we compute its inverse:

$$T_{cam}^{world} = (T_{world}^{cam})^{-1} = \begin{bmatrix} R^T & -R^T t \\ \mathbf{0}^T & 1 \end{bmatrix}$$

where we exploit the property that $R^{-1} = R^T$ for rotation matrices (orthogonality). Points are converted to homogeneous coordinates by appending a 1, transformed using T_{cam}^{world}, then converted back to Cartesian coordinates. This unified framework handles both rotation and translation in a single matrix multiplication.

2.3 Robot Motions and Wrist Camera Extrinsics

As the robot moves, the wrist camera (rigidly mounted to the end-effector) moves with it. The `calculate_wristcam_extrinsics` function computes updated camera extrinsics for any robot configuration.

Key Insight: The relative transformation between the end-effector and wrist camera is fixed—it never changes regardless of robot motion. This can be expressed as:

$$T_{world}^{wrist} = T_{world}^{base} \cdot T_{base}^{eef}(q) \cdot T_{eef}^{wrist}$$

where T_{eef}^{wrist} is constant and $T_{base}^{eef}(q)$ depends on robot joint configuration q.

Implementation:

1. From the calibration pose, compute the fixed relationship:

$$T_{eef}^{wrist} = (T_{base}^{eef})^{-1} \cdot (T_{world}^{base})^{-1} \cdot T_{world}^{wrist}$$

2. For any new robot configuration, apply this constant transform:

$$T_{world}^{wrist}(new) = T_{world}^{base} \cdot T_{base}^{eef}(new) \cdot T_{eef}^{wrist}$$

This allows us to track the wrist camera's pose throughout the robot's workspace without re-calibration.

2.4 Visualize

The final step merges point clouds from multiple cameras into a unified 3D reconstruction of the workspace.

Pipeline:

1. Loaded RGB-D data from front camera (fixed viewpoint) and wrist camera (robot-mounted)
2. Applied inverse projection to convert depth images to 3D points in respective camera frames
3. Transformed both point clouds to world frame using their respective extrinsics
4. Applied workspace bounding box filter to remove outliers and focus on region of interest

5. Combined filtered point clouds and their RGB colors for visualization

Observations: The merged point cloud demonstrates successful multi-view fusion. The front camera provides stable, wide-field coverage of the tabletop workspace, while the wrist camera contributes detailed close-up geometry that varies with robot motion. Together, they provide complementary perspectives that reduce occlusions and improve scene completeness. The bounding box effectively isolates the relevant workspace from background clutter.

Visual Verification of Wrist Camera Extrinsics: The correctness of the computed wrist camera extrinsics can be verified through several visual cues in the Open3D point cloud visualization:

1. **Geometric alignment:** Objects visible from both cameras (e.g., the green mug, table surface) appear as single coherent 3D structures rather than duplicated or misaligned ghosts. If extrinsics were incorrect, we would see doubled objects offset from each other.

2. **Surface continuity:** The table plane and object surfaces show smooth, continuous geometry across the transition between front-camera and wrist-camera regions, indicating proper spatial registration.

3. **No obvious distortions:** Objects maintain correct proportions and shapes without warping, stretching, or unnatural deformations that would result from misaligned reference frames.

4. **Consistent scale:** Objects appear at the correct relative sizes between viewpoints, confirming that translation components of the extrinsics are accurate.

3 Semantic Segmentation

3.1 Visualize the Segmentation of Pens

Semantic segmentation assigns object-level labels to points in the 3D scene. This section extends 2D segmentation masks into 3D by associating them with the corresponding point cloud geometry.

Implementation Pipeline:

1. **Point cloud generation:** Loaded front camera RGB-D image, applied inverse projection to obtain 3D points in camera frame, then transformed to world frame

2. **Workspace filtering:** Applied bounding box constraints to isolate the tabletop region and remove background points

3. **Mask-point correspondence:** Loaded 4 binary segmentation masks (one per pen), flattened them to 1D arrays matching the point cloud indexing, and applied identical bounding box filtering to maintain alignment

4. **Color assignment:** Replaced original RGB values with distinct colors for visualization (red, green, blue, yellow for pens 0-3 respectively)

5. **Centroid calculation:** For each pen, computed the 3D centroid as:

$$\mathbf{c}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbf{p}_j$$

   where N_i is the number of points belonging to pen i and $\mathbf{p}_j$ are the 3D point coordinates

Technical Challenge: The primary difficulty was maintaining precise correspondence between 2D segmentation masks and 3D points through multiple filtering operations. Since the bounding box removes certain points, the mask indices must be filtered identically to preserve alignment. This was achieved by applying the same boolean indexing operation to both the point array and flattened mask arrays.

3.2 Geometric Distance Calculation

Computing pairwise distances between object centroids provides quantitative spatial relationships useful for manipulation planning and workspace analysis.

Distance Calculation: For centroids $\mathbf{c}_i$ and $\mathbf{c}_j$, the Euclidean distance is:

$$d_{ij} = \|\mathbf{c}_i - \mathbf{c}_j\| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$$

Pairwise Distances:

- Pen 0 to Pen 1: 0.3540 m (35.40 cm)
- Pen 0 to Pen 2: 0.2161 m (21.61 cm)
- Pen 0 to Pen 3: 0.2022 m (20.22 cm)
- Pen 1 to Pen 2: 0.1953 m (19.53 cm)
- Pen 1 to Pen 3: 0.1527 m (15.27 cm)
- Pen 2 to Pen 3: 0.1025 m (10.25 cm)

Analysis:

- **Closest pair:** Pen 2 and Pen 3 at 0.1025 m (10.25 cm)
- **Farthest pair:** Pen 0 and Pen 1 at 0.3540 m (35.40 cm)

Interpretation: These geometric relationships provide actionable information for robotic manipulation. The closest pair (pens 2 and 3) are separated by just over 10 cm, indicating objects that may require careful grasp planning to avoid collisions during pick operations. The farthest pair (pens 0 and 1) defines the maximum workspace extent at approximately 35 cm. For multi-object manipulation tasks, these distances inform collision avoidance strategies, optimal pick sequences to minimize end-effector travel distance, and potential simultaneous grasping opportunities for nearby objects.

4 Semantic Features

Self-supervised vision models like DINOv2 learn rich semantic representations without explicit labels. This section explores these learned features by computing cosine similarity between a query point's feature vector and all other pixels in the image.

Method: DINOv2 extracts a 384-dimensional feature vector for each patch in the image. For a query point at pixel (x, y), cosine similarity with all other pixels is computed as:

$$\text{sim}(q, p_i) = \frac{f_q \cdot f_{p_i}}{\|f_q\| \|f_{p_i}\|} = \frac{f_q}{\|f_q\|} \cdot \frac{f_{p_i}}{\|f_{p_i}\|}$$

where f_q is the query feature and f_{p_i} are features of all other pixels. The resulting similarity map is visualized as a heatmap where warm colors (red/yellow) indicate high semantic similarity and cool colors (blue) indicate low similarity.

4.1 Query Point Analysis

For this analysis, I selected three query points on different objects and regions to explore what semantic features DINOv2 has learned.

Query Point 1 (470, 268) - Red Pen: This point was selected on the red pen in the lower portion of the scene. The heatmap reveals strong activation (red/yellow/green regions) primarily on the pens in the scene, with the highest similarity on the queried red pen itself and moderate-to-high similarity on the other pens regardless of their color. Notably, other cylindrical objects like the green mug show moderate similarity, suggesting the model recognizes geometric shape patterns. The background table surface, walls, and robot arm display minimal activation (blue regions), demonstrating effective foreground-background segmentation.

Semantic Interpretation: DINOv2 has learned to group objects by both shape and category. The model identifies pen-like cylindrical objects and associates them together despite color differences, indicating it captures geometric and functional properties rather than purely appearance-based features. The cross-object similarity between different colored pens demonstrates the model's ability to recognize object categories invariant to color, which is essential for robust object recognition in manipulation tasks.

Query Point 2 (350, 400) - Green Mug: This query point was placed on the green mug. The heatmap shows the strongest activation concentrated on the mug itself, with significant similarity extending to other cylindrical and rounded objects including the pens. Interestingly, the pink cup in the background also shows moderate similarity, suggesting DINOv2 groups objects by their container-like properties and circular geometry. The flat background surfaces remain largely dissimilar (blue).

Comparison to Query Point 1: While Query Point 1 (pen) showed similarity across all pens with some activation on the mug, Query Point 2 (mug) shows its strongest response on the mug itself with moderate similarity to pens. This asymmetry reveals that DINOv2's feature space captures hierarchical relationships - the mug is recognized as cylindrical like pens, but has additional semantic properties (container, larger scale) that distinguish it as its own object category. The model appears to encode both fine-grained geometric features and higher-level semantic categories.

Query Point 3 (200, 150) - Background Wall: This point was selected on the white background wall in the upper-left region. The heatmap displays dramatically different behavior: strong activation (red/yellow) appears across the entire white wall region and extends to other flat, uniform background surfaces. All foreground objects (pens, mugs, containers, robot arm) show very low similarity (blue/dark blue), creating a stark foreground-background dichotomy.

Interpretation: The background query demonstrates that DINOv2 has learned scene structure and context segmentation. The model clearly distinguishes between manipulable foreground objects and environmental background surfaces. This is particularly valuable for robotic applications where identifying the "operating surface" versus objects-of-interest is fundamental. The strong similarity across all background regions regardless of lighting variations indicates the model has learned texture and semantic context rather than just pixel intensity, showing robustness to illumination changes.

4.2 Comprehensive Analysis

Analyzing the three query points together reveals key properties of DINOv2's learned representations:

Semantic Granularity: DINOv2 captures hierarchical semantic information across multiple levels. At the finest level, it distinguishes individual object instances (highest similarity on the queried object itself). At the category level, it groups functionally similar objects (all pens show high mutual similarity). At the coarsest level, it separates major scene components (foreground objects versus background surfaces). This multi-scale representation enables both precise object localization and broader scene understanding, critical for manipulation planning where the robot must identify specific grasp targets while understanding workspace constraints.

Shape vs. Appearance: The model demonstrates strong shape-based semantic understanding that transcends superficial appearance. The red, green, blue, and yellow pens all exhibit high feature similarity despite dramatically different colors, while the white background wall (which shares color with some pens) shows near-zero similarity to foreground objects. Similarly, cylindrical objects (pens, mug) share geometric features in the embedding space. This indicates DINOv2 learns 3D shape priors and geometric structure rather than 2D color patterns, a property essential for robust perception under varying lighting and appearance conditions in real-world robotics.

Foreground-Background Separation: The stark contrast between Query Point 3 (background) and Query Points 1-2 (foreground objects) reveals sophisticated scene understanding. Background regions show strong mutual similarity and near-zero similarity to any foreground object, regardless of geometric or color properties. This demonstrates that DINOv2 has learned contextual and affordance-relevant features - it implicitly understands that tables, walls, and support surfaces constitute the "environment" while pens, mugs, and containers are "objects." This distinction aligns with manipulation-relevant affordances where some regions are graspable targets and others are constraints.

Cross-Category Relationships: Examining similarity patterns across object types reveals nuanced categorical relationships. Pens show moderate similarity to the mug (both cylindrical), but the mug shows stronger self-similarity, suggesting it occupies a distinct region in feature space. The pink cup shows some similarity to the green mug but not to pens, indicating that "container" properties are encoded separately from pure geometry. These graded similarities suggest DINOv2's feature space is not organized into discrete categories but rather represents continuous semantic relationships based on shared visual and functional properties.

Implications for Robotics: These semantic features provide several advantages over raw pixels for robotic perception:

1. **Generalization:** Object recognition across appearance variations (lighting, color, texture). The model's color-invariant pen recognition suggests it would identify objects under different illumination or with surface variations, reducing the need for extensive data collection across environmental conditions.

2. **Few-shot learning:** New object categories can be learned from minimal examples by leveraging learned geometric and semantic feature representations. A robot could learn to identify new pen types from a single demonstration by recognizing their similarity in DINOv2 feature space to known pens.

3. **Robustness:** Invariance to viewpoint, lighting conditions, and partial occlusion. The strong geometric priors mean objects remain recognizable even when partially occluded or viewed from novel angles, crucial for manipulation where objects are frequently occluded by the gripper or other objects.

4. **Semantic grouping:** Functionally similar objects share similar representations, enabling transfer of manipulation strategies. A grasping policy learned on one cylindrical object (like the green mug) could potentially transfer to other cylindrical objects (pens) due to their shared geometric features.

5. **Scene understanding:** Automatic foreground-background segmentation provides task-relevant scene structure without explicit supervision, enabling the robot to focus computational resources on manipulable objects while using background features for localization and collision avoidance.

The heatmaps demonstrate that DINOv2 learns meaningful, hierarchical semantic structures without supervision, making it a powerful foundation for robotic vision systems that must operate in diverse, unconstrained environments. The model's ability to balance geometric invariance, categorical grouping, and instance discrimination addresses fundamental challenges in robot perception for contact-rich manipulation tasks where both precise localization and semantic understanding are essential. I have generated a pdf that is attached below, it matches the implementation of the Jupyter notebook for all visuals that might be used for reference in the above report. During the pdf generalization, I commented out the sections which rendered video.