Project 2 guidelines - Augmented reality (AR)

In this project you are going to build an AR application.

This project will conclude the second part of the course, tackling the topics of:

- Transformation
- Camera calibration
- Feature detection

Part 1: perspective warping (35 points)

1. Choose a good "feature-full" 2d image that you can track- this is your reference image to find the features and track them.
2. Print out the image and make a video of it on a planar surface while moving around the image- the movie must have big changes in rotation, translation and scale relatively to the image. The movie must be at least 20 seconds.
3. The output of this part is a 2d warped template that rests on the original 2d image. what does it mean? that when you move the camera around the printed image, you will see the warped template moving with it as if it is stuck to the printed image- replacing it.

Part 2: planar AR - demo cube (15 points)

1. Calibrate your camera from before with a printed chessboard (follow our calibration notebook). you will need `K` and `dist_coeffs` for this part.
2. NOTE: in this part you'll need to use `cv2.solvePnP`. **Add a section in the final PDF that talks about PnP and its different variants, and include some of the basic math regarding the basic PnP approach.** This section should be at least a page (including equations) of your final PDF.
3. Render a 3D cube on top of the planar image from part 1. The cube should appear to be "standing" on the planar image as you move the camera around it.

Part 3: planar AR - full 3d model (10 points)

Render more elaborate 3D objects (textured .obj/.ply files) (try to use `open3d`/`trimesh`/`torch3d`). Here you will tackle again the projection from (virtual) camera to image using extrinsics and intrinsics data.

Part 4: Occlusion Handling (20 points)

Implement realistic occlusion where real-world objects can hide virtual AR content- a hand passing in front of a virtual object should occlude it correctly. **Requirements:** - Detect foreground objects using classical CV techniques (background subtraction, color segmentation, or morphological operations) - Generate an occlusion mask and render AR content with proper depth ordering - Video must show hand passing in front of virtual objects at least 3 times - Note: Do NOT use deep learning methods for foreground detection. - Choose the tracked image wisely- good colors that contrast with skin tones will help.

Part 5: Multi-Plane Tracking with Visualization (20 points)

Track three planar surfaces at different angles simultaneously and visualize their spatial relationship.

Variant Selection

- Sum all digits from both group member IDs
- Take the last digit of this sum
- **Even digit** → Portal Visualization
- **Odd digit** → Particle Flow Visualization

- **Include this calculation at the start of Part 5 documentation.** 10-point deduction for incorrect calculation.

Multi-Plane Tracking (All Variants)

- Track 3 different planar markers at different orientations (not parallel)
- All planes must be simultaneously visible for at least 15 seconds in a 30+ second video
- Camera must show multiple viewpoints
- At least one plane should exit and re-enter the frame

Variant A: Portal Visualization (Even)

- Place a circular or rectangular portal at the center of each tracked plane, covering ~50% of the plane area
- Each portal shows a view into a different 360° environment
- Portal view must change based on camera position and angle (parallax effect - different camera poses show different parts of the 360° world)
- Add visible borders/frames around portals

Variant B: Particle Flow Visualization (Odd)

- Connect the 3 plane centers with Bézier curves forming a triangle
- 100-300 particles continuously flow along curves between planes
- Each particle has a color based on its origin plane (use 3 distinct colors)
- Particles leave motion trails
- Draw glowing spheres at plane centers that pulse when particles spawn/arrive
- Particles should travel in all 6 directions between the 3 planes

General Requirements (Both Variants)

- Use calibrated camera intrinsics from Part 2
- Visualization must maintain correct 3D spatial relationships as camera moves
- Handle plane tracking loss gracefully
- You don’t need to implement occlusion handling for this part.

Documentation

- Show ID calculation and variant selection
- Explain multi-plane tracking approach
- Describe visualization implementation
- Where does your system work well and where does it fail?
- Analysis of tracking success rate and FPS performance

Submission guidelines

1. Groups of up to 2 people.
2. Please add to the PDF some explanations. Maybe some debug outputs you have (images or data that is relevant).
3. Results expected in a .zip file with the name `PROJ2_NAME1_ID1_NAME2_ID2.zip` with content of:
    - A detailed summary of the work done and assumptions made. Where does your algorithm succeed and where it failed?
    - Code in .py files
    - The output videos in a reasonable format.
    - **Remember to add the final result video as a file in the .zip directory or as a link to youtube inside the PDF** (in this case please put it at the start and at the end of the report).
4. Submission is due 3 weeks from the last class.

Good luck!

Yoni Chechik