Proposal for final project

Rules: Must involve images and programming. Can involve hardware, human, animal, nature, and other things in between.

You can take a look at these prior student projects for inspiration.

Guidelines

- Feel free to use images, diagrams to convey your idea, not just text.
    - For diagrams, recommended too: https://www.drawio.com/
- Your final project submission will be graded based on the following criteria: **Creativity**, **Technical work**, **Social Impact**, and **Public Engagement** .
- Provide explanations to your choice for data, model, approaches as needed.
- Find a fun name for your project that is engaging.
- Use bullet points, but bring your personal connection to the project. We want your excitement (and skepticism) about computer vision trickle into the narrative.

- **Project Title (1pt): (Replace the yellow text with your project title)**

NETHERGAZE

- **Goal (6 pts):**
    - What am I going to do?
    - Why is this going to be useful?

My goal for this project is to develop an augmented reality system that empowers users to place virtual objects within a real-time video stream with exceptionally stable tracking . This means that once a virtual object is placed, it should appear firmly "anchored" to its real-world position, maintaining its relative scale and orientation even as the camera moves or the scene changes. This will be very useful when it comes to designing a room, like seeing if something from a website will fit in your space. It could even apply to gamers who could place virtual characters in their room to interact with.

- **Problem Statement (6 pts)**

1. Real-time Performance: Processing video frames to detect and track features while simultaneously rendering virtual objects demands high computational efficiency, especially on consumer-grade hardware.
2. Environmental Variance: Changes in lighting, shadows, textures, and the presence of dynamic objects (e.g., people moving) can easily confuse tracking algorithms, leading to "jitter" or "drift" of virtual objects.
3. Occlusion: When real-world objects temporarily block the view of the tracked surface or the virtual object itself, maintaining pose estimation without losing track is crucial
4. Accuracy and Precision: Even slight inaccuracies in pose estimation accumulate over time, leading to virtual objects "sliding" or detaching from their intended real-world anchors.
5. Computational Complexity: Robust tracking often involves complex algorithms like SLAM (Simultaneous Localization and Mapping) or advanced feature descriptors, which are resource-intensive.

This problem is important to me because I am fascinated by the potential of augmented reality to bridge the digital and physical worlds. However, I've often been frustrated by the instability and lack of realism in many current AR experiences. I believe that reliable and stable tracking is the cornerstone of truly immersive AR, and by tackling this challenge, I aim to contribute to a future where virtual content feels genuinely integrated into our perception of reality.

- **Approach (5 pts)**
    - What step by step approach am I going to try?
    - What tools or prior knowledge I am going to use/need? **References Recommended**

My approach is will involve a multi-step pipeline designed to achieve robust, real-time tracking:

1. Video Capture and Preprocessing

Implement camera input integration with OpenCV for real-time video frame capture. Preprocess each frame: resize to optimize performance, normalize lighting if needed, and convert color space to grayscale for feature extraction. Ensure robust real-time frame handling (minimum latency, dropped frame prevention).

2. Initial Pose Detection & Estimation

Integrate ArUco marker detection to provide fast, reliable pose initialization. Develop fallback or alternative feature-based detection (e.g., ORB) to allow anchoring on user-selected surfaces,

enabling generalization beyond markers. Establish initial pose (‘anchor’) data structure for all virtual objects.

3. Real-Time Feature Tracking

Implement feature tracking using algorithms such as KLT or robust visual odometry approaches. Update camera pose in real time based on environmental movement and feature drift. Maintain a list of currently tracked points for pose continuity.

4. Virtual Object Rendering

Integrate a 3D rendering library (e.g., PyOpenGL or a similar light engine). Develop code for drawing basic 3D primitives and models at anchored real-world poses. Ensure rendering integrates the estimated camera pose for proper overlay and scale.

5. User Interaction Interface

Design and implement an interface to allow users to select and place virtual objects onto detected surfaces in real-time. Enable interactive controls for users to rotate and scale placed objects using simple input gestures or UI controls.

6. MVP Milestone

Verify all core components (steps 1–5) work cohesively, achieving stable, interactive virtual object placement and basic tracking. Conduct real-world testing on multiple surfaces with different lighting and movement.

7. Pose Refinement and Optimization (Post-MVP)

Integrate pose refinement with a Kalman Filter or Extended Kalman Filter for smoothing and drift correction. Optimize tracking pipeline for low computational load, implementing batch processing of features if required. Study visual odometry or simple SLAM integration for robustness against occlusion and significant camera movement. Benchmark stability (drift, jitter, object anchoring) and iterate for maximum real-world reliability.

8. Final Demo Preparation

Prepare a showcase demo with multiple scenarios and record a short video to document the stability and feature set of the system. Develop documentation and annotated diagrams for final presentation, using visual assets created in earlier steps.

Tools and Prior Knowledge:

Libraries: OpenCV for video processing, feature detection, and tracking. NumPy for numerical operations. Potentially specific libraries for 3D rendering (e.g., a simple OpenGL wrapper or integration with a lightweight game engine for ease of 3D asset handling).

Prior Knowledge: A solid understanding of neural network fundamentals (CNNS specifically), linear algebra, 3D geometry, and basic graphics programming will be essential. Experience with real-time systems and optimization will also be beneficial.

References:

For marker-based tracking: ArUco Library documentation and related academic papers on fiducial markers.

For feature-based tracking: "Features from Accelerated Segment Test" (FAST) and "Oriented FAST and Rotated BRIEF" (ORB) papers for feature detection and description.

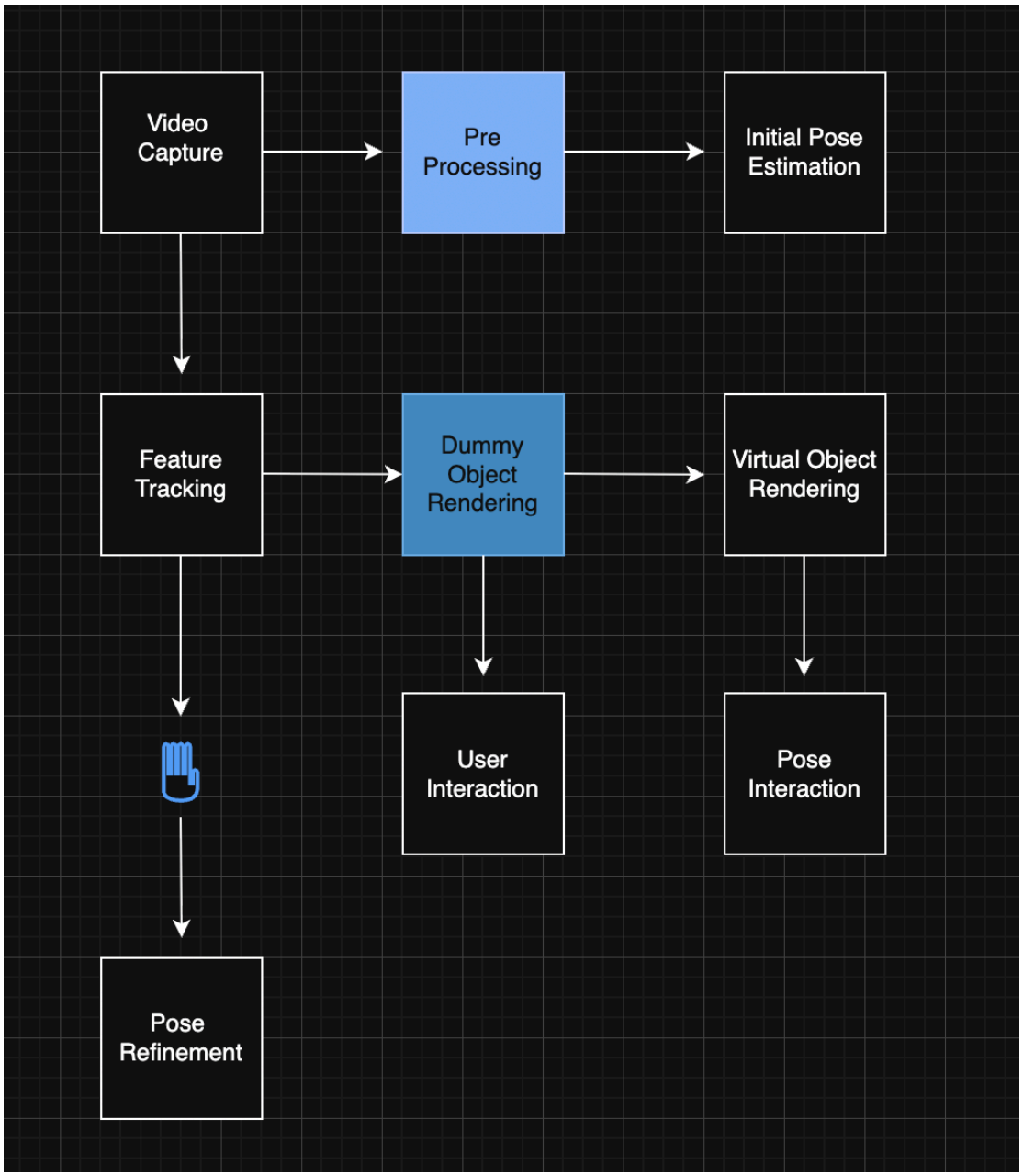
For robust tracking: Research papers on Visual Odometry and SLAM algorithms (e.g., ORB-SLAM, LSD-SLAM) for inspiration on advanced pose estimation techniques.

- **Summary (6 pts)**
    - What will I learn by doing this project?
    - How do I plan to present this as the final outcome to an audience to showcase my accomplishment and growth?

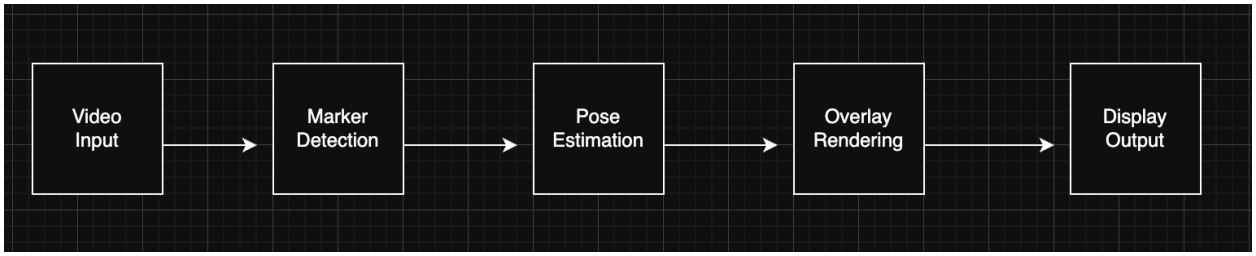
I anticipate gaining profound insights and practical experience in several areas. I will learn to implement and optimize real-time computer vision algorithms for object detection and feature tracking. Furthermore, I will deepen my understanding of 3D pose estimation, camera calibration, and the integration of virtual graphics with real-world video feeds. The iterative process of debugging and refining tracking stability will hone my problem-solving skills in dynamic environments and teach me the intricacies of managing computational resources for high-performance applications. I also expect to learn about the trade-offs between accuracy, speed, and robustness in AR systems.

The demo will showcase the system's ability to stably anchor multiple virtual objects (e.g., a virtual teapot, a small 3D model) onto a real-world surface (e.g., a table, a book), even as the camera moves around the scene. I will explicitly highlight the stability of the virtual objects, demonstrating minimal jitter or drift. A supplementary short video recording will also be provided, illustrating various scenarios and the different virtual objects supported.

Step by Step Diagram:



System Diagram:

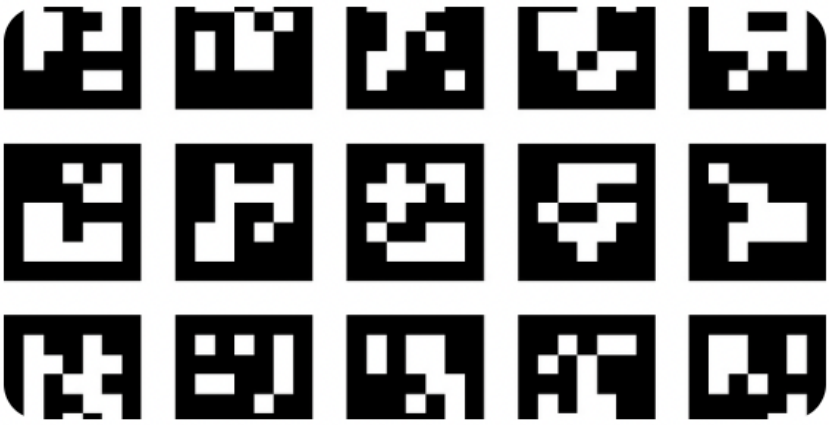


- **Resources/References (6 pts):**
  What resources, libraries, datasets, models do I plan to use for this project? Cite those resources.

Libraries:

- OpenCV: For all core computer vision tasks including video I/O, image processing, feature detection (FAST, ORB), feature description, and tracking algorithms (KLT, ArUco).
- NumPy: For efficient numerical operations and array manipulation in Python
- Pygame/PyOpenGL: For 3D rendering of virtual objects and integrating them into the video stream

Potential Datasets:

- While primarily focused on live camera input, if I explore training custom object detection models for specific virtual objects, I may use publicly available datasets like COCO (Common Objects in Context) or Open Images Dataset for pre-training, or create a small custom dataset for fine-tuning specific object recognition.
- https://www.kaggle.com/datasets/crsuthikshnkumar/aruco-marker-data-set
  - Contains a large set of classic ArUco markers, often used as a first baseline for pose estimation and marker-based AR workflow development
  - 
- https://exhibits.stanford.edu/data/catalog/ph459zk5920
  - Provides videos of both static and moving objects, with ground-truth 3D localization and diverse objects, ideal for mobile AR.

Models:

- Initially, I will rely on traditional computer vision algorithms for feature detection and tracking. If time permits and robustness becomes a major bottleneck, I might explore integrating lightweight, pre-trained machine learning models for object detection (e.g., MediaPipe Objectron or a compact YOLO model).

Academic References:

- E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," 2011 International Conference on Computer Vision, Barcelona, Spain, 2011, pp. 2564-2571, doi: 10.1109/ICCV.2011.6126544. keywords: {Boats},
- Garrido, C., Muñoz-Salinas, R., & Medina, J. M. (2014). Generation of fiducial markers based on circular symmetry. Pattern Recognition Letters, 47, 1-10.