Outline
Visual servoing theory
System implementation
Improvements & Applications
Questions
What is Visual Servoing?

Use of visual feedback to control robot motion

3 types
- Position Based (PBVS)
- Image Based (IBVS)
- Hybrid

Camera Configuration
- Mount to manipulator
- Fixed position

Diagram:
- Reference
- Measured error
- Measured output
- System input
- System output
- Controller
- System
- Sensor

Video / Image
Position Based Visual Servoing

Loop Tasks
- Capture scene
- Identify manipulator and targets
- Calculate pose
- Compute Error \( P_{\text{GOAL}} - P_{\text{CURRENT}} \)
- Evaluate inputs (PID Control)
- Actuate joints
How to Identify the Manipulator

Fiducial marker
- Distinct marker
- Simple to implement
- Sometimes not visible

Edge Detection
- Better for complex shapes
- Usually compare to computer model
- Can detect partially visible manipulator

60% of the time these work every time...
How to figure out pose

Projection: \( p(x, y) = KM_{EXT}P^w \)
- \( K \) = intrinsic camera parameters
  - Get from camera calibration
- \( M_{EXT} \) = extrinsic camera parameters
  - Transform matrix
  - World to Camera \( (^CT)^W \)

Solve least squares problem
- Or use OpenCV's solvePnP()
System
Pose

1. Estimate Pose
   - OpenCV’s solvePnP()

2. Convert to World Frame
   \[ p^W = W_C^T P^C \]
   - \( W_C^T \) is determined from measuring robot

\[ x = -11.5 \quad y = 62.9 \quad z = 395.6 \]
Control Loop

1. Get goal coordinates ($G^W$)
2. Get current position coordinates ($P^W$)
3. Use inverse kinematics to get joint angles
   - Current: ($\Theta_p, \phi_p$)
   - Goal: ($\Theta_g, \phi_g$)
4. Calculate joint angle error
   - $\Theta_e = \Theta_g - \Theta_p$
   - $\phi_e = \phi_g - \phi_p$
5. Calculate next angle step
6. Actuate servo motors
Improvements & Applications

Issues
- Render view not functional
- Limited servo angles
- No force feedback for hand
- Scaling to Multi DOF robot

Applications
- Interactive game
- Surgery
- Any dangerous, robot assisted job
Questions?