04 Problem Statement IPO

2025-10-19

Contents

1 Problem Statement + IPO Analysis - Vision-Based Pick and Place System

1.1 1. Problem Statement

1.1.1 1.1 Business Problem

Context: Manufacturing, warehousing, and logistics industries face challenges with: - **Labor-intensive** manual pick-and-place operations - **High error rates** in object sorting and placement - **Scalability limitations** due to workforce constraints - **Repetitive strain injuries** from repetitive manual tasks - **Inconsistent throughput** due to human fatigue - **Inability to operate 24/7** without shift rotations

Opportunity: Automate object detection, grasping, and placement using vision-guided robotics to: - Increase throughput (target: 30 picks/minute) - Reduce errors (target: <1% failure rate) - Operate continuously (24/7 with minimal supervision) - Handle varying object types/sizes (within defined workspace) - Ensure worker safety (collaborative operation)

1.1.2 1.2 Technical Problem

Challenge: Develop a robotic system that can: 1. **Perceive:** Detect and localize objects in a cluttered workspace 2. **Plan:** Compute collision-free trajectories to pick and place objects 3. **Execute:** Precisely grasp objects and place them at target locations 4. **Adapt:** Handle variations in object pose, lighting, occlusions 5. **Ensure Safety:** Operate safely near humans, detect collisions

Constraints: - **Real-time:** Vision processing <50ms, control loop 1kHz - **Accuracy:** Position repeatability ±0.1mm - **Reliability:** 99.9% uptime, graceful error recovery - **Cost:** Solution must be cost-effective (ROI <2 years)

1.1.3 1.3 Success Criteria

Metric	Target	Measurement Method
Pick rate	30 picks/min	Throughput test (100 cycles)
Grasp success rate	99%	1000-pick test, count failures
Positional accuracy	±0.1mm	CMM measurement at target location
Vision detection accuracy	95% mAP	Test on labeled dataset
Cycle time	2 sec/object	Time from scan to place completion
Uptime	99.5%	Track operational hours vs downtime
Safety incidents	0	Collision detection, E-stop tests

1.2 2. System-Level IPO (Input-Process-Output)

1.2.1 2.1 High-Level System IPO

```
                      SYSTEM BOUNDARY

  INPUTS                 PROCESS                 OUTPUTS
```

- Objects in workspace
- Target locations
- User commands (start/stop)
- Environmental data (lighting, obstacles)

VISION PERCEPTION

MOTION PLANNING

EXECUTION & CONTROL

- Objects at target locations
- Status reports
- Logs/ telemetry
- Alerts/ alarms

1.2.2 2.2 Detailed System IPO Table

Phase	Input	Process	Output
1. Scan	- RGB-D camera stream- Trigger command	- Capture image- Preprocess (denoise, crop)	- RGB image (1920x1080)- Depth map
2. Detect	- RGB image- Pre-trained model	- Run object detection (YOLO)- Filter low-confidence detections	- Bounding boxes- Class labels- Confidence scores
3. Localize	- Bounding boxes- Depth map- Camera calibration	- Extract 3D point cloud- Estimate 6DoF pose- Transform to robot frame	- Object pose (x,y,z,r,p,y)- Point cloud
4. Plan Grasp	- Object pose- Point cloud- Gripper constraints	- Compute grasp candidates- Rank by quality score- Select best grasp	- Gripper pose- Approach vector- Grasp quality score
5. Plan Motion	- Gripper pose- Current robot state- Obstacles	- Inverse kinematics- Path planning (RRT*)- Trajectory generation	- Joint trajectory (waypoints)- Collision-free path
6. Execute Pick	- Joint trajectory- Gripper command	- Send trajectory to controller- Monitor execution- Close gripper	- Joint positions (actual)- Grasp force feedback
7. Plan Place	- Target location- Current robot state	- Compute placement pose- Plan trajectory (pick → place)	- Placement trajectory- Orientation
8. Execute Place	- Placement trajectory- Gripper release command	- Move to target- Open gripper- Retract	- Object at target location- Task completion status

Phase	Input	Process	Output
9. Verify	- Camera image- Expected location	- Capture post-placement image- Verify object position	- Success/failure flag- Error magnitude

---

1.3 3. Module-Level IPO

1.3.1 3.1 Vision Perception Module

Purpose: Detect and localize objects in 3D space

Component	Input	Process	Output
Image Acquisition	- Camera trigger- Exposure settings	- Capture RGB-D frame- Sync RGB and depth	- RGB image- Aligned depth map
Preprocessing	- Raw RGB image	- Resize to 640x640- Normalize pixel values	- Preprocessed tensor (3x640x640)
Object Detection	- Preprocessed image	- Forward pass through YOLOv8- NMS (non-max suppression)	- Bounding boxes [x,y,w,h]- Class IDs- Confidence scores
Pose Estimation	- RGB-D image- Bounding box- Object model	- Extract object ROI- Run PnP or deep pose estimator- Refine with ICP	- 6DoF pose [x,y,z,qx,qy,qz,qw]- Covariance (uncertainty)
Point Cloud Gen.	- Depth map- Camera intrinsics	- Deproject pixels to 3D points- Filter outliers (statistical)	- Point cloud (XYZ + RGB)- Downsampled cloud
Coordinate Transform	- Object pose (camera frame)- TF tree (camera→robot)	- Apply homogeneous transformation- Publish TF	- Object pose (robot base frame)

IPO Summary:

```
INPUT: RGB-D frames (30 Hz)
PROCESS: Detection → Pose Estimation → Coordinate Transform
OUTPUT: Object poses in robot frame (x,y,z,roll,pitch,yaw) at 10 Hz
```

1.3.2 3.2 Grasp Planning Module

Purpose: Compute optimal gripper pose for stable grasping

Component	Input	Process	Output
Grasp Candidate Gen.	- Object point cloud- Gripper geometry	- Sample grasp poses (centroid, normals)- Check reachability	- List of candidate grasps [pose, score]

Component	Input	Process	Output
Collision Check	- Grasp candidates- Scene point cloud	- Check gripper-object collision- Check gripper-table collision	- Collision-free grasps
Grasp Ranking	- Grasp candidates- Force closure metrics	- Compute grasp quality (wrench space)- Sort by score	- Ranked list of grasps
Grasp Selection	- Ranked grasps	- Select top-ranked grasp- Fallback to 2nd if 1st fails	- Selected grasp pose- Approach vector

IPO Summary:

```
INPUT: Object pose, point cloud
PROCESS: Sample grasps → Filter collisions → Rank by quality → Select best
OUTPUT: Gripper pose (6DoF) + approach direction
```

1.3.3 3.3 Motion Planning Module (MoveIt2)

Purpose: Generate collision-free trajectories

Component	Input	Process	Output
Planning Scene	- Robot URDF- Point cloud (obstacles)	- Build occupancy grid- Update collision objects	- Planning scene (internal state)
IK Solver	- Target end-effector pose- Current joint state	- Solve inverse kinematics- Validate joint limits	- Joint angles [1.. 6]- IK success flag
Path Planner (OMPL)	- Start state- Goal state- Planning scene	- Run RRT*/PRM- Search for collision-free path	- Path (sequence of joint configs)
Trajectory Generator	- Path waypoints- Velocity/accel limits	- Time-parameterization (parabolic blend)- Smooth jerk	- Joint trajectory (time-stamped)
Trajectory Smoothing	- Raw trajectory	- Iterative optimization (shortcut, smooth)	- Smoothed trajectory

IPO Summary:

```
INPUT: Target pose (x,y,z,roll,pitch,yaw), obstacles
PROCESS: IK → Path Planning (RRT*) → Trajectory Generation → Smoothing
OUTPUT: Time-parameterized joint trajectory
```

1.3.4 3.4 Control & Execution Module (ros2_control)

Purpose: Execute trajectories with real-time feedback control

Component	Input	Process	Output
Trajectory Interpolator	- Joint trajectory- Current time	- Interpolate setpoints at control frequency (1kHz)	- Joint position/velocity setpoints
Joint Controller (PID)	- Setpoint- Actual position (encoder)	- Compute error- PID control law- Feedforward compensation	- Motor torque command
Motor Driver Interface	- Torque command	- Convert to current command- Send via EtherCAT	- Motor current (3-phase)
Feedback Loop	- Encoder position/velocity	- Read encoder data- Publish joint states	- Joint states (position, velocity, effort)
Safety Monitor	- Following error- Joint limits	- Check error bounds- Detect collisions (F/T sensor)	- Safety status- E-stop trigger

IPO Summary:

```
INPUT: Joint trajectory
PROCESS: Interpolate → PID Control → Motor Drive → Feedback
OUTPUT: Robot motion (joint positions), safety status
```

1.3.5 3.5 Gripper Control Module

Purpose: Actuate gripper (open/close) with force control

Component	Input	Process	Output
Gripper Controller	- Gripper command (open/close/force)	- Compute motor PWM- Apply force setpoint	- Gripper motor PWM signal
Force Feedback	- F/T sensor readings	- Measure grip force- Compare to setpoint	- Actual grip force- Force error
Position Feedback	- Encoder/limit switches	- Measure jaw opening- Detect object presence	- Jaw position- Grasp success flag

IPO Summary:

```
INPUT: Gripper command (open/close), target force
PROCESS: Force control loop (PID)
OUTPUT: Gripper state (open/closed), grip force
```

1.3.6 3.6 Task Orchestration Module (State Machine)

Purpose: High-level task sequencing and error recovery

Component	Input	Process	Output
State Machine	- System events (triggers)- Sensor data	- Evaluate state transitions- Execute state actions	- Current state- State outputs
Error Handler	- Fault signals (vision fail, grasp fail)	- Analyze failure type- Trigger recovery action	- Recovery command- Retry/abort decision
Task Scheduler	- Task queue- Robot availability	- Prioritize tasks- Dispatch to modules	- Task assignments- Execution order

States: 1. **IDLE:** Wait for start command 2. **SCAN:** Capture image 3. **DETECT:** Run vision pipeline 4. **PLAN_PICK:** Compute grasp and trajectory 5. **EXECUTE_PICK:** Move and grasp 6. **PLAN_PLACE:** Compute placement trajectory 7. **EXECUTE_PLACE:** Move and release 8. **VERIFY:** Check placement success 9. **ERROR:** Handle failures, retry or abort 10. **HOME:** Return to home position

IPO Summary:

```
INPUT: User commands, sensor events
PROCESS: State transitions based on events and conditions
OUTPUT: High-level commands to subsystems (vision, motion, gripper)
```

1.3.7 3.7 Monitoring & Logging Module

Purpose: Collect telemetry, logs, and metrics for observability

Component	Input	Process	Output
Data Collector	- ROS topics (joint states, images, etc.)	- Subscribe to topics- Timestamp data	- Time-series data streams
Logger	- Log messages (INFO, WARN, ERROR)	- Format logs- Write to file/database	- Log files- Database entries
Metrics Aggregator	- Performance metrics (latency, success rate)	- Compute statistics (mean, std, percentiles)	- Aggregated metrics
Alert Manager	- Metric thresholds- Anomalies	- Evaluate alert rules- Trigger notifications	- Alerts (email, SMS, dashboard)
Visualization	- Time-series data- Logs	- Render graphs (Grafana)- Display logs (Kibana)	- Dashboards- Real-time plots

IPO Summary:

```
INPUT: ROS topics, log messages, performance metrics
PROCESS: Collect → Store → Aggregate → Visualize
OUTPUT: Dashboards, alerts, historical logs
```

1.4 4. Data Flow Diagram

1.4.1 4.1 End-to-End Data Flow

```
   Camera    (RGB-D frames @ 30 Hz)



 Vision Pipeline  (Object poses @ 10 Hz)



 Grasp Planner    (Gripper pose)



 Motion Planner   (Joint trajectory)



 ros2_control     (Motor commands @ 1 kHz)



  Motor Drivers   (Currents to motors)



 Robot Actuators  (Physical motion)

        (feedback)

    Encoders      (Joint positions @ 1 kHz)

           (feedback to ros2_control)
```

1.5 5. IPO for Key Interfaces

1.5.1 5.1 Camera   Vision Pipeline

Aspect	**Details**
Input	USB 3.0 stream (RGB 1920x1080 @ 30fps, Depth 1280x720 @ 30fps)
Process	Image transport (compressed), synchronization (ApproxTime)
Output	ROS2 topics: `/camera/color/image_raw`, `/camera/depth/image_rect`
Latency	<30ms (USB transfer + decompression)
Data Rate	~200 MB/s (uncompressed), ~50 MB/s (compressed JPEG)

1.5.2 5.2 Vision Pipeline   Motion Planning

Aspect	**Details**
Input	Object pose (geometry_msgs/PoseStamped)
Process	ROS2 service call: `/compute_grasp` → `/plan_pick_motion`
Output	Joint trajectory (trajectory_msgs/JointTrajectory)
Latency	200-500ms (IK + planning)
Frequency	On-demand (per object detected)

1.5.3 5.3 Motion Planning   Control

Aspect	**Details**
Input	Joint trajectory (moveit_msgs/action/ExecuteTrajectory)
Process	ROS2 action interface (goal, feedback, result)
Output	Joint commands @ 1kHz (control_msgs/JointTrajectoryControllerState)
Latency	<10ms (action call overhead)
Frequency	1 kHz (controller loop)

1.5.4 5.4 Control   Motor Drivers

Aspect	**Details**
Input	Motor current commands (EtherCAT PDO, 16-bit signed int)
Process	EtherCAT cyclic communication (1 kHz)
Output	Motor phase currents (Ia, Ib, Ic)
Latency	<1ms (deterministic EtherCAT cycle)
Jitter	<10 s (EtherCAT distributed clocks)

1.6 6. Performance Requirements per Module

Module	Throughput	Latency	Accuracy	Reliability
Vision Perception	10 detections/sec	<50ms per frame	mAP 0.95	99% uptime
Pose Estimation	10 poses/sec	<100ms per object	±5mm position, ±5° orientation	95% accuracy
Grasp Planning	5 grasps/sec	<200ms per object	Quality score 0.8	90% success
Motion Planning	2 plans/sec	<500ms per plan	Collision-free (100%)	99% plan success
Trajectory Execution	1 kHz control loop	<10ms per cycle	Tracking error <2mm	99.9% uptime
Gripper Control	100 Hz	<10ms	Force ±1N	99% grasp success
Task Orchestration	1 task/2sec	<50ms state trans.	N/A	100% state integrity

---

1.7 7. Error Handling IPO

1.7.1 7.1 Error Detection Inputs

Error Type	Input Signal	Detection Method
Vision failure	No objects detected for >5 sec	Timeout counter
Grasp failure	F/T sensor: grip force <threshold	Force threshold check
Motion planning failure	Planner returns no solution	Planner status code
Collision	F/T sensor: force spike >150N	Anomaly detection (force)
Joint limit violation	Encoder position outside [q_min, q_max]	Range check
E-stop	Emergency stop button pressed	Digital input (hardwired)

1.7.2 7.2 Error Recovery Process

Error Type	Recovery Action	Outcome
Vision failure	Re-trigger camera, adjust lighting, rescan	Resume detection or abort
Grasp failure	Retry with alternate grasp, reduce speed	Retry (max 3x) or skip object
Motion planning failure	Relax constraints, replan with wider clearance	New plan or abort task
Collision	E-stop, retract, re-home robot	Safe state, await manual reset
Joint limit violation	Stop motion, move back to safe position	Resume from safe configuration
E-stop	Power off motors, log event, await user reset	System halted, manual intervention

---

1.8 8. IPO Summary Matrix

Module	Input	Process	Output	Frequency
Vision Perception	RGB-D frames	Detection + Pose Estimation	Object poses (robot frame)	10 Hz
Grasp Planning	Object pose, point cloud	Sample + Rank grasps	Gripper pose, quality score	On-demand
Motion Planning	Target pose, obstacles	IK + Path Planning	Joint trajectory	On-demand
Trajectory Execution	Joint trajectory	Interpolation + PID control	Motor commands	1 kHz
Gripper Control	Gripper command, force setpoint	Force control loop	Gripper state, grip force	100 Hz
Task Orchestration	User commands, sensor events	State machine transitions	High-level commands to modules	Event-driven
Monitoring & Logging	ROS topics, logs, metrics	Collect + Aggregate + Visualize	Dashboards, alerts	Continuous

---

1.9 9. Dimensional Analysis

1.9.1 9.1 Data Dimensions

Data Type	Dimensions	Units	Example Value
RGB Image	$1920 \times 1080 \times 3$	pixels (uint8)	[0-255] per channel
Depth Image	1280×720	mm (uint16)	0-10000 mm
Point Cloud	$N \times 3$ (XYZ)	m (float32)	N 100,000 points
Object Pose	7 (x,y,z,qx,qy,qz,qw)	m, quaternion (float64)	[0.5, 0.2, 0.1, 0,0,0,1]
Joint Angles	6 (1.. 6)	rad (float64)	[- ,]
Joint Trajectory	$T \times 6$ (time, positions)	s, rad	T 100 waypoints
Force/Torque	6 (Fx,Fy,Fz,Tx,Ty,Tz)	N, $N \cdot m$ (float64)	Fx [-100, 100] N
Gripper State	2 (position, force)	m, N	[0.05m, 20N]

1.10 10. Conclusion

This IPO documentation provides a **complete mapping** of: - **System-level** inputs, processes, and outputs - **Module-level** IPO for each major subsystem - **Data flow** through the entire pipeline (camera → motors) - **Interfaces** between modules with latency and data rate specs - **Error handling** IPO for fault detection and recovery

Key Takeaways: - **Vision → Planning → Control** pipeline with clear IPO at each stage - **Real-time performance** requirements (1 kHz control, <50ms vision) - **Dimensional consistency** enforced across all data types - **Error recovery** mechanisms for robust operation

---

Document Status: Complete **Last Updated:** 2025-10-18 **Author:** System Engineering Team **Review Status:** Pending Review