Automated Toolbox Inventory Control System [ATICS]

Sponsor: Hiline Engineering
Mentor: Dr. Neil Corrigan

Team Members:
Reem Osman, Project Manager
Caitlyn Powers
Caleb Thomas
Navin Sabandith
Steven Pixler

Background

Sponsor has many buildings, each containing several toolboxes

Employees occasionally forget to return tools after use, leading to:

- Disrupted workflows
- Delayed projects
- Increased operational costs

Auditing toolbox inventory is expensive and time consuming

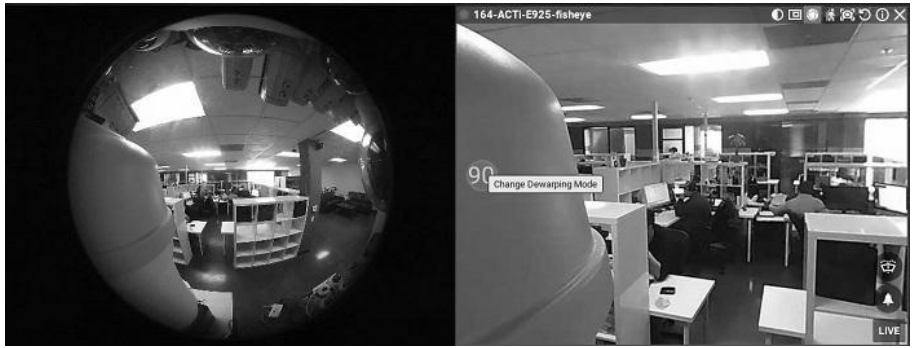
What if auditing process could be automated…

Solution

Develop a computer program implementing the following model:

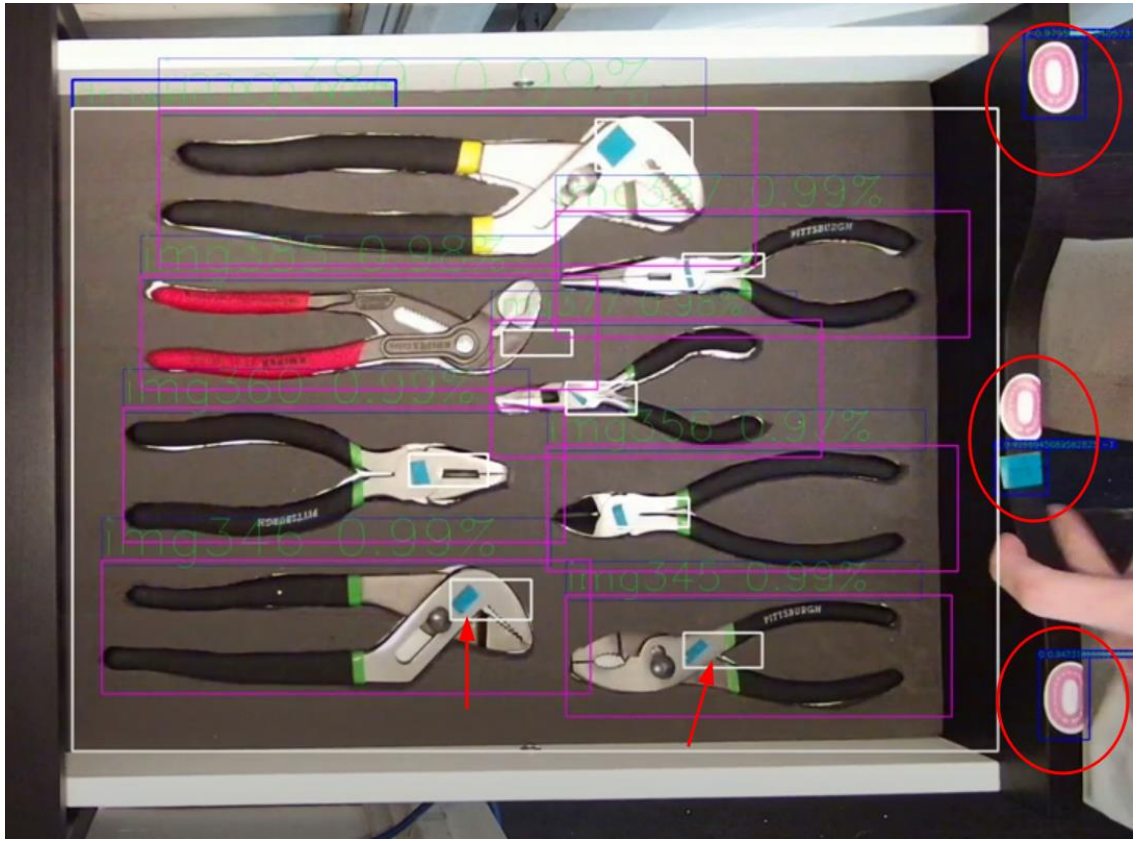
- Toolbox unlocked via RFID badge; triggers inventory audit
- Process video camera footage to identify missing tools
- Assign tools as checked out to employee
- Log tool checkouts in database

Initial Assumptions



- A Camera
  - No warping (fisheye lens)
  - No automatic Zoom
  - Streams over RTSP
- A ToolBox
  - Each Drawer
    - Unique identifying symbols on each drawer
    - Tools arranged in shadow box
  - Each Tool
    - Has Unique symbol
    - Assigned a single location in the drawer

Example Drawer and Tools



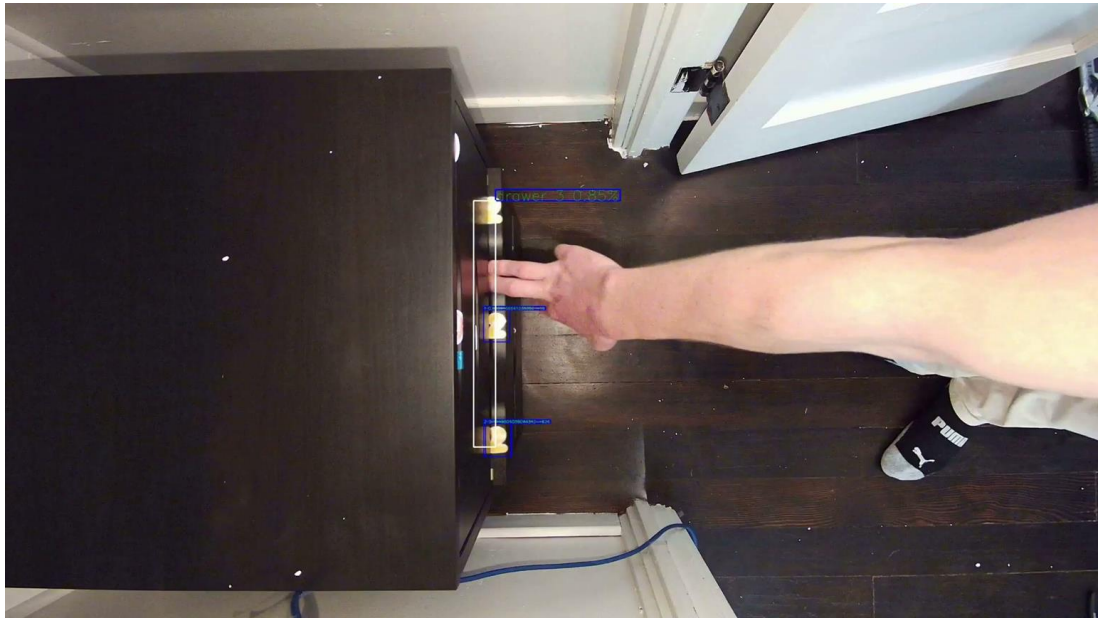
ATICS Components

- Computer Vision:
  - Template Matching
  - Classifier

- Backend Infrastructure :
  - API
  - Database
  - File Server

Computer Vision - Frame processing

- Find drawer using drawer symbols
- Locate tools and determine their status
- Look for extra tools in the drawer



Computer Vision - Template Matching

Template matching - A technique to match areas of an image to a template image.

- Takes 2 inputs:
  - Image in which an item is being searched for
  - Picture of the item to use as a template
- Inputs are used to compare the template picture to the image, moving the template one pixel at a time.
  - After each move, a metric is calculated to show how similar the template is to that particular area of the picture.

Template Matching - Challenges

Technology has difficulties with differences in:

- Lighting
- Rotation
- Size

Computer Vision - Classifier and Deep Neural Network

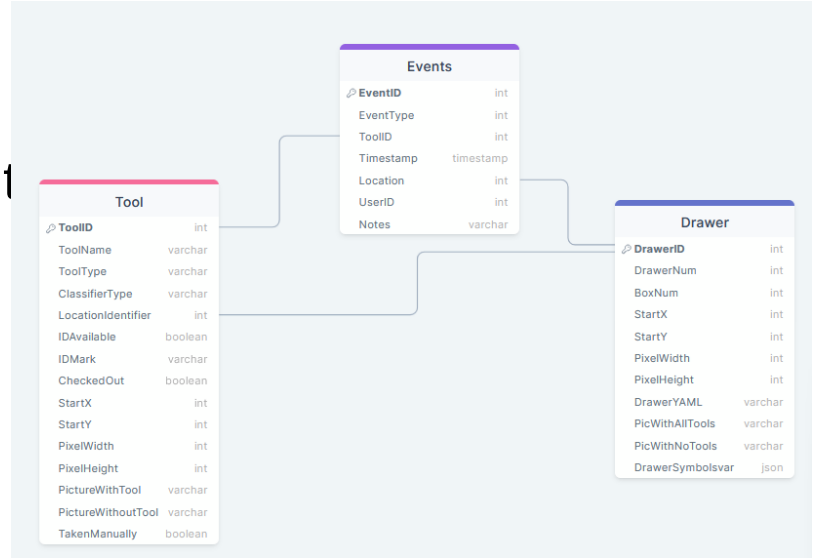
Image Classifier- A trained neural network designed to use PyTorch to use the computer to label images as a predetermined class.

- Training involves exposing the model to a dataset containing annotated images.
- Through iterative optimization, classifier learns to recognize patterns corresponding to the specified tools.
- The learned patterns are then exported as an ONNX configuration file to be used by the template matching portion of the program.

Backend Infrastructure

Database:

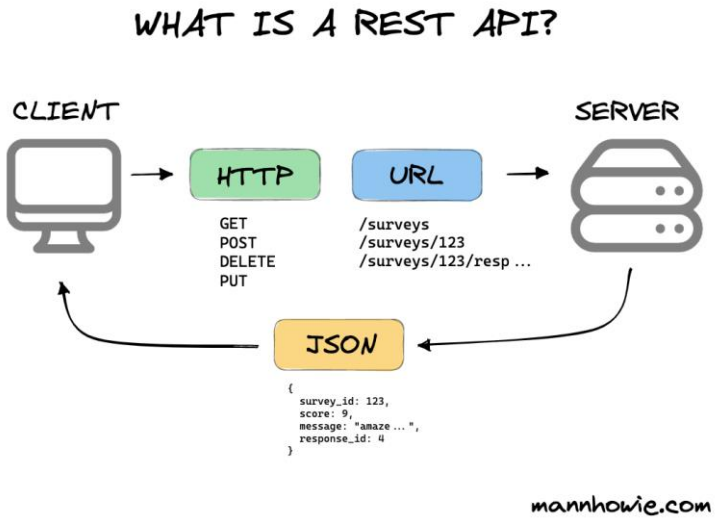
- MySQL server 8.0
- Stores information about tool boxes and drawers
  - Records tool location and checkout status
  - Records check in, check out, and error events
- Schema
- 3 Tables
  - Tools, Events, and Drawers



Backend Infrastructure

API:

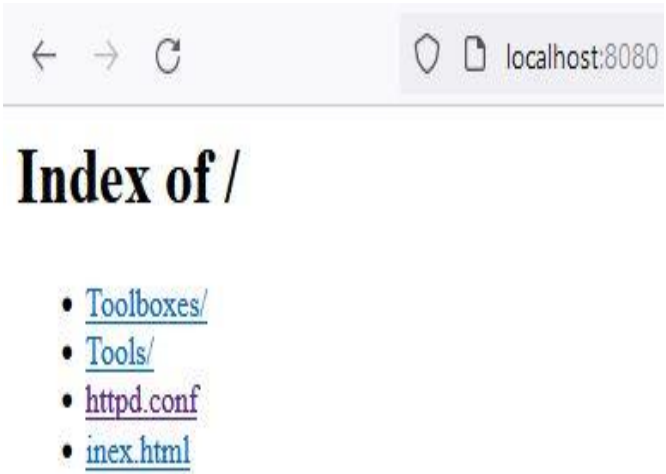
- Written with Flask and Python
- provides communication between the computer vision component and the SQL server.
- Queries are issued by custom python functions that send HTTP ‘GET’ and ‘POST’ requests to different API endpoints.



Backend Infrastructure

File Server:

- Web server which holds image and configuration data for drawers and tools
  - Apache server chosen - best at natively serving a directory
- Single, central server can be accessed by multiple toolboxes simultaneously
  - Eliminates the need to duplicate data across every toolbox

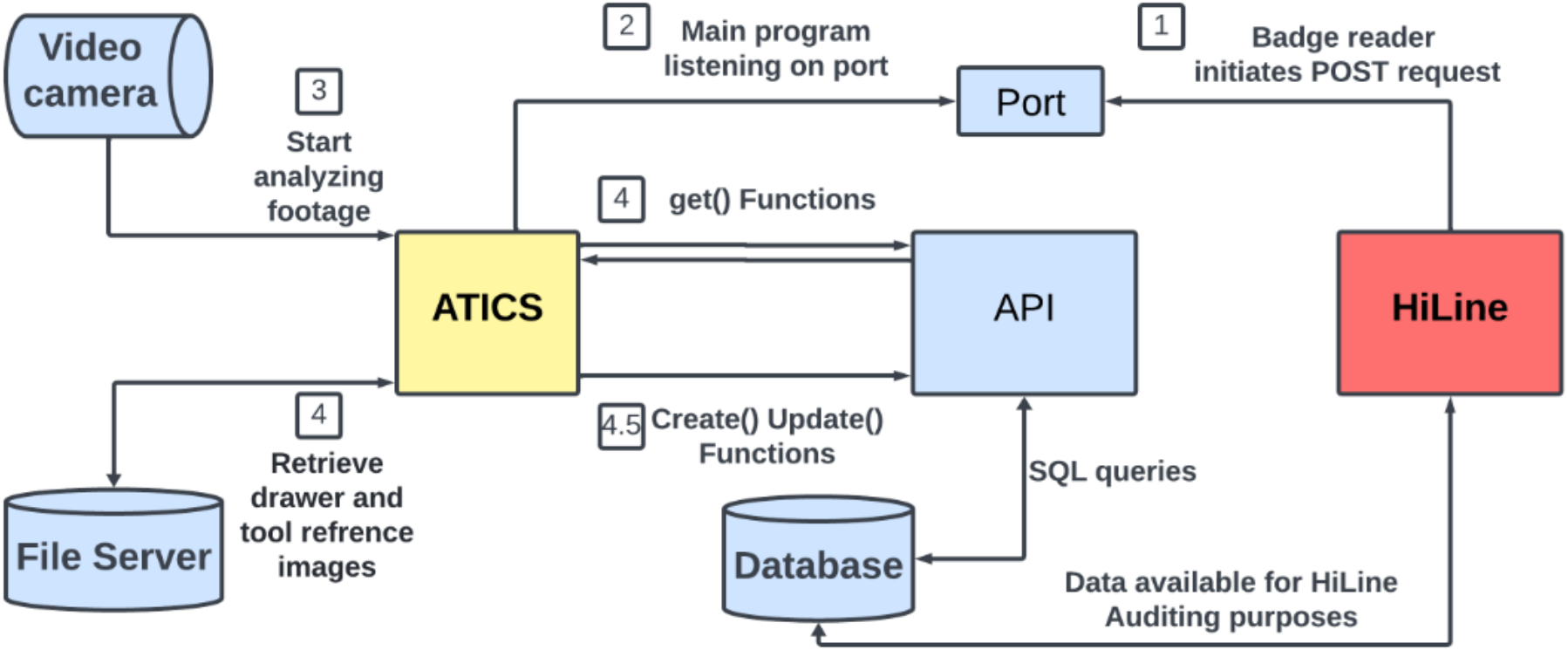


Backend Infrastructure

Architecture:

- Collection of microservices run on Docker
- Single command to start entire backend
  - `Docker compose up`
- Multiplatform - works on both Windows and Linux hosts
- Single physical server can host all services

Backend Flow Chart



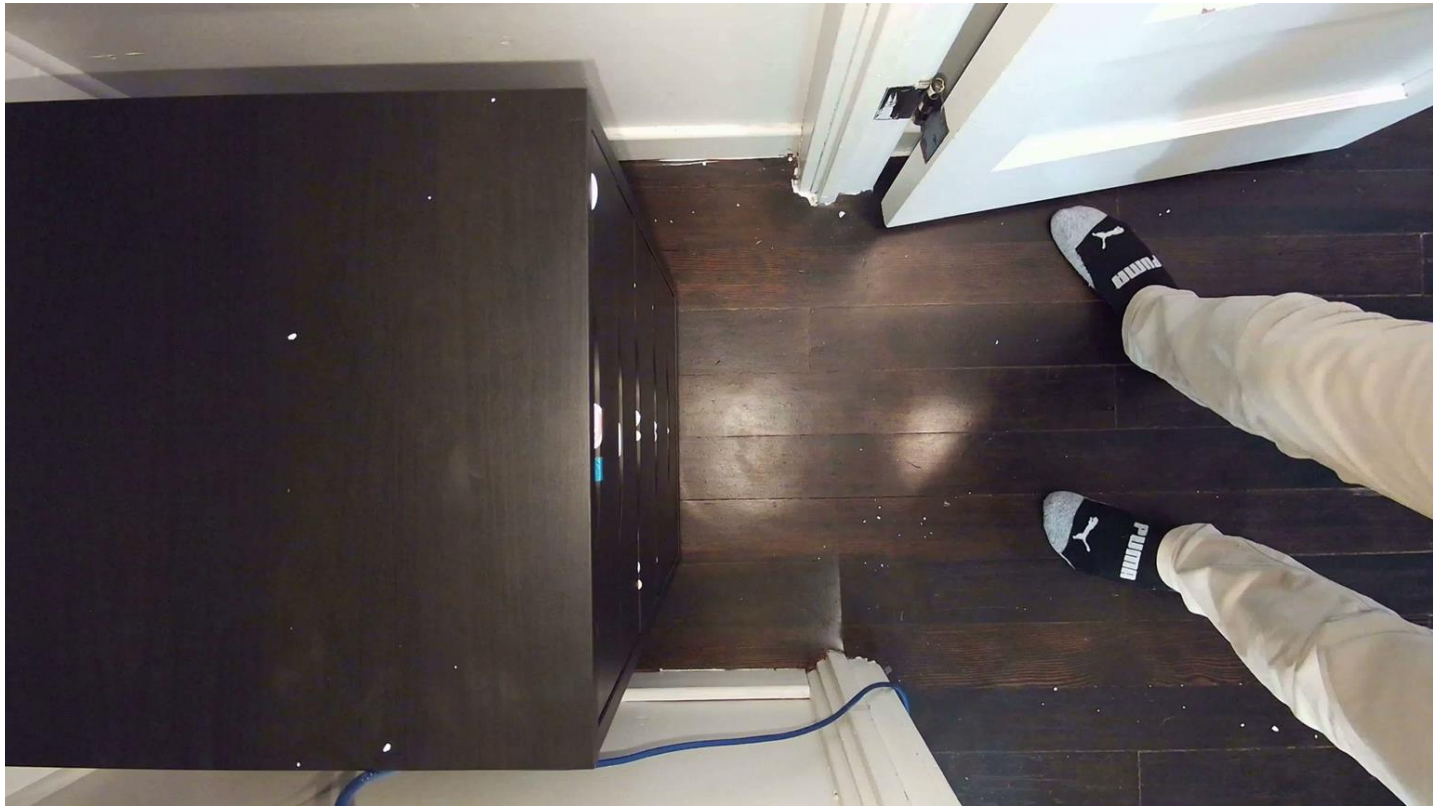
Result

Program creates:

- Bounding Box around tool and drawer displaying:
  - Name from the database
  - Status (Tool: checked out, checked in, or error)
  - Confidence percentage
- Output to the Database

Accuracy: 84%

Result - Video



Challenges

During the project, several challenges were encountered

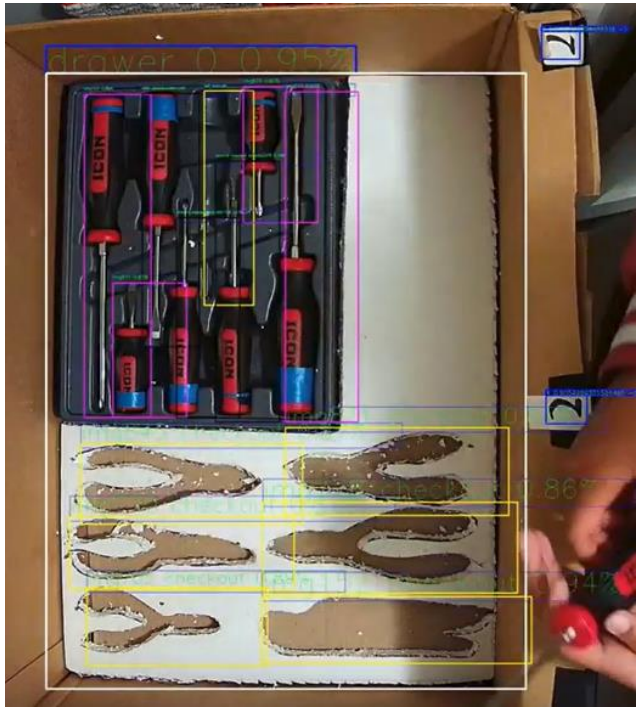
- Speed of execution
- Blurring when opening and closing the drawer
  - caused by the framerate not being high enough
- Small objects like sockets
  - difficult for computer vision to distinguish.

Future Work

- Increase speed
  - Multithread the program
  - Use specialized hardware to improve speed.
- Restrict small objects like sockets to the drawer closest to the camera.

Questions?

Initial testing setup



Example of a good template vs a bad template