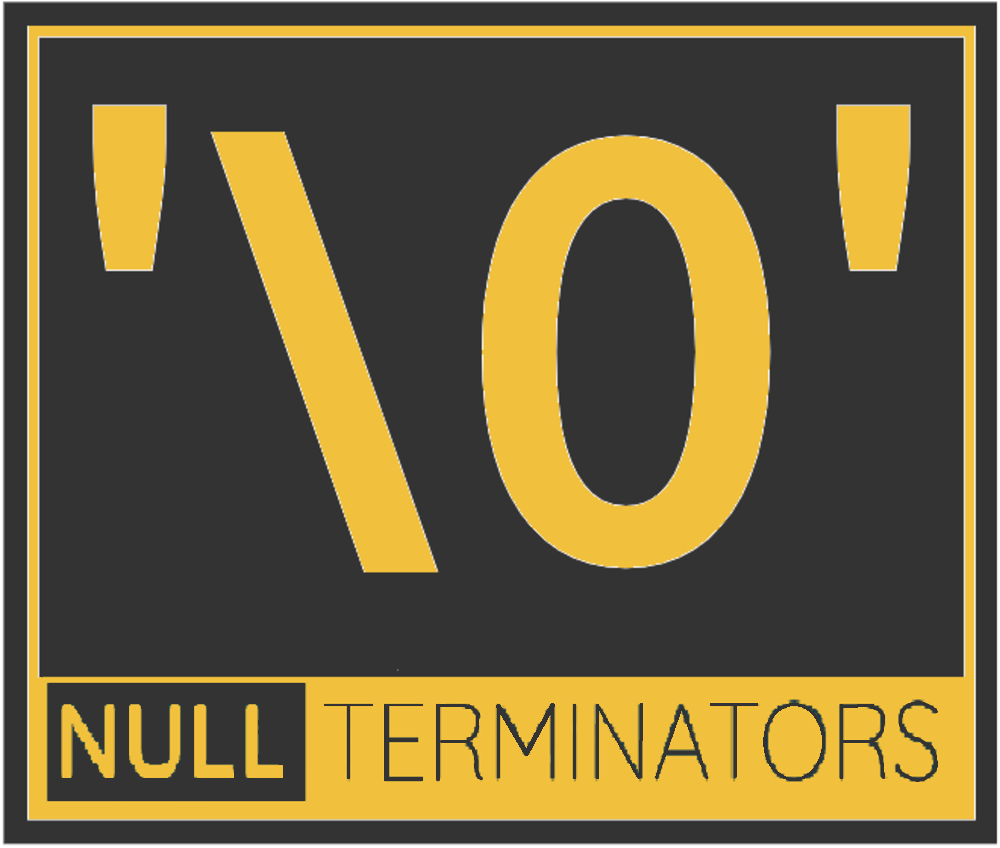
Automated Toolbox Inventory Control System

Design Document

Null Terminators

Version: 1.2





Senior Project Design Document

CptS 421/423

WSU Tri-Cities

1710 Crimson Way, Richland, WA 99354

**Title:**

Automated Toolbox Inventory Control System

**Project Team:**

Null Terminators

**Participants:**

Reem Osman (leader)

Caitlyn Powers

Steven Pixler

Navin Sabandith

Caleb Thomas

**Client:**

Troy Stokes, HiLine Engineering

**Client Technical Contact:**

Andrew Tolman, HiLine Engineering

**Supervisor:**

Dr. Neil Corrigan, WSUTC

**Table of Contents**

[**List of Figures 4**](#_4ir4ofw2pido)

[**Glossary 5**](#_rh04cuwsse0l)

[**Revision History 6**](#_3ed2krojcto6)

[**1.0 Introduction 7**](#_slk7xvm6kxqy)

[**2.0 Architecture 8**](#_619kd0dbguqw)

[2.1 AutomatedToolbox program 8](#_f2zamfjo84ti)

[2.1.1 Drawer Recognition 10](#_8uk4k5qvxokr)

[● Function: find\_drawer 10](#_a54g70lwrmbh)

[● Function: drawer\_location 11](#_852car9kgd9r)

[● Function: is\_open 12](#_savxiss83wqr)

[2.1.2 Template Match 13](#_eycyniosnkd8)

[● Function: drawtemp 14](#_7q6el9mk4tw4)

[● Function: rotate\_max\_area 16](#_sy3a8w9i94be)

[● Function: rotate\_bound 17](#_veupofro0kyu)

[● Function: rotated\_rect\_with\_max\_area 17](#_bl53ntcbe2vb)

[2.1.3 Helper functions 18](#_50egwt8notra)

[● Function: main 18](#_4y79rm1qaiik)

[● Function: wait\_for\_signal 20](#_s9zpmunkb6mq)

[● Function: get\_footage 21](#_hqqrurp7mi6t)

[● Function: create\_error\_records 22](#_9611w128e1ce)

[● Function: print\_records 23](#_amuf4k3q500)

[● Function: retrieve\_drawers 24](#_jdaxpc2i9lin)

[● Function: retrieve\_tools 24](#_d73jomhst7mt)

[● Function: update\_events 25](#_3c0r0xofk4x5)

[● Function: update\_tools 25](#_9opeixuy7erc)

[2.1.4 Tool recognition 26](#_x6ze97r2whtj)

[● Function: update\_tools\_for\_frame 27](#_eo296l37ho3k)

[● Function: drawer\_segment 28](#_6mw04moisspa)

[● Function: is\_visible 29](#_gvo4xjlibbxi)

[● Function: caclulate\_location 29](#_apcp4kqvwq4h)

[● Function: remove\_from\_contours 30](#_r7nofwuoqz1s)

[● Function: check\_extra\_tools 30](#_u4tu699xnlus)

[● Function: is\_checked\_out 32](#_6kp5hfj4p66f)

[● Function: classifier\_check 34](#_mm6ywnzhzp8p)

[● Function: symbol\_check 35](#_aympgdsn5jli)

[2.2 Additional script 36](#_3kf7ly6i1gj8)

[● Function: main 36](#_n5pcgrbjcg44)

[2.3 Classifier 38](#_nxnkoc4oqdky)

[● Function: data\_preprocessing 38](#_lo27xh1lett1)

[● Function: create\_model 38](#_gbkn06uddrd3)

[2.4 Signal Scripts 39](#_qzmelta0a2r0)

[● script: sendsignal.py 39](#_4s7pahexk21x)

[● script: sendsignal2.py 39](#_exrpvryx1bej)

[● script: makeDrawer\_1.py 39](#_bdsrj9jf1r5n)

[**3.0 Data Dictionary 41**](#_7elj14utxynv)

[3.1 Global configuration file 41](#_owgkimriudo4)

[3.2 Drawer configuration file 42](#_tt6q2maqbozw)

[3.3 Database Schema 43](#_ryy7lu5jw550)

[3.3.1 Tables 43](#_w65abb1iu3w)

[● Relationship Table 43](#_lxutg1loqy77)

[● Tools Table 43](#_u6414hfjpulj)

[● Drawer Table 44](#_3ju2wpom5rln)

[● Events Tabel 45](#_ppih87ye5a8u)

[3.3.2 API Endpoints | Custom function call 45](#_xrxqnr7ywxe9)

[● Endpoint: get\_tools\_info | Function: getToolsInfo() 45](#_arc61evr26yu)

[● Endpoint: get\_drawers\_info | Function: getDrawersInfo() 46](#_iwaal0pn2e5q)

[● Endpoint: add\_event | Function: addEvent() 46](#_vj6rfj2ugkrl)

[● Endpoint: update\_tool | Function: updateToolsInfo() 46](#_3unm9vnzz5rt)

[● Endpoint: add\_drawer | Function: addDrawer() 47](#_l6975enqd5rw)

[● Endpoint: add\_tool | Function: addTool() 47](#_ic00dnp9shed)

[3.4 Signal Schema 48](#_54f3jnngr7ug)

[**4.0 User Interface 49**](#_ez68hg68ghoc)

[4.1 Command Line 49](#_jl1b28sry1t2)

[4.1.1 Input 49](#_852l8rxl5f73)

[4.1.2 Output 49](#_kgipzl4mb4jr)

[**5.0 Information Repositories 50**](#_96b76fjzfmuw)

[5.1 Multimedia File Server 50](#_eapkpngpsad4)

# 

# List of Figures

[Figure 2.1.0.1 Data Flow Diagram of Program](#_er7dw0z97s4p) 9

[Figure 2.1.0.2 Structure chart of program](#_z9ssyphaj8px) 9

[Figure 2.1.1.1 Data Flow Diagram of drawer recognition functions](#_77itxgbk445o) 10

[Figure 2.1.2.1 Data flow Diagram of template match functions 1](#_t5n6lxgw65s0)4

[Figure 2.1.4.1 Data Flow Diagram of tool recognition functions 2](#_lkxpnaq4e4os)6

# 

# 

# 

# 

# 

# 

# 

# 

# Glossary

WSU Washington State University

WSUTC WSU Tri-Cities - branch campus of WSU located in Richland, WA

HiLine HiLine Engineering - Mechanical Engineering firm located in Richland, WA

RFID Radio Frequency IDentification - technology used by HiLine for employee badges

MVP Minimum Viable Product - the smallest set of features required in a final product

UI User Interface - an interface through which users interact with a system

RTSP Real-Time Streaming Protocol - network protocol for video streaming

ONNX Open Neural Network Exchange - free neural network design format

# Revision History

1.2 - 04/27/2024 Remove Highlights; Fix typos auth. Caleb Thomas

1.1 - 03/08/2024 Added more description auth. Null Terminators

1.0 - 03/06/2024 First published version auth. Null Terminators

# 1.0 Introduction

As a condition of graduation, Washington State University (WSU) requires bachelor’s students in their final year to complete a “Senior Design Project” showing mastery of the course material covered throughout the program. For students in the Computer Science program, this takes the form of a yearlong project where students form a team and act as a small development firm given the responsibility of completing a software development project for a commercial sponsor.

This document outlines the design the Null Terminators have come up with to construct an automated tool recognition system for HiLine engineering. This document reflects a functional oriented design methodology, in which the program is broken into a series of interacting pieces or modules, each with a distinct function. This model is a good fit for more experimental projects, and software where there isn’t a need for a hierarchical relationship between elements. Since this project is designed as a distributed system, organizing development by function is the most efficient solution.

The basic architecture of this system involves a camera mounted on a toolbox, and a computer where machine learning and computer vision is used to identify which tools are present, and which tools are not. Information about a tool’s location, status, and who checked out the tool will be stored in a single, centralized database. Any photos, videos, or media files needed by the program will be stored on a media server, and an API Gateway server will allow each toolbox system to safely modify information in the database. Having just one repository for each type of information makes the system much more scalable and easy to manage, as well as easier to get reporting information from. The database, API Gateway, and media server will each be distributed as Docker containers - the industry standard for developing microservices.

For ease of reading, the design document has been broken into several sections. Section 2.0 outlines the architecture of the program. Following the principles of function oriented programming, the program is broken down into constituent functions which are then defined with their purpose, needed parameters, and output. The diagrams at the beginning of the section show how the functions fit together to produce the program as a whole. Section 3.0 describes the format of data storage used by the program, with particular attention being paid to the database schema. Section 4.0 shows what command line options are available for users to start the program, and section 5.0 defines the format of the folder structure used by the media server to store images and videos needed as input for the computer vision system.

# 2.0 Architecture

In this section, we will discuss three distinct programs implemented in python: the 'automatedtoolbox' program, the additional script, and the classifier; along with example scripts that will send signals to the 'automatedtoolbox' program. The 'automatedtoolbox' program serves as the project's main component, utilizing computer vision for tool check-in and check-out processes. The additional script, a small single-function script, aids in gathering essential pictures and information for the database, which is then used by the ‘automatedtoolbox’, to determine if tools are checked out. The classifier functions are employed to train the PyTorch classifier before exporting the classifier into the ONNX format, to be utilized by the ‘automatedtoolbox' program. We will also talk about some examples of sending signals to the ‘automatedtoolbox’ program.

## 2.1 AutomatedToolbox program

The program is organized into distinct sections: drawer recognition, helper functions, template matching, and tool recognition. Each section is named to convey its purpose, with the exception of the helper functions, which are exclusively utilized within the main function. The main function serves as the program's starting point and is located within the helper functions section. In the event of runtime errors, an event is triggered and logged in the events list. These events can be viewed either through terminal output or stored in the events database. Subsequently, the program will either terminate or await further instructions.

Diagrams are provided to aid in understanding both the pseudocode and the program's functionality.

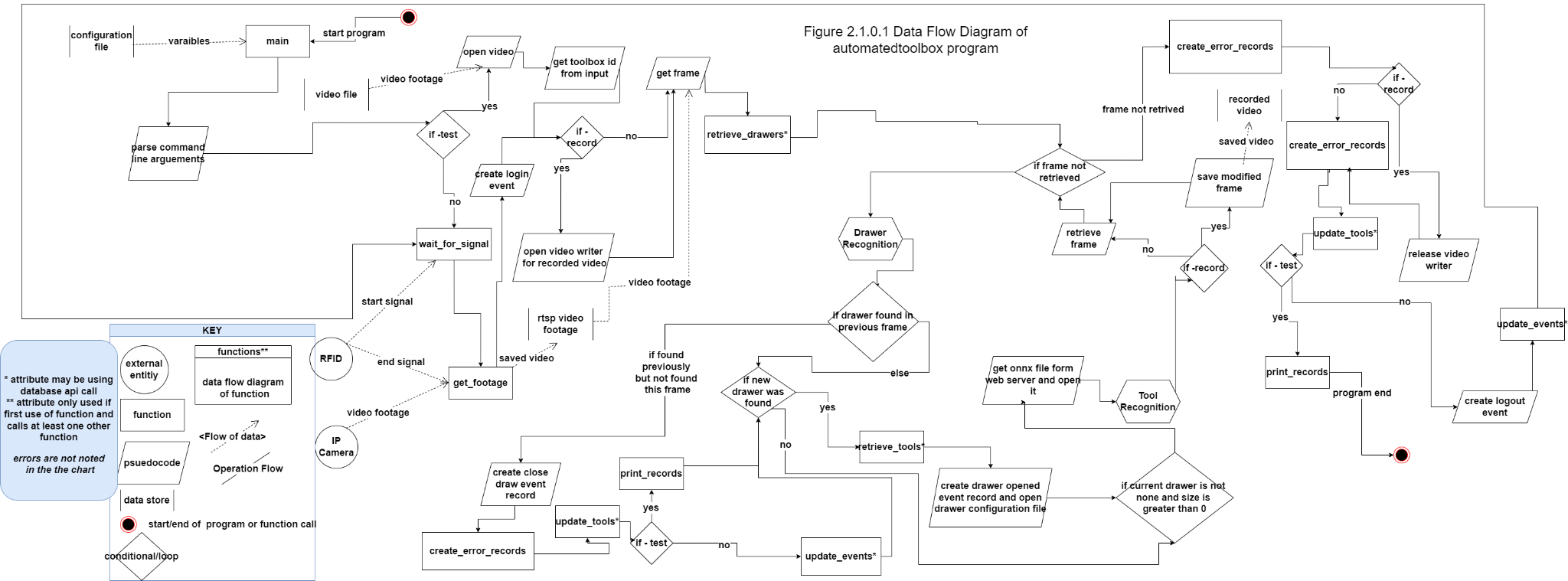


Figure 2.1.0.1 Data Flow Diagram of automatedtoolbox program

|  |
| --- |
| Figure 2.1.0.2 Structure chart of automatedtoolbox program |

### 2.1.1 Drawer Recognition

|  |
| --- |
| Figure 2.1.1.1 Data Flow Diagram of drawer recognition functions |

#### Function: find\_drawer

* + Function Description :
    - Finds which if any drawer is open and calculates where in frame draw is and then returns the modified frame, the drawer and the drawerSize
  + Parameters:
    - drawers:
      * Purpose: json list of drawers in the toolbox
      * Type: json object
    - frame:
      * Purpose: current frame of the video
      * Type: cv2 image object
    - record:
      * Purpose: is it in record mode
      * Type: bool
  + Pseudo-Code:
    - Loop over drawers
      * if drawer is open
        + find where in frame the drawer is
        + then return the frame with any drawings for the record mode on it, the json object reflecting the drawer that was open, and the calculated width and height of the drawer in the frame
    - if drawer not in frame return modFrame and have drawer and drawerLocation be returned as None
  + Return:
    - modFrame:
      * Purpose: drawn on frame
      * Type: cv2 image object
    - drawer:
      * Purpose: the drawer that was found
      * Type: json object
    - drawerLocation:
      * Purpose: location of drawer in frame
      * Type: Tuple : (x,y,w,h)

#### Function: drawer\_location

* + Function description :
    - Find drawer in the picture and return the height and width of the drawer in the frame, drawer a rectangle around the drawer if in record mode
  + Parameters :
    - modFrame:
      * Purpose: current frame of video to use for drawing if in record mode
      * Type: cv2 image object
    - place:
      * Purpose: pixel location that drawer symbol was found
      * Type: Tuple :(x,y)
    - similarity:
      * Purpose :similarity score of the best matching template
      * Type: float between 0 and 1
    - drawer:
      * Purpose :json object of found drawer
      * Type: json object
    - template
      * Purpose: json object of drawer symbol
      * Type: json object
    - record:
      * Purpose: is it in record mode
      * Type: bool
  + Pseudo-Code:
    - find difference between templates saved location and the place it was found
    - subtract difference from the width and height of the drawer
    - if in record mode
      * draw a rectangle around the drawer and put text describing the similarity score and what drawer it is.
    - return the position of the drawer in the frame
  + Return:
    - drawLocation:
      * Purpose: pixel location of drawer in frame
      * Type: Tuple :(x,y,w,h)

#### Function: is\_open

* + Function Description :
    - Find if the drawer symbols are visible in the frame
  + Parameter :
    - frame:
      * Purpose: clean version of current frame of video
      * Type: cv2 image object
    - modFrame:
      * Purpose: current frame of video to use for drawing if in record mode
      * Type: cv2 image object
    - templates:
      * Purpose: json list of 3 drawer symbols
      * Type :json object
    - record:
      * Purpose: is it in record mode
      * Type: bool
  + Pseudo-Code:
    - database returns drawer symbols as a string so turn that string into a json
    - loop over templates
      * retrieve drawer symbol from webserver
      * use the reference picture of the drawer symbol to template match
      * if a match is found and that match is a better match than the previous templates and the difference between the y difference between where it was found and where it is suppose to be is not greater than multfordrawersymbolbuffer\* buffery , set that matches values as the values to be returned
        + (checking the y difference is to make sure its not confusing one of that drawers symbols for another of that drawers symbols)
      * if in record mode and found is true put text labeling the symbol, the similarity score and the y difference between where it was found and the template
    - return foundMax, foundTemplate, placeMax, similarityMax
  + Return:
    - foundMax
      * Purpose: if template was found
      * Type: bool
    - foundTemplate:
      * Purpose: template found
      * Type: json object
    - placeMax:
      * Purpose: pixel location of template
      * Type: Tuple : (x,y)
    - similairtyMax
      * Purpose: similairt score of template match
      * Type: float between 0 and 1

### 2.1.2 Template Match

|  |
| --- |
| Figure 2.1.2.1 Data flow Diagram of template match functions |

#### Function: drawtemp

* + Function Description :
    - Finds the best match of the template in the picture above a certain threshold of similarity
  + Parameter :
    - template:
      * Purpose: cv2 image object of the template
      * Type: cv2 image object
    - frame:
      * Purpose: cv2 image object of the frame or picture that should contain the object of the template
      * Type: cv2 image object
    - modFrame:
      * Purpose: Current frame of video to use for drawing if in record mode
      * Type: cv2 image object
    - w:
      * Purpose: Width of the template in pixels
      * Type: int
    - h:
      * Purpose: Height of the template in pixels
      * Type: int
    - color1:
      * Purpose: Color of drawing on frame
      * Type: Tuple: (red,green, blue)
    - threshold:
      * Purpose: Minimum similarity score
      * Type: real 0 to 1
    - draw:
      * Purpose: Is it in -record mode
      * Type: bool
    - degrees:
      * Purpose: The angle positive and negative the match is allowed to be off by
      * Type: Int between 0 and 180
    - degreeDiv:
      * Purpose: What fraction of a degree to rotate
      * Type: Int greater than 1
  + Pseudo-Code:0
    - initialize values used in loop
    - while x <= degrees\*degreeDiv
      * if x does not equal 0
        + rotate template by x
      * if template is larger than or equal to the given image crop the template
      * try to template match template and image
        + if get a cv2.error (Assumed to be issue with sizes of template and image because of human error with input)

if 0>= x

x = x-1

template = temp

* + - * find the maximum match and its location in the image
      * if peak\_value >= similarity
        + set similarity to peak\_value
        + if peak\_value >= threshold

set values to be max values and values to be returned

* + - * turn x negative
      * if 0>= x
        + x = x-1
      * template = temp
    - if draw == 1 and found is true
      * draw on frame
      * return found,place and similarity
  + Return:
    - found:
      * Purpose: if match was found
      * Type: bool
    - place:
      * Purpose: The x,y coordinates of the start and of the match and the width and height in pixels
      * Type: Tuple: (x,y)
    - similarity :
      * Purpose: The similarity score of the match
      * Type: int between 0 and 1

#### Function: rotate\_max\_area

* + Function Description :
    - Rotates image by degrees, using rotated\_rect\_with\_max\_area and rotate\_bound to remove the added black borders
  + Parameter :
    - image:
      * Purpose: Image to be rotated
      * Type: cv2 image matrix object
    - angle:
      * Purpose: Amount to be rotated in degrees
      * Type: real
  + Pseudo-Code[[1]](#footnote-0):
    - get max rectangle
    - rotate image
    - crop rotated image to max rectangle area
    - return rotated image
  + Return:
    - rotatedObject:
      * Purpose: rotated image
      * Type: cv2 image object

#### Function: rotate\_bound

* + Function Description:
    - Rotates the picture
  + Parameter:
    - image:
      * Purpose: Image to be rotated
      * Type: cv2 image matrix object
    - angle:
      * Purpose: amount to be rotated in degrees
      * Type: real
  + Pseudo-Code[[2]](#footnote-1):
    - uses cv2.getRotationMatrix2D and cv2.warpAffine to rotate the picture
  + Return:
    - rotatedObject:
      * Purpose: rotated image with black borders
      * Type: cv2 image object

#### Function: rotated\_rect\_with\_max\_area

* + Function Description :
    - Given a rectangle of size w,h that has been rotated by 'angle' (in radians), computes the width and height of the largest possible axis-aligned rectangle (maximal area) within the rotated rectangle.
  + Parameter:
    - w
      * Purpose: The width of the image in pixels
      * Type: int
    - h
      * Purpose: The height of the image in pixels
      * Type: int
    - angle:
      * Purpose: Amount to be rotated in radians
      * Type: real
  + Pseudo-Code[[3]](#footnote-2):
    - Calculate the largest rectangle
  + Return:
    - wr
      * Purpose: Max width in pixels
      * Type: int
    - hr
      * Purpose: Max height in pixels
      * Type: int

### 2.1.3 Helper functions

#### Function: main

* + Function Description:
    - The starting function of the program, it controls the overall process of the program.
  + Input :
    - -test
      * Purpose: Is it in -test mode, and if so the file location of the video it the automatedtoolbox program is looking at.
      * Type: string
    - -record
      * Purpose: Is the program in -record mode
      * Type: bool
  + Pseudo-Code:
    - parse command line arguments, and initialize values
    - try infinite while loop ( only exited if in -test mode or run into an error)
      * if - test
        + open video
        + get input from user for toolboxID
      * else
        + wait for start signal
        + try to get footage
        + if can’t find the location of footage given

close socket

create event

update events in database

continue to wait for signal again (Assumed to be issue with only one of many toolbox rtsp links)

* + - * + create login event
      * if -record
        + open video writer for the recorded video
      * if video from -test or get footage is open
        + try retrieve drawers
        + if exception occurs

create event for the exception

if -test

print records to terminal

else

update events in database

end program

* + - * + set current timestamp to start time stamp
        + try to process frames :

loop over video given by -test or get\_footage until out of frames

find drawer within frame

if a drawer was found in previous frame but not found this frame

create close drawer record

create error records

update tools

if - test

print records

else

update events database

if current drawer is found for the first time

try retrieve tools

if exception occurs

create event for the error

if -test

print records to terminal

else

update events

exit

create drawer opened event record

get and open drawer configuration file

create old and new tools from retrieved tools

if current drawer doesn’t equal none

retrieve onnx file and open it as a neural network

figure out if tools have been checked in or out and if their are any errors in the drawer

if -record

save modified frame

get frame

update current timestamp

release video from get footage or test

if -record

release video writer

create error records

update tools

if - test

print records to terminal and end program

remove temporary video made by get footage

create logout event

update events

* + - * + if exception occurs

create error record for issue processing frames

create error record

if old tools and new tools does not equal none

update tools

if test

print records to terminal

exit program

esle

update events to database

continue infinite while loop

* + - * else
        + create error event for video file not opening
        + if - test

print records to terminal and end program

* + - * + else

update events to database

continue infinite while loop

* + - if exception occurs
      * create event for the exception
      * if -test
        + print records to terminal
      * else
        + update events in database
      * end program
  + Output:
    - Dependent on mode see section 4.1.2 Output.

#### Function: wait\_for\_signal

* + Function Description:
    - Waits for start signal on port from host, sends heartbeat while waiting
  + Parameter :
    - hostIP
      * Purpose: Address of the item sending the signal
      * Type: string
        + can be the empty string
    - port:
      * Purpose: The port numberer the signal will be sent to
      * Type: int
  + Pseudo-Code:
    - create jsonschema
    - bind socket
    - listen on socket
    - wait for signal
      * while waiting send heartbeat periodically
      * if signal received check to make sure it is correct format using jsonschema.validate
        + if it is not correct continue
        + if it is correct get timestamp and break out of loop
    - return signal, startimeStamp, connection, and socket
  + Return:
    - startTimeStamp:
      * Purpose: The time stamp that the system received the signal
      * Type: dateTime object
    - data:
      * Purpose: The signal received
      * Type: json object
    - s
      * Purpose: The socket that was used to get signal
      * Type: socket object

#### Function: get\_footage

* + Function Description : Gets RTSP stream until have gotten end signal
  + Parameter:
    - rtspStream
      * Purpose: RTSP url to get footage from
      * Type: string
    - savedFootage
      * Purpose: file location to save footage
      * Type: string
    - host
      * Purpose: address of the item sending the signal
      * Type: string
    - port
      * Purpose: the port numberer the signal will be sent to
      * Type: int
    - startTimeStamp:
      * Purpose: The time stamp that the system received the signal
      * Type: dateTime object
    - s
      * Purpose: The socket that was used to get signal
      * Type: socket object
  + Pseudo-Code:
    - create jsonschema
    - open RTSP stream
    - check if opened
      * start videowriter using saved footage as location
      * while end singal not received or endtimeStmap <=endtimestamo
        + if end signal not received get signal

try connecting to socket and receiving data without blocking

if blocking error created

ignore the error

if signal received

test if correct signal with jsonschma.validate

if it is get end timestamp

if it isn’t continue

* + - * get frame
      * if frame not retrieved continue
      * else write the frame to videowriter and update timestamp
    - else
      * raise file not found error
    - close videowriter
    - close socket
    - open written video as savedVideo
    - return endTimestamp and savedVideo
  + Return:
    - endTimestamp
      * Purpose: The time stamp that the system received the end signal
      * Type: datetime object
    - savedVideo
      * Purpose: Footage that was saved
      * Type: cv2 video object

#### Function: create\_error\_records

* + Function Description :
    - Make the error events and append them to the events list
  + Parameter :
    - events
      * Purpose: json list of events
      * Type: json object
    - errors
      * Purpose: json list of errors
      * Type: json object
  + Pseudo-Code:
    - loop over errors
      * add them to events
      * increment events
    - return updated events
  + Return:
    - updatedEvents
      * Purpose:json list of events
      * Type:json object

#### Function: print\_records

* + Function Description:
    - Print the events in the events list to the terminal.
  + Parameter :
    - events
      * Purpose: json list of events
      * Type: json object
    - toolBoxID
      * Purpose: the id of the toolbox
      * Type: int
  + Pseudo-Code:
    - loop over events
      * print item in event with different formats depending on the event’s eventType value
        + 0 -> drawer opened
        + 1 -> drawer closed
        + 2 -> tool checked out
        + 3 ->tool check in
        + 4 -> wrong tool error
        + 5-> extra tool error
        + 6 -> runtime error
        + tool login and logout should not appear
  + Return:
    - Nothing

#### Function: retrieve\_drawers

* + Function Description :
    - Retrieve drawers that are in the given toolbox from the database.
  + Parameter :
    - toolBoxID
      * Purpose: The id of the toolbox
      * Type: int
  + Pseudo-Code:
    - create json schema for error message
    - create url with APIgatewayurl and title of api call
    - database api call getDrawersInfo
    - check if matches schema
      * if it does not then no eros have occurred and return drawers
      * if it does then raise error
  + Return:
    - drawers
      * Purpose: list of drawers in the database for the toolbox
      * Type: json object

#### Function: retrieve\_tools

* + Function Description :
    - Retrieve the tools that are in the given drawer from the database.
  + Parameter :
    - drawerID
      * Purpose: The ID of the drawer
      * Type: int
  + Pseudo-Code:
    - create json schema for error message
    - create url with APIgatewayurl and title of api call
    - database api call getToolsInfo
    - check if return of getToolsInfo matches schema
      * if it does not
        + loop over tools list and append ‘timestamp’ and ‘error’ (necessary info for tracking checkin/checkout between frames)
        + return tools
      * if it does raise runtime error
  + Return:
    - tools
      * Purpose: list of tools in database of the drawer
      * Type:json object

#### Function: update\_events

* + Function Description :
    - Add the events from the events list to the events table in the database.
  + Parameter :
    - events
      * Purpose: list of events
      * Type: json object
  + Pseudo-Code:
    - create url with APIgatewayurl and title of api call
    - for event in evets list
      * database api call addEvent
  + Return:
    - indicator:
      * Purpose: indicates success of function
      * Type: bool

#### Function: update\_tools

* + Function Description :
    - Update the status of the tools in the database and add tool checkout and checkin events to the events list.
  + Parameter :
    - oldTools
      * Purpose: Original list of tools
      * Type: json object
    - newTools
      * Purpose: Updated list of tools
      * Type: json object
    - events
      * Purpose: list of events
      * Type: json object
    - test
      * Purpose: is it in - test mode
      * Type:bool
  + Pseudo-Code:
    - if newtools and oldtools don’t equal none
      * zip together tool lists and loop over them
        + create url with APIgatewayurl and title of api call
        + if status changed

if not -test mode

database api call updateToolsInfo

if its a checkout event

create checkout event

if its a check in event

create checkin event

* + - return events
  + Return:
    - events
      * Purpose: updated events list
      * Type: json object

### 2.1.4 Tool recognition

|  |
| --- |
| Figure 2.1.4.1 Data Flow Diagram of tool recognition functions |

### 

#### Function: update\_tools\_for\_frame

* + Function Description : finds tools and errors in the drawer
  + Parameter :
    - frame
      * Purpose: Clean current frame of video
      * Type: cv2 image object
    - modFrame:
      * Purpose: Current frame of video to use for drawing if in record mode
      * Type: cv2 image object
    - tools
      * Purpose: List of tools in drawer
      * Type: json object
    - errors
      * Purpose: List of errors in drawer
      * Type: json object
    - drawerLocation
      * Purpose: Location of drawer in frame in pixels
      * Type: Tuple:(x,y,w,h)
    - timestamp
      * Purpose: Timestamp of frame
      * Type: datetime object
    - drawer
      * Purpose: json object of drawer
      * Type: json object
    - configuration
      * Purpose: The drawer configuration file
      * Type: yaml object
    - classifier:
      * Purpose: Computer vision tool object recognition model used to classify image
      * Type: onnx classifier
    - userID
      * Purpose: UserID to use when making error records
      * Type: int
    - record
      * Purpose: Is it in record mode
      * Type: bool
  + Pseudo-Code:
    - set up onnx and drawer configuration file
    - segment the drawer
    - loop over list of tools
      * if tool is visible
        + remove any contours in the contours list that are where the tool is
        + crop the frame to just the tool using toollocation information and global buffer values
        + see if its checked in or out or if there is an error
        + if tools table status changed and error not associated with it

change tool information

* + - * + if else error is associated with the tool

set tool error to 1

* + - check if there are extra tools
    - return updated tools list and errors list
  + Return:
    - updatedTools
      * Purpose: Updated list of tools
      * Type: json object
    - updatedErrors
      * Purpose: Updated list of errors
      * Type: json object

#### Function: drawer\_segment

* + Function Description :
    - Create list of contours of the frame
  + Parameter:
    - frame
      * Purpose: Current frame of video
      * Type: cv2 image object
    - drawerLocation
      * Purpose: Location of the drawer
      * Type: Tuple:(x,y,w,h)
  + Pseudo-Code:
    - if segment in drawer configuration file is -1
      * return empty list
    - crop frame to just drawer
    - threshold frame
    - find contours
    - return contours
  + Return:
    - contours
      * Purpose: list of contours found in image
      * Type: list of lists of vectors of points

#### Function: is\_visible

* + Function Description :
    - Checks if tool is visible in the frame and if so where it is
  + Parameter :
    - tool
      * Purpose: json object of tool
      * Type: json object
    - drawerLocation
      * Purpose: location of the drawer in the frame
      * Type: Tuple :(x,y,w,h)
    - drawer
      * Purpose: json object of drawer
      * Type: json object
    - buffer
      * Purpose: Amount of abject that must be visible
      * Type: real between 0 and 1
  + Pseudo-Code:
    - get tool location
    - if toolLocation w\*h >= tool total w\*h\*buffer
      * tool is visible
    - else
      * tool is not visible
    - return if visible and toolLocation
  + Return:
    - toolLocation
      * Purpose: location of tool in the frame
      * Type: Tuple (x,y,w,h)
    - visible
      * Purpose: Is the tool visible
      * Type: bool

#### Function: caclulate\_location

* + Function Description:
    - Calculates location of tool in drawer
  + Parameter:
    - tool
      * Purpose: json object of tool
      * Type: json object
    - drawer
      * Purpose: json object of drawer
      * Type: json object
    - drawerLocation
      * Purpose: Location of the drawer in the frame
      * Type: Tuple:(x,y,w,h)
  + Pseudo-Code:
    - find difference between drawer saved location and the place it was found
    - check how far tool start x should be from the drawer start x
    - if tool is partially visible
      * subtract the x difference from the width of the tool and subtract the y difference from the tool start Y, and set starting x location to be the drawer X location
    - else (tool is fully visible but may be shifted)
      * shift the x and y starting locations according to the difference
  + Return:
    - toolLocation
      * Purpose: location of tool in frame
      * Type: Tuple:(x,y,w,h)

#### Function: remove\_from\_contours

* + Function Description:
    - Checks contours from drawer segmentation toolLocation
  + Parameters:
    - contours
      * Purpose: list of contours in frame
      * Type: list of lists of vectors of points
    - toolLocation
      * Purpose: location of tool in frame
      * Type: Tuple:(x,y,w,h)
    - drawerLocation
      * Purpose: Location of the drawer in the frame
      * Type: Tuple:(x,y,w,h)
  + Pseudo-Code:
    - make polygon object using tool locations
    - loop over list of contours
      * make polygon of contour
      * use polygon functions to see if tool is in contour or contour in tool and if it isnt append the contour to a new list of contours
    - return new contours list
  + Return:
    - contours
      * Purpose: list of contours in frame
      * Type: list of lists of vectors of points

#### Function: check\_extra\_tools

* + Function Description :
    - Checks if there is any extra tools
  + Parameters:
    - tools:
      * Purpose: List of tools in drawer
      * Type: json object
    - contours
      * Purpose: list of contours in frame
      * Type: list of lists of vectors of points
    - errors
      * Purpose: list of errors in drawer
      * Type: json object
    - timeStamp
      * Purpose: timestamp of frame
      * Type: datetime object
    - drawer
      * Purpose: json object of drawer
      * Type: json object
    - drawerLocation
      * Purpose: location of drawer in frame in pixels
      * Type: Tuple:(x,y,w,h)
    - frame
      * Purpose: Clean current frame of video
      * Type: cv2 image object
    - modFrame:
      * Purpose: current frame of video to use for drawing if in record mode
      * Type: cv2 image object
    - classifier
      * Purpose: Computer vision tool object recognition model used to classify image
      * Type: onnx classifier
    - userID
      * Purpose: UserID to use when making error records
      * Type: int
    - record
      * Purpose: Is it in record mode
      * Type: bool
  + Pseudo-Code:
    - crop frame to be just the drawer
    - loop over list of contours
      * if is greater than minwidth and height from drawer configuration
        + crop frame to be just the contour area
        + classify what is in contour
        + create polygon for contour
        + if it is a tool

loop over error list

create polygon for error

loop over errors in error list

create polygon for error from the inner loop

use polygon functions to see if any of the errors are inside the other

remove error

break

use polygon functions to see if any of the errors are inside of the contour or contour inside of the error and if previous if statment in loop was never entered

remove from errors

add to updated errors

if it is in record mode

draw rectangle

if else previous if in loop was never entered

make error record

* + - loop over original errors list
      * if it is visible
        + add to new error list
      * if error is for tool home and tool home is still an error
        + add to new error list
    - return new error list
  + Return:
    - newErrors
      * Purpose: Updated list of errors in drawer
      * Type: json object

#### Function: is\_checked\_out

* + Function Description:
    - Determines if a tool is checked in or out or if there's an error
  + Parameters:
    - image
      * Purpose: Image of tool
      * Type: cv2 image object
    - modFrame
      * Purpose: frame ot be drawn on
      * Type: cv2 image object
    - tool
      * Purpose: json object of tool
      * Type: json object
    - toolLocation
      * Purpose: location of tool in frame
      * Type: Tuple:(x,y,w,h)
    - threshold
      * Purpose: Minimum similarity score
      * Type: real 0 to 1
    - thresholdSymbol
      * Purpose: Minimum similarity score for symbols
      * Type: real 0 to 1
    - degrees
      * Purpose: The angle positive and negative the match is allowed to be off by
      * Type: int between 0 and 180
    - degreesDiv
      * Purpose: What fraction of a degree to rotate
      * Type: int greater than 1
    - errors
      * Purpose: list of errors in drawer
      * Type: json object
    - timeStamp
      * Purpose: timestamp of frame
      * Type: datetime object
    - drawerID
      * Purpose:the ID of the drawer
      * Type: int
    - symbolBuffer
      * Purpose: Amount of abject that must be visible
      * Type: real between 0 and 1
    - userID
      * Purpose: UserID to use when making error records
      * Type: int
    - record
      * Purpose: Is it in -record mode
      * Type: bool
  + Pseudo-Code:
    - read in template images from webserver
    - if tool is partially visible
      * crop template so only the same part of it is visible.
      * set status to not entirely visible
    - template match with no tool
    - template match with tool
    - if tool similarity > than no tool similarity or template matches not found
      * check what type of tool it is
      * if correct type of tool or tool type is none or not completely visible and check symbol and match for tool was found
        + create text for tool checkout and create color
        + set checked out to 1
      * else
        + if the classifier says its not a tool and similarity of no tool match + thresholdadd > tool similarity

create text for tool checkin and color

set checked out to 0

* + - * + else

create text and color for error

if tool does not already have error

make error record

set checked out to -1

* + - else
      * create text for tool checkin
      * set checkout to 0
    - if in record mode
      * draw rectangle around tool and put text
    - return checked out and errors( modified in place)
  + Return:
    - checkedOut:
      * Purpose: Is tool checked out, checked in or a error found
      * Type: int
        + 0 - tool is checked out
        + 1 - tool is checked in
        + -1 - error found
    - errors
      * Purpose:updated list of errors in drawer
      * Type: json object

#### Function: classifier\_check

* + Function Description :
    - Checks what type of tool is in the picture
  + Parameters:
    - classifier:
      * Purpose: Computer vision tool object recognition model used to classify image
      * Type: onnx classifier
    - image:
      * Purpose: Image to put into classifier
      * Type: cv2 image object
  + Pseudo-Code:
    - normalize image as close to possible as normalization on training set when creating classifier
    - give image to classifier to classify
    - find highest score
    - return label of the index with the highest score
  + Return:
    - typestring
      * Purpose: What tool the classifier thinks it is
      * Type: string

#### Function: symbol\_check

* + Function Description:
    - Checks if the symbol is in the image, returns true if tool is not suppose to have a symbol or symbol is found
  + Parameters:
    - symbolBuffer
      * Purpose: Amount of abject that must be visible
      * Type: real between 0 and 1
    - toolLocation
      * Purpose: location of tool in frame
      * Type: Tuple:(x,y,w,h)
    - tool
      * Purpose :json object of tool
      * Type: json object
    - image
      * Purpose: Image of tool in frame
      * Type: cv2 image object
    - modFrame:
      * Purpose: Current frame of video to use for drawing if in record mode
      * Type: cv2 image object
    - threshold
      * Purpose: Minimum similarity score
      * Type:real 0 to 1
    - degrees
      * Purpose: The angle positive and negative the match is allowed to be off by
      * Type:int between 0 and 180
    - degreesDiv
      * Purpose: What fraction of a degree to rotate
      * Type: int greater than 1
    - record
      * Purpose: Is it in -record mode
      * Type: bool
  + Pseudo-Code:
    - if their is supposed to have a symbol and the enough of the tool is visible that it should be their
      * retrieve symbol from webserver
      * template match symbol to image
      * found is results of template match
      * if in record mode and symbol found
        + draw rectangle around symbol
    - else
      * found is true
    - return found
  + Return:
    - found:
      * Purpose: if symbol was found
      * Type: bool

## 2.2 Additional script

Errors are not handled in this script; the script will simply end the program and print the error to the command line.

#### Function: main

* + Function Description:
    - Create initial drawer and tool json’s of drawer using thresholding and image segmentation, help determine the appropriate drawer configuration file settings, and create some of the necessary template images
  + Input:
    - toolPicture
      * Purpose: file location of picture of drawer with all tools in it
      * Type: string
    - noToolPicture
      * Purpose: file location of picture of drawer with no tools in it
      * Type: string
    - confName
      * Purpose: Name of drawer configuration file, it is optional, default name is conf.yaml
      * Type: string
  + Pseudo-Code:
    - load config file
    - process and threshold image
    - find contours
    - open tools.json
    - loop over contours
      * get dimensions of contour
      * if width and height of contour greater than minimum
      * if think it is drawer
        + increase dimensions by buffers
        + draw rectangle around drawer in both images
        + make cropped image of drawer in from both images
        + open drawer.json
        + put info into drawer.json
        + close drawer.json
      * else
        + increase dimensions by buffers
        + draw rectangle around tool in both images
        + make cropped image of tool from both images
        + put info into tool.json
    - close drawer.json
  + Output: All of these are put in a folder called drawer in the same folder as the additona lscript , it will create the folder if it does not exist and will overwrite files if a file of the same name already exists
    - drawer.json
      * Purpose: Initial json of drawer has same categories as a row in drawer database
      * Type: json file
    - tools.json
      * Purpose: Initial json of tools list, has same categories as a row in tools database, is invalid because it has an extra comma, which is put in to prevent the user from putting it in database without looking at and modifying the file themself
      * Type: json file
    - imgbounded1.jpg
      * Purpose: Image of segmented drawer from first image to help with debugging
      * Type: jpeg file
    - imgbounded2.jpg
      * Purpose: Image of segmented drawer from second image to help with debugging
      * Type: jpeg file
    - \*.jpgs
      * Purpose: Images of tool in tools.json with tool and without tool
      * Type: jpeg file

## 2.3 Classifier

#### Function: data\_preprocessing

* + Function Description:
    - This function preprocesses the data by splitting it into training and testing sets, applying transformations, and creating data loaders.
  + Parameter:
    - Data\_root:
      * Purpose: Root directory where the data is stored.
      * Type: string
  + Pseudo-Code:
    - get class names from folder names
    - initialize training and testing lists
    - define data transformations
    - create training and testing datasets and data loaders
  + Return:
    - trainloader
      * Purpose: Load batches of training data with their labels
      * Type: torch.utils.data.dataloader
    - Testloader:
      * Purpose: Loads the data to be run against the resnet model
      * Type: torch.utils.data.dataloader

#### Function: create\_model

* + Function Description:
    - This function creates a pre-trained ResNet model with modified fully connected layers for the given dataset.
  + Parameter: N/A
    - Classes: N/A
    - Purpose: List of class labels in the dataset.
    - Type: list
  + Pseudo-Code:
    - load resnet model
    - modify for your classes, criterion and optimizer values
  + Return:
    - Resnet:
      * Purpose: Pre-Trained classifier
      * Type: Tensor
    - Criterion
      * Purpose: List of values being check
      * Type: String
    - Optimizer:
      * Purpose: Ensures that criterion are met
      * Type: String

## 2.4 Signal Scripts

There are three example scripts: sendsignal.py, sendsingal2.py, and makeDrawer\_1.py. Sendsignal.py is a script that can send the automatedtoolbox.py a valid start signal, and receive the heartbeat signal. Sendsignla2.py is a script that can send the stop signal to the automatedtoolbox script. MakeDrawer\_1.py can use the API functions to upload the drawer contained in drawer.json and the the tools contained in tools.json. There is also a bash script called “instalreq.bash “ which helps with setting up all the libraries needed to run the program.

#### script: sendsignal.py

* + Function Description:
    - Sends a start signal to the automatedtoolbox program.
  + Parameter:
    - None
  + Pseudo-Code:
    - Connect to port on host
    - receive heartbeat signal from the automatedtoolbox program.
    - send encoded json of start signal to automatedtoolbox over connection.
  + Return:
    - Prints data received from automated toolbox program.

#### script: sendsignal2.py

* + Function Description:
    - Sends stop signals to the automatedtoolbox program.
  + Parameter:
    - None
  + Pseudo-Code:
    - Connect to port on host
    - send encoded json of stop signal to automatedtoolbox over connection.
  + Return:
    - None

#### script: makeDrawer\_1.py

* + Function Description:
    - Put modified drawer.json and tools.json from additional script into the database.
  + Parameter:
    - drawer.json
      * Purpose: Information that describes drawer
      * Type: json object
    - tools.json
      * Purpose: Information that describes tools
      * Type: json object
  + Pseudo-Code:
    - open drawer json
    - upload drawer to database using add\_drawer api function
    - open tools
    - for all tools
      * upload tool to database using add\_tool
  + Return:
    - Print to terminal whether the drawer and each tool was added successfully, and the return value of the add\_drawer and add\_tool functions. .

# 3.0 Data Dictionary

## 3.1 Global configuration file

* Name: Global\_Config.yaml
* Purpose: Holds configuration values for the automated toolbox program
* Variables
  + APIgatewayurl: string
    - url of the API gateway server which is how database is accessed
  + webserverurl: string
    - url of the multimedia file server
  + rfidport: int
    - port rfid will send signals too
  + rfidhost: string
    - IP address rfid will use to send signals
      * if empty string will just listen to all ip addresses
  + fps: int
    - frames per second of the RTSP stream or given video
  + RTSP: array of strings
    - array of RTSP urls
      * index of array should equal the toolboxnum of the toolbox the camera is attached to
  + onnxlabels: array of strings
    - array of labels of onnx classes
      * index of array should match index of score for that class in the array that onnx returns
  + onnxtools: array of strings
    - array of tools that onnx has classes for
  + onnxnontools:array of strings
    - array of non-tool things that onnx has classes for
  + thresholdsymbol: real between 0 and 1
    - threshold for symbols
  + thresholdtool: real between 0 and 1
    - threshold for tools
  + degrees: int between 0 and 180
    - maximum rotation to try to find template match
  + degreediv: int between 1 and infinity
    - what fraction of a degree to rotate
  + onnxfile: string
    - file location of onnx classifier
  + buffer:real between 0 and 1
    - amount of tool that must be in frame to be considered visible
  + bufferx: int greater than 0
    - how many pixels to expand the horizontal start and end of the tool space in the picture by
  + buffery: int greater than 0
    - how many pixels to expand the vertical start and end of the tool space by
  + thresholdignoreonnx: 0 to 1
    - If the tool confidence threshold is above this amount from template matching ignore the results of the onnx tool model.
  + thresholdadd : 0 to 1
    - If the onnxclassifier recognizes the image as a nontool add this much to the confidence interval of tool checked out to see if it is now greater than the tool checked in
  + multfordrawersymbolbuffer: 1 to infinity
    - multiplied by buffery to calculate area in which the drawer symbol should be on the y axis,

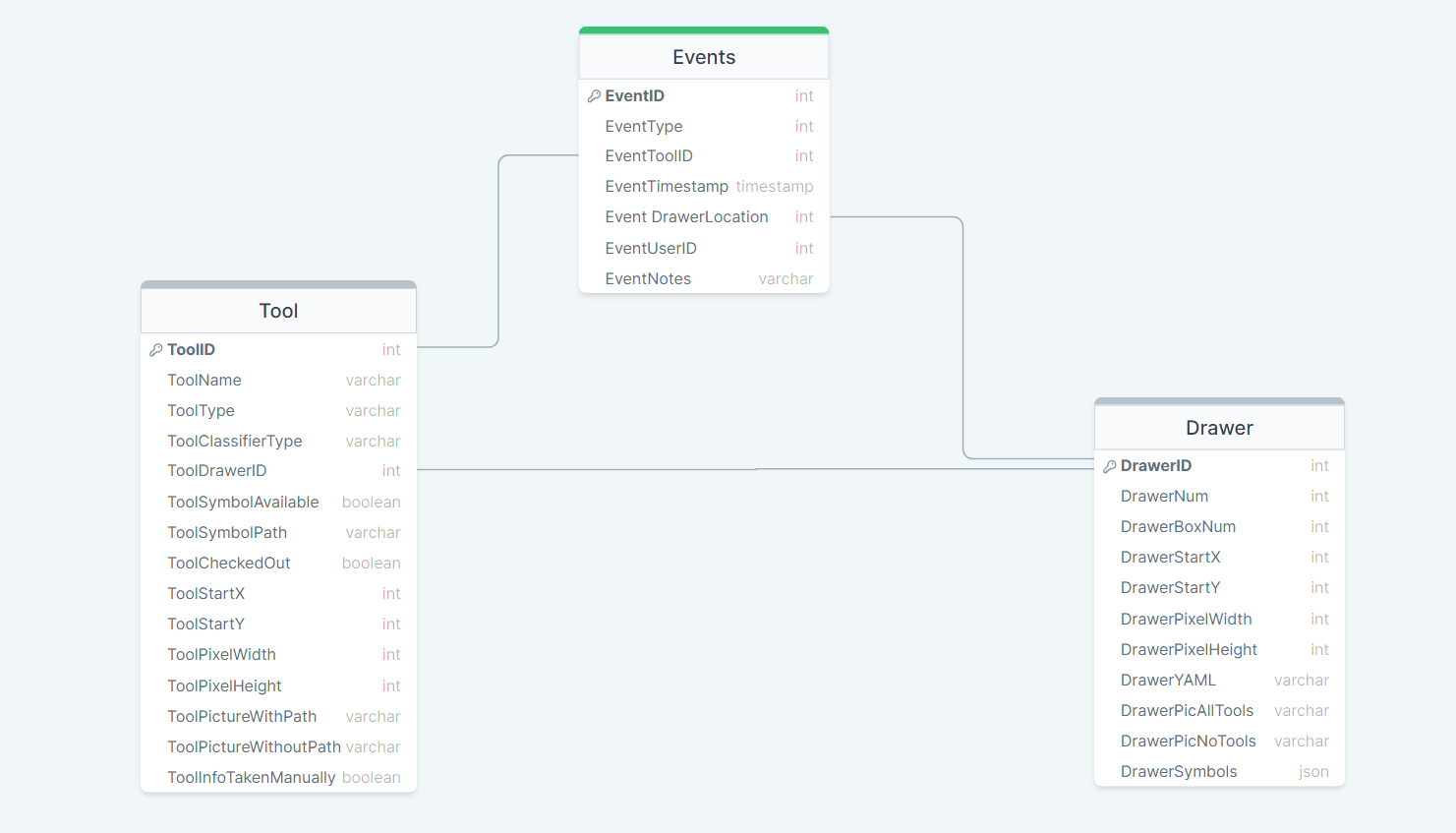
## 3.2 Drawer configuration file

* Name: This is not hardcoded and is up to the client.
* Purpose: Holds configuration values specific to drawer
* Variables:
  + grayscale: 6,7,10 or 11
    - what cv2 color conversion code to use to turn the image into grayscales
  + blur: odd int between 1 and 31
    - how much to blur image
      * used to reduce noise
  + threshtype: 0, 1, 2, 3, 4, 7, 8, 16
    - cv2 threshold type to use to turn picture into just white and black
  + minthreshvalue: int between 0 and 256
    - minimum value for determining if pixel should be black or white
  + minwidth: int between 0 and infinity
    - minimum number of pixels that make up the width of a object
  + minheight: int between 0 and infinity
    - minimum number of pixels that make up the height of an object
  + dincreasewhite: 0 to infinity
    - how many iterations to increase white area in created thresholded (black and white) image
  + increaseblack: 0 to infinity
    - how many iterations to increase black area in created thresholded (black and white) image
  + bufferx: int between 0 and width of picture
    - how many pixels to expand the horizontal start and end of the template by
  + buffery: int between 0 and height of picture
    - how many pixels to expand the vertical start and end of the template by
  + segment: -1 0 1
    - determines what picture to use drawer segmentation on
      * -1 turn off drawer segmentation in automatedtoolbox program
      * 0 used tools in drawer picture for additional script segmentation
      * 1 used no tools in drawer picture for additional script segmentation
  + symbolbuffer: 0 to 1
    - The amount of the tool that must be visible for the program to check for the symbol.

## 3.3 Database Schema

### 3.3.1 Tables

#### Relationship Table



#### Tools Table

|  | Name | Data Type | Description |
| --- | --- | --- | --- |
| P🔑 | ToolID | INT | Unique to each tool |
|  | ToolName | VARCHAR | Name of tool |
|  | ToolType | VARCHAR | Socket/Hammer/etc. |
|  | ToolClassifierType | VARCHAR | What classifier classifies it as |
| F🔑 | ToolDrawerId | INT | DrawerID |
|  | ToolSymbolAvailable | BOOLEAN | If a unique symbol is on the tool |
|  | ToolSymbolPath | VARCHAR | File path to unique symbol image |
|  | ToolCheckedOut | BOOLEAN | If the tool is checked out |
|  | ToolStartX | INT | Starting X pixel |
|  | ToolStartY | INT | Starting Y pixel |
|  | ToolPixelWidth | INT |  |
|  | ToolPixelHeight | INT |  |
|  | ToolPictureWithPath | VARCHAR | File path to image |
|  | ToolPictureWithoutPath | VARCHAR | File path to image |
|  | ToolInfoTakenManually | BOOLEAN | Was tool info taken manually |

#### Drawer Table

|  | Name | Data Type | Description |
| --- | --- | --- | --- |
| P🔑 | DrawerID | INT | Unique to each drawer |
|  | DrawerNum | INT | Which drawer number in toolbox |
|  | DrawerBoxNum | INT | Which toolbox |
|  | DrawerStartX | INT | Starting X pixel |
|  | DrawerStartY | INT | Starting Y pixel |
|  | DrawerPixelWidth | INT |  |
|  | DrawerPixelHeight | INT |  |
|  | DrawerYAML | VARCHAR | URL to drawer specific yaml file |
|  | DrawerPicAllTools | VARCHAR | File path to image |
|  | DrawerPicNoTools | VARCHAR | File path to image |
|  | DrawerSymbols | JSON | List of the drawers 3 symbols file paths and their locations |

#### Events Tabel

|  | Name | Data Type | Description |
| --- | --- | --- | --- |
| P🔑 | EventID | INT | Unique event ID |
|  | EventType | INT | Which event |
| F🔑 | EventToolID | INT | Which tool |
|  | EventTimestamp | TIMESTAMP |  |
| F🔑 | EventDrawerLocation | INT | Which drawer the event happened at |
|  | EventUserID | INT | Which employee |
|  | EventNotes | VARCHAR | Other details about the event |

### 3.3.2 API Endpoints | Custom function call

These functions are designed to fetch and update data within the database. They are located on the API Gateway server and can be triggered using an HTTP GET request.

#### Endpoint: get\_tools\_info | Function: getToolsInfo()

| Input/Output | Name | Data Type | Description |
| --- | --- | --- | --- |
| Input | drawerID | int | Which drawer |
| Output | returns | json | Returns all tool information with corresponding drawerID and boxNum |

#### Endpoint: get\_drawers\_info | Function: getDrawersInfo()

| Input/Output | Name | Data Type | Description |
| --- | --- | --- | --- |
| Input | DrawerBoxNum | int | Which box |
| Output | returns | json | Returns all drawer information with corresponding boxNum |

#### Endpoint: add\_event | Function: addEvent()

| Input/Output | Name | Data Type | Description |
| --- | --- | --- | --- |
| Input | EventType | int | Take tool/return/error etc. |
| Input | EventtoolID | int | Which tool |
| Input | EventTimestamp | timestamp | What time |
| Input | EventDrawerLocation | int | Which drawer |
| Input | EventUserID | int | Who did it |
| Input | EventNotes | varchar | Notes |
| Output | void | void | Create event record in event table |

#### Endpoint: update\_tool | Function: updateToolsInfo()

| Input/Output | Name | Data Type | Description |
| --- | --- | --- | --- |
| Input | ToolcheckedOut | bool | Is the tool checked out now |
| Input | ToolID | int | Which tool |
| Input | ToolDrawerID | int | Which drawer |
| Output | void | void | Updates “checkedOut” column in tool table |

#### Endpoint: add\_drawer | Function: addDrawer()

| Input/Output | Name | Data Type | Description |
| --- | --- | --- | --- |
| Input | DrawerNum | int | See drawer table for descriptions |
| Input | DrawerBoxNum | int |  |
| Input | DrawerStartX | int |  |
| Input | DrawerStartY | int |  |
| Input | DrawerPixelWidth | int |  |
| Input | DrawerPixelHeight | int |  |
| Input | DrawerYAML | varchar |  |
| Input | DrawerPicAllTools | varchar |  |
| Input | DrawerPicNoTools | varchar |  |
| Input | DrawerSymbols | json |  |
| Output | void | void | Add a drawer to drawer table |

#### Endpoint: add\_tool | Function: addTool()

| Input/Output | Name | Data Type | Description |
| --- | --- | --- | --- |
| Input | ToolName | varchar | See tool table for descriptions |
| Input | ToolType | varchar |  |
| Input | ToolClassiferType | varchar |  |
| Input | ToolDrawerID | int |  |
| Input | ToolSymbolAvailable | boolean |  |
| Input | ToolSymolPath | varchar |  |
| Input | ToolCheckedOut | boolean |  |
| Input | ToolStartX | int |  |
| Input | ToolStartY | int |  |
| Input | ToolPixelWidth | int |  |
| Input | ToolPixelHeight | int |  |
| Input | ToolPictureWithPath | varchar |  |
| Intput | ToolPicutreWithoutPath | varchar |  |
| Input | ToolInfoTakenManually | varchar |  |
| Output | void | void | Add tool to tool table |

#### 

## 3.4 Signal Schema

The automated toolbox program expects a start signal and a stop signal to be given to it over a socket connection.These should be json objects. The start signal will have the program start getting footage from the RTSP camera feed. The stop signal will cause the program to stop getting footage from the camera feed.

* Stop Signal
  + fields:
    - stop :
      * type: string
      * Allows developers to include a note for debugging purposes .
    - bool :
      * type : boolean
      * If you want program to stop getting footage it must be set to true
* Start Signal
  + fields:
    - toolbox :
      * type: number
      * This should be the number associated with the toolbox that was logged into in the database.
    - UserID :
      * type: number
      * This should be the number associated with the employee.

# 4.0 User Interface

This project features a minimally designed user interface by intention. The objective is to minimize user interaction for smooth day-to-day operation. Users are not expected to directly interact with the program; rather, they will tap their RFID card on the RFID reader to initiate a trigger, indicating their presence and intention to use the toolbox. The primary interaction with this program is reserved for administrators, who will initiate it through the command line interface.

## 4.1 Command Line

### 4.1.1 Input

* The command line is used to start the program.
  + python automatedtoolbox [-test <path/to/video/file>] [-record]
    - The command may need to use python3 instead depending on the linux distribution.
  + It takes two optional command line arguments
    - - test
      * Argument: filepath to a video file of the drawer
        + Argument is optional, default value is “none”
    - - record
      * Does not take an argument
        + Program will create a video with debug information
* Additional input:
  + In test mode the program will also ask the user to input the toolboxID as user input

### 4.1.2 Output

* The command line will display any runtime errors
  + Errors will also be logged to events database
  + If -test is true, no errors will be logged
* If -test is true, all events will be output to command line instead of being logged in the database
* If -record is true, the program will save a video file showing debug information.

# 5.0 Information Repositories

While SQL Databases are the industry standard for storing large quantities of textual information, it is difficult to store particularly large images, videos, or other binary objects. Since computer vision requires images and videos as input the system will need a way to store these files for easy retrieval.

## 5.1 Multimedia File Server

The selected solution will be to use an Apache web server instance to serve any necessary files. The server will run in a Docker container, and have the following directory structure:

.

└── Apache Root/

├── DrawerInputData/

│ └── Toolboxes/

│ ├── Box1/

│ │ ├── Drawer1/

│ │ │ ├── Drawer1Config.conf

│ │ │ ├── DrawerWithNoTools.jpg

│ │ │ ├── DrawerWithAllTools.jpg

│ │ │ ├── Tool\_1\_1.jpg

│ │ │ ├── Tool\_1\_2.jpg

│ │ │ └── Symbol\_1.jpg

│ │ └── Drawer 2/

│ │ ├── Drawer2Config.conf

│ │ ├── DrawerWithNoTools.jpg

│ │ ├── DrawerWithAllTools.jpg

│ │ ├── Tool\_1\_1.jpg

│ │ ├── Tool\_1\_2.jpg

│ │ └── Symbol\_1.jpg

│ └── Box2/

│ └── Drawer1/

│ ├── Drawer1Config.conf

│ ├── DrawerWithNoTools.jpg

│ ├── DrawerWithAllTools.jpg

│ ├── Tool\_1\_1.jpg

│ ├── Tool\_1\_2.jpg

│ └── Symbol\_1.jpg

└── ClassifierData/

├── classifier.onxx

└── Pics/

├── Hammer/

│ ├── Hammer1.jpg

│ └── Hammer2.jpg

└── Socket/

├── Socket1.jpg

└── Socket2.jpg .

The files will be served over http using Apache’s ability to generate an interactive interface for a file hierarchy. This allows the database to store a url to the desired folder location and is highly scalable, allowing any number of toolbox installations to access the same data. Data can be retrieved using a regular http GET request, and for the purposes of this project we are not placing any security constraints or limitations on access. In the future, it would be ideal for HiLine to create a login system that would authorize each toolbox installation as well as database administrators to view the files, and reject connections from anyone else. Changing the connection method to https would also provide better security and prevent eavesdropping on what files are traversing the network, although this is nontrivial to implement and would result in a less complete product being delivered.

1. credit to [Naofumi](https://stackoverflow.com/users/2079580/naofumi) for their answer to this question :<https://stackoverflow.com/questions/16702966/rotate-image-and-crop-out-black-borders> [↑](#footnote-ref-0)
2. Credit to the maker of this tutorial <https://www.pyimagesearch.com/2017/01/02/rotate-images-correctly-with-opencv-and-python/> [↑](#footnote-ref-1)
3. Credit to [coproc](https://stackoverflow.com/users/1725562/coproc) and [Sigurdur Mar Valsson](https://stackoverflow.com/users/14646718/sigurdur-mar-valsson) for their answer on this

   <https://stackoverflow.com/questions/16702966/rotate-image-and-crop-out-black-borders> [↑](#footnote-ref-2)