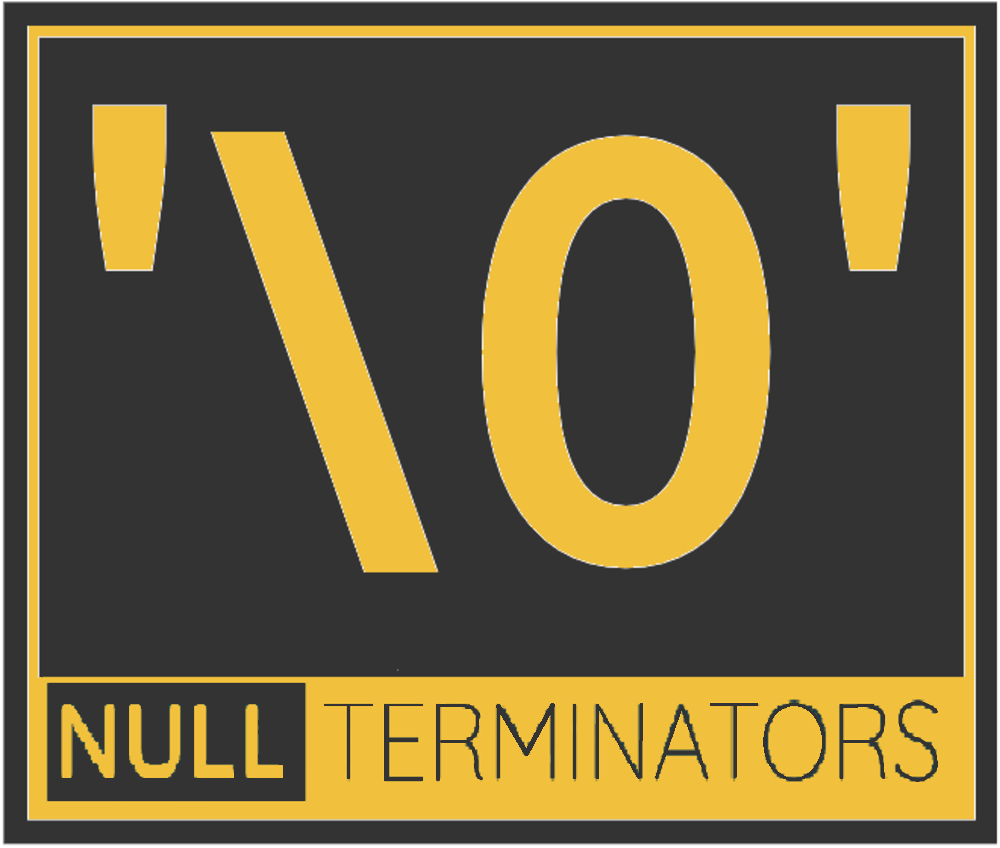
ATICS -

Automated Toolbox Inventory Control System

User Guide

Null Terminators

Version: 1.1





Senior Project Requirement Specifications

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 (Project Manager)

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**

[**Revision History 4**](#_tul8t8cqyjgg)

[**Automated toolbox program installation 5**](#_3ms9ill83rk7)

[**Servers installation 5**](#_r260pxjf62zo)

[**Camera Setup 6**](#_g4nvrs39j1to)

[**Additionalscript guide 8**](#_3nt15bpbsceb)

[**How to use: 8**](#_boeahejoojkd)

[**How to manually find pixel locations: 10**](#_w49qq3mgntq8)

[Instructions: 10](#_agfkot2m0u92)

[Example: 11](#_s377brj6f0s3)

[**Running the program 17**](#_ikrcoj1oah3e)

[**Increasing Performance 18**](#_agvsawlvkos8)

[Software 18](#_y4sm7gpnawoa)

[Hardware 18](#_ivt72njd8ngs)

# 

# 

# 

# Revision History

1.0 - 4/30/2024 First Published Version auth. Null Terminators

1.1 - 5/1/2024 Add performance section auth. Caleb Thomas

# 

# Automated toolbox program installation

From an Ubuntu Linux terminal, run the following commands:

1. sudo apt install git
2. git clone <https://github.com/CaitlynCagaaa/ATICS.git> or the hiline version of the repository
3. sudo apt install bash
4. cd < *ATICS* or name of git repository>
5. bash ./instalreq.bash
   1. if file does not work try doing sudo bash ./instalreq.bash

Optional: If you want to be able to grab frames on this server you would need to install a video editing software capable of grabbing frames

# Servers installation

1. Install docker using the appropriate terminal commands for the specific version of Linux Docker is being installed on as written here: <https://docs.docker.com/engine/install/>. Alternatively, install docker desktop using these instructions: <https://www.docker.com/products/docker-desktop/>

If using a recent version of Ubuntu, the following installation method may be used:

* 1. Run the following in an Ubuntu terminal:
     1. sudo apt-get install ca-certificates curl
     2. sudo install -m 0755 -d /etc/apt/keyrings
     3. sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
     4. sudo chmod a+r /etc/apt/keyrings/docker.asc
     5. sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
        1. if this doesn't work just try sudo apt-get install docker

Returning to the terminal, run the following commands:

1. sudo apt install git
2. git clone <https://github.com/CaitlynCagaaa/ATICS.git>
   1. If HiLine customizes the software and uses their own build of ATICS, then clone that repository instead of the Null Terminators’s release
3. cd into ACTIS\_Docker\_V2 folder
4. docker compose up
   1. if this returns an error, try running: sudo docker compose up

# Camera Setup

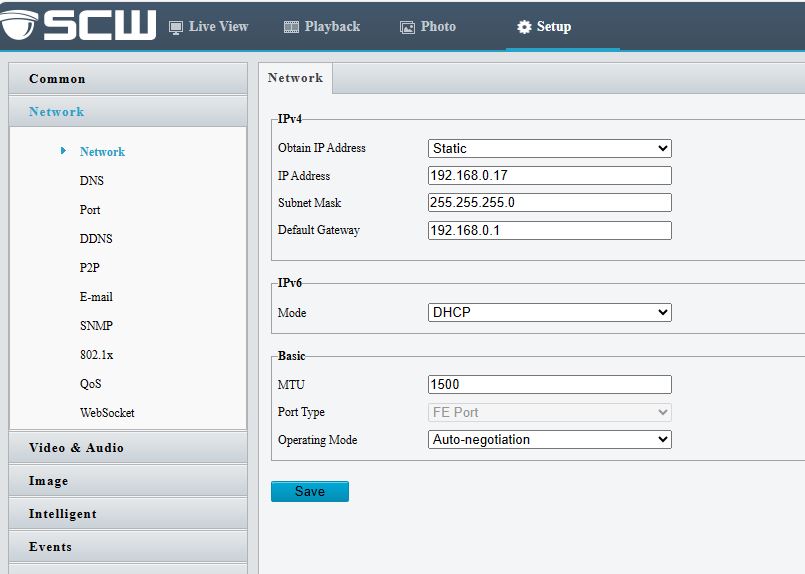
This section assumes the SCW Deputy 4.0 Security camera is being set up. If a different camera make or model is used then these instructions may not specifically apply.

In order to log into the camera’s web console, you will need to use either Microsoft Edge running in Internet Explorer (legacy mode) or Internet Explorer itself. The web console relies on legacy Microsoft-only features which are only properly supported by these products.

WARNING: When setting up the camera you will lose the ability to access the internet from your computer!

To set up the camera, look at the documentation in the box until you find a “default IP” address listed. In your operating system, set your IP address to another IP on the same network, and plug an Ethernet cable between the camera and your computer.

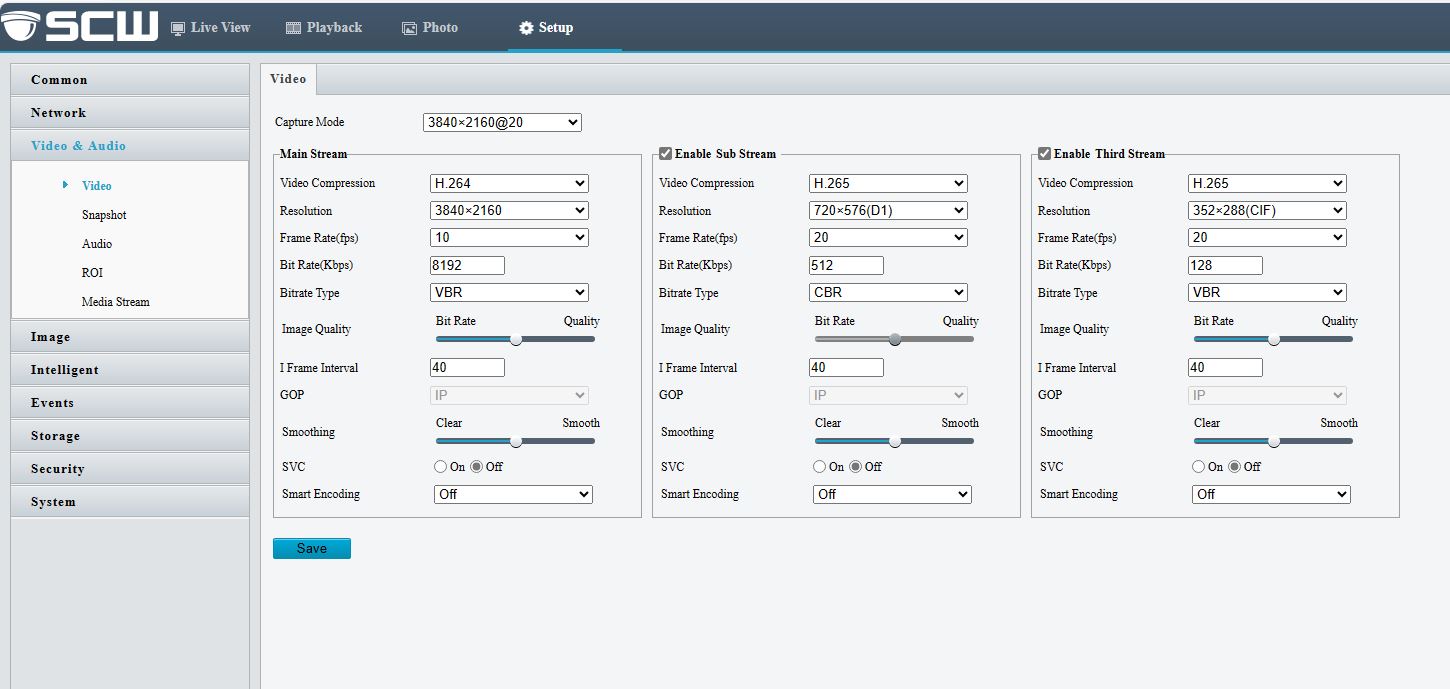
Choose an IP address, and navigate to the setup tab, then click “network” and change the IP address, subnet mask, and default gateway to the desired IP and the network configuration of your local network. Then click “save”



After making this change the camera will reboot, and you’ll need to undo the changes made in step 1 to connect your computer back to the LAN.

NOTE: The camera will not be accessible over the network until the computer is switched back to the LAN.

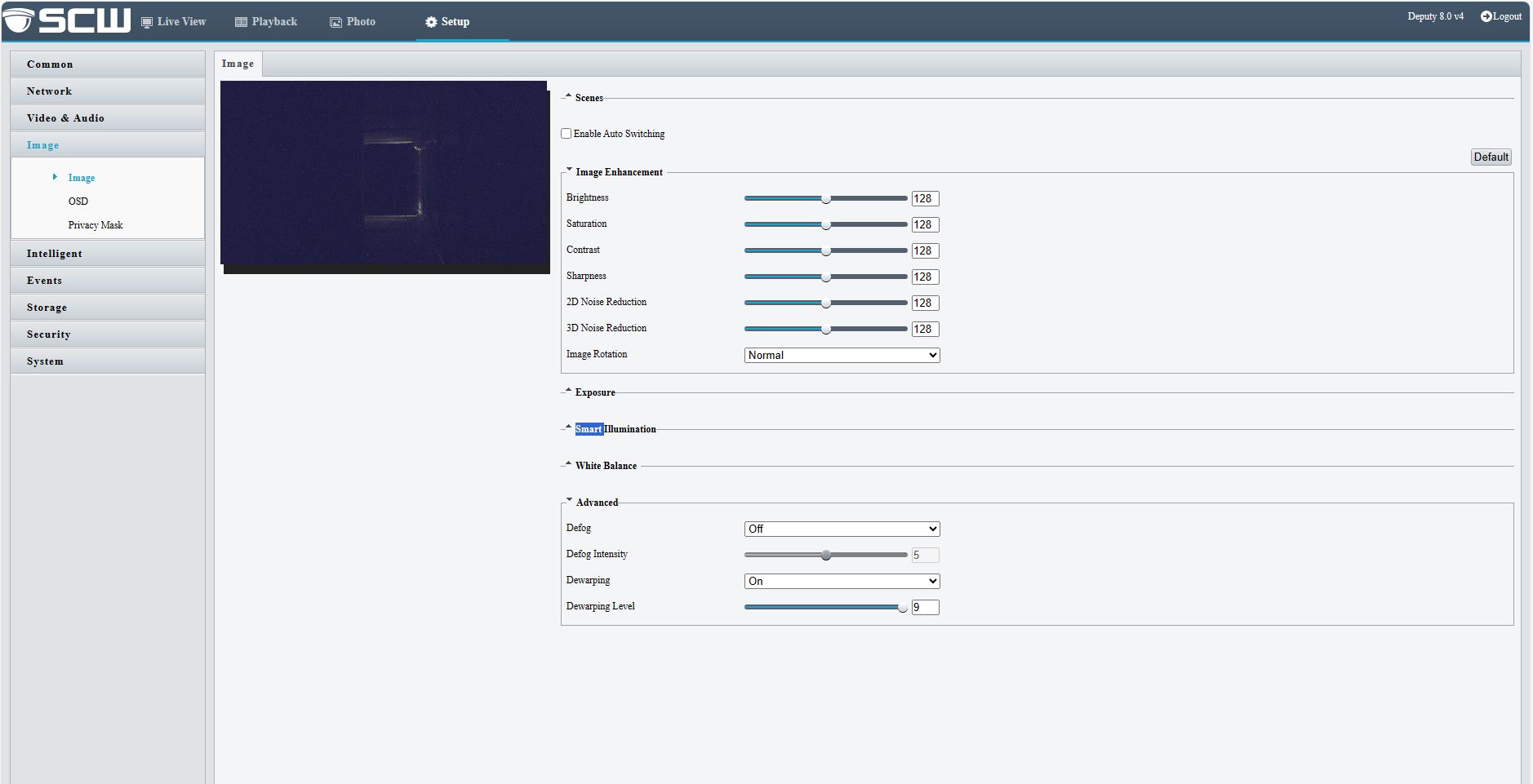
Once the network is configured as desired, the next step is to set up the image settings

Under “Video and Audio” -> “Video” change the following settings:

Video compression should be H.264

Resolution should be set to 3840x2160

Frame Rate should be set to 10 fpts

Under “Image” settings, change the following:

Advanced -> Dewarping should be 9 (the max value)

Adjust smart illumination and white balance until the lighting in the picture is bright enough to show detail on the tools, but not so bright that the image appears washed out

# Additionalscript guide

The additional script is meant to help gather preliminary information that the automatedtoolbox program needs. This script requires a human to double check its work, and may require more or less assistance depending on the drawer type. The script cannot automatically detect the drawer symbols, so that will need to be added to the output manually, although the script does significantly reduce the manual effort required to prepare a new drawer. The output should include a folder called “drawer” that contains the template pictures, other supplementary pictures, tools.json, and drawer.json.

## How to use:

1. Take video using the IP camera in the position it would be during use of the automatedtoolbox program. You can also use a frame with the drawer devoid of tools and a frame with all tools in their proper places within the drawer. By taking the video you can also double check the information placed in the database using the -test mode of the automated toolbox program.
   1. The steps to follow to get the necessary information in the video are:
      1. Open the drawer all the way with either all tools checked in or all tools checked out .
      2. Wait one second without your hands in or on the drawer.
      3. Check all tools in or check all tools out.
      4. Wait one second without your hands in or on the drawer.
      5. Close the drawer completely.
   2. During the above steps make sure that you are not in between the drawer and the camera.
2. Take two frames from the video - one with all tools checked in, and one with all tools checked out. Make sure all tools are visible during these frames and all drawer symbols are visible( ie, no hands in or on the drawer).
   1. These must not be screenshots. Screenshots will be a different resolution than the actual frames from the video feed, thus the pixel locations will not be correct.
3. Use the two frames as command line input.
   1. **python additonalscript.py <file location of one frame> <filelocation of other frame> [optional]<fiel location of configuration file>**
      * 1. **Depending on linux version may need to use python3 instead of python**
      1. Ex:python additonalscript.py "drawer330\_0\_0\_empty.jpg" "drawer330\_0\_0\_full.jpg"
         1. The first frame given to the program (in the example “drawer330\_0\_0\_empty.jpg”) , will be the frame that is thresholded and used to find where the drawer and tools are.
            1. If this is the frame with no tools set the 'segment' variable in the configuration file to 1 before running. If this frame is the frame with no tools, set the 'segment' variable in the configuration file to 0 before running.

This just makes it so the correct picture is put in the correct location to the json if you forget just go through the drawer and tool jsons and swap the picture location in ‘ToolPictureWithPath’ and the picture location in ‘ToolPictureWithoutPath’ and vice versa.

* + - 1. The second frame will have the same locations found in the first frame cropped into individual pictures.
      2. The optional third argument is the file location of the drawer configuration file, if you do not give it one it will assume the file location is 'conf.yaml'.

1. If you are happy with the work that the additional script has done, do nothing for this step. If you are not happy with the work that the additional script has done, change the configuration file and try step 3 again.
2. Modify drawer.json and tools.json to include all the necessary information for the automatedtoolbox program.
   1. Add drawer symbol locations and file location of pictures to the drawer json under the “DrawerSymbols" list into “picall” “x” “y” “w” “h” for all three drawer symbols.
   2. Double check the drawer location ( look at the drawer\_1 .jpg and drawer\_2.jpg and confirm that it is correct, it should be just the drawer not the entire frame). If incorrect, crop the picture again and modify the drawer location to match.
      1. If the drawer exposes the inner front of the drawer like below, you want that to be included in the pixel locations of the drawer, and to ensure that bufferx in the Global\_Config.yaml is at a minimum the width of that.
         1. 
   3. Double check the tool locations and make sure all tools are in the file. If there are missing tools, add them to the json file, and crop pictures from the two frames.
      1. It is recommended that the “ToolInfoTakenManually” variable is set to true for each tool that is edited manually. This helps flag which tools to review in case there are issues in the future caused by miscalculated pixel values or mismatches between the pictures of the tool checked in and the tool checked out.
3. If you want the drawer to not look for extra tools, change the segment to be -1
4. Put the pictures and drawer configuration file on the webserverAPI. Then change the file locations of the tools.json and drawer.json picture files, as well as the drawer configuration file location to match the file location denoted in the global configuration file for the automatedtoolbox program.

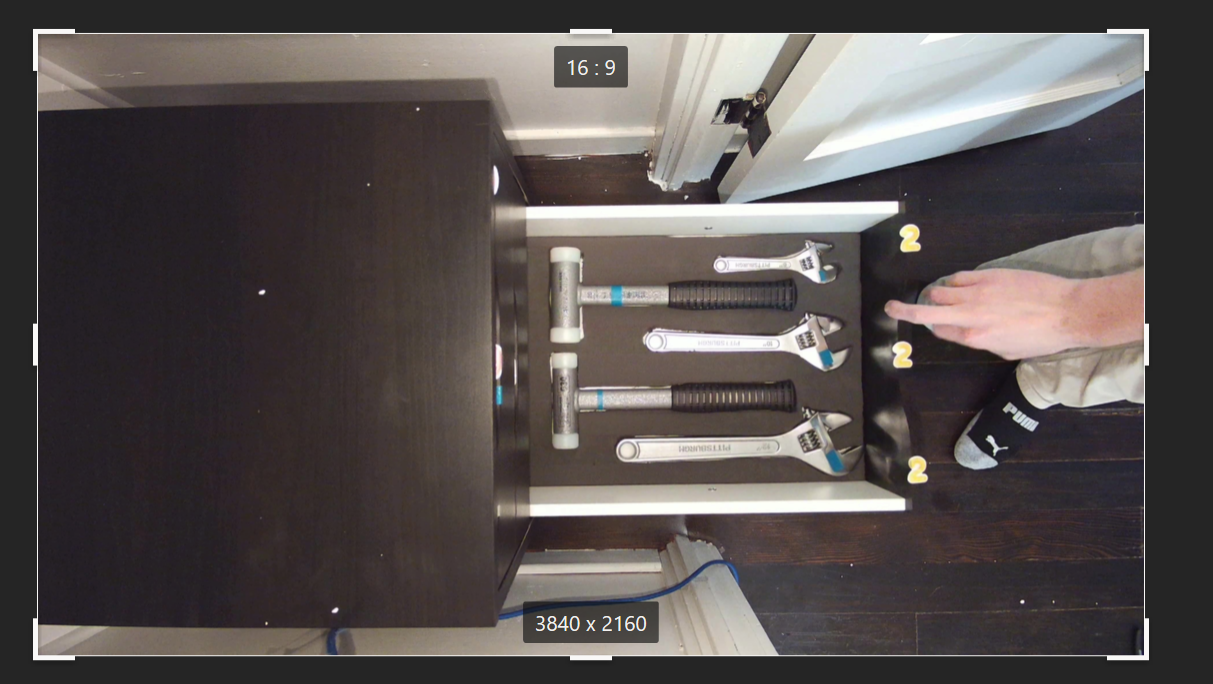
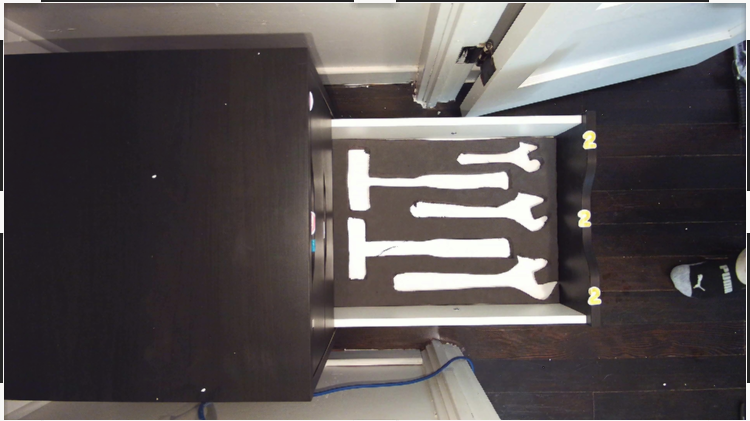
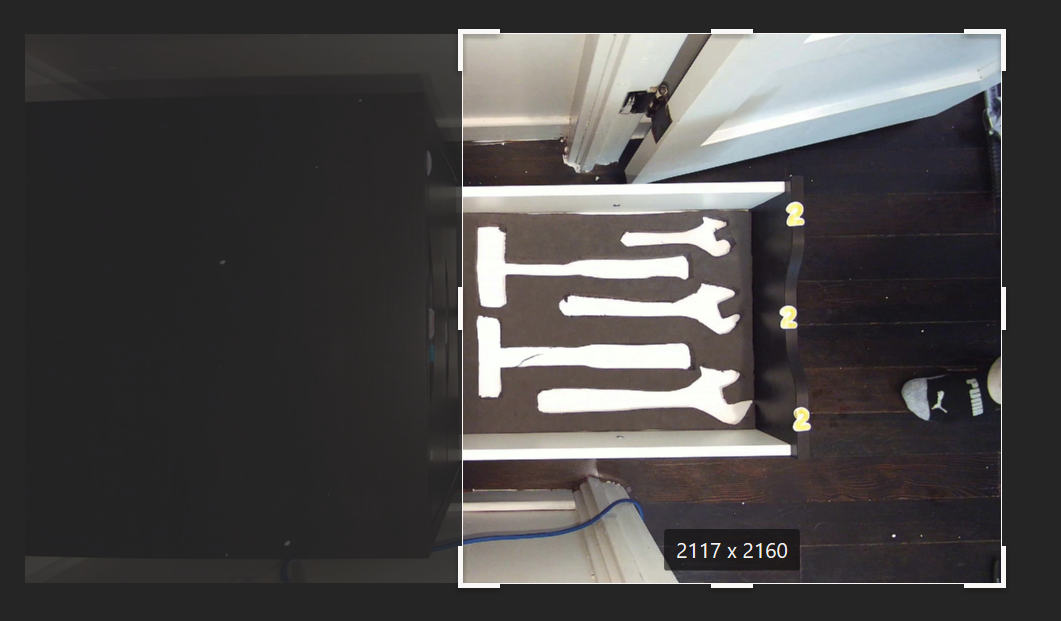
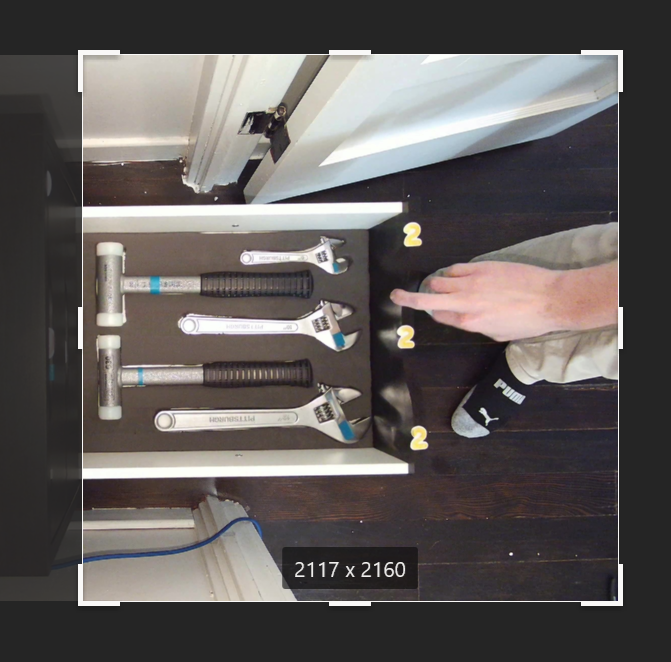
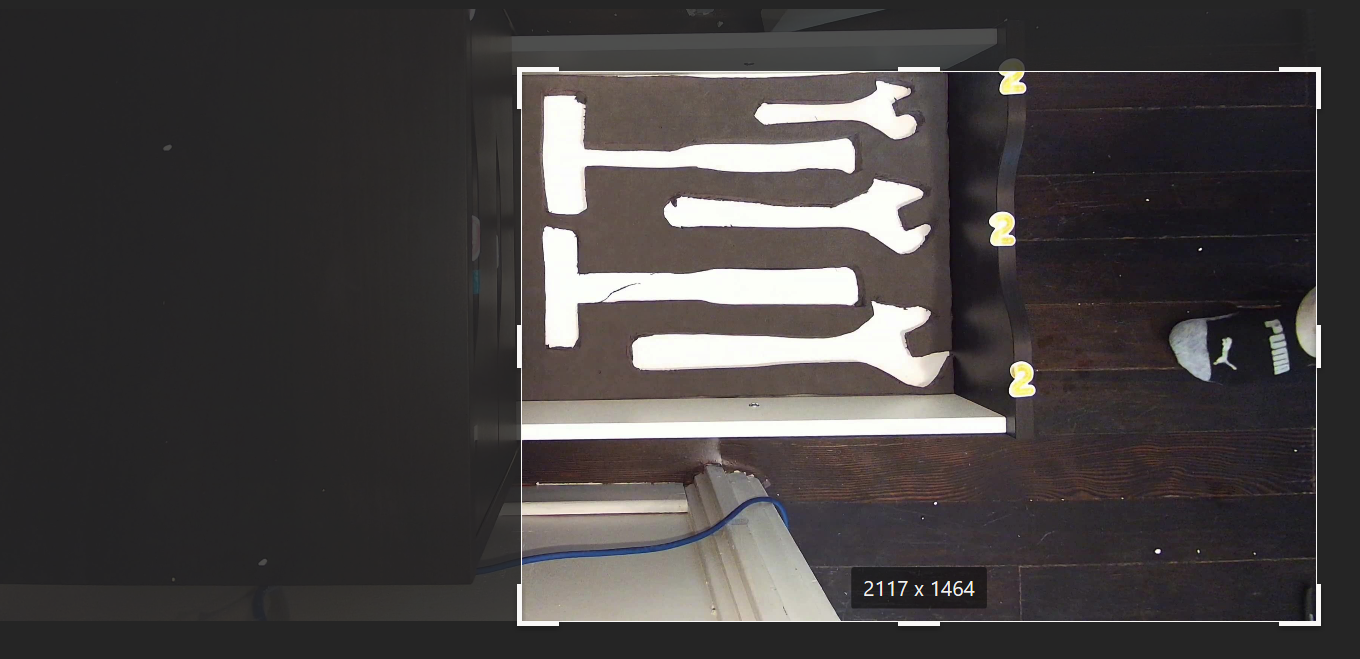
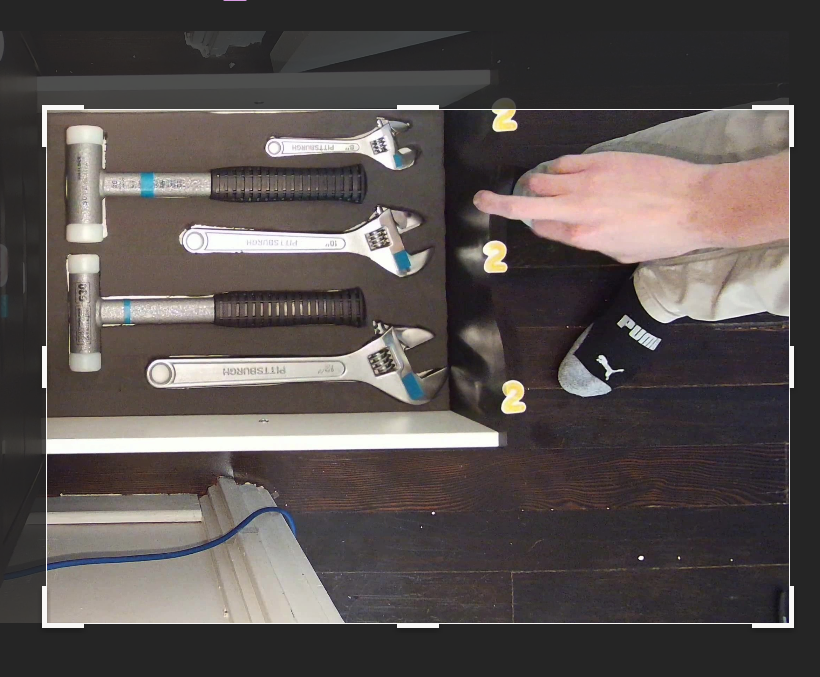
## How to manually find pixel locations:

### Instructions:

1. Open image editing software with the frame with no tools, and frame with all tools
2. Note the size of the frame
3. Move the start of the pictures (the leftmost part) from the x side to the start of the tool.
   1. Subtract the current width from the size of the frame.
      1. That is the starting X of the tool.
4. Move the start of the pictures (the uppermost part) from the y side to the start of the tool.
   1. Subtract the current height from the size of the frame.
      1. That is the starting Y of the tool.
5. Move the other sides of the pictures (the rightmost and downmost sides) until only the tool is visible.
   1. The new width and height are the tools W and H respectively.
6. Crop the pictures. Put the location of the cropped pictures in the tools.json, or drawer.json depending on what it is.
7. Try to have the x,y,w,h values of the 2 pictures be as close as possible, and try not to have the part of any other tool in the image.
8. To get symbols, crop the symbol from the frame and place the file location in the tools.json. Exact x,y,w, and h values are not needed.

### Example:

In this example we are trying to get the tool locations for the drawer.

1. Note the size of the frame: Width = 3840, Height = 2160
   1. 
   2. 
2. 
   1. 3840 -2117 = 1723. So DrawerStartX is 1723
3. 
   1. 2160 - 1464 = 696. So DrawerStartY is 696.
4. 
   1. So the DrawerPixelWidth is 1141 and the DrawerPixelHeight is 864.

# Running the program

* Type the following command into the terminal to start the program.
  + python automatedtoolbox [-test <path/to/video/file>] [-record]
    - Some Linux distributions will require you to replace “python” with “python3”
  + ATICS takes two command line arguments
    - - test
      * Argument: filepath to a video file of the drawer
        + Argument is optional, default value is “none”
      * Used to test the information you have given to the database, or changes made to the program.
    - - record
      * Does not take an argument
        + Program will create a video with debug information
        + Argument is optional
      * Used to debug issues with the database inputs, changes made to the program, or if you want additional info for the login events.
* Additional input:
  + In test mode the program will also ask the user to input the toolboxID as user input

# Increasing Performance

## Software

The speed of the application could be increased in the future through the use of multithreading or multiprocessing. Multithreading refers to a method of multitasking which breaks program execution into threads which are then executed by the operating system as CPU cycles are available. Multiprocessing refers to a computer with multiple processors - either multiple discrete CPUs or a single CPU with multiple cores. Each of these cores can run a single thread at a time, which allows portions of the program to run in parallel. This allows the operating system to schedule as many resources as are available to ensure ATICS is running at its best performance. Several open source Python libraries exist which could be leveraged to implement these concepts .

The recommended places in the program to implement this are:

* Retrieving the footage from the camera and processing the frames .
* The loop which determines which drawer is open in the toolbox in find\_drawer.
* The loop which finds the drawer symbols in is\_open.
* The loop over the list of tools in update\_tools\_for\_frame.
* The loop within drawtemp which conducts template matching and rotating templates.

## Hardware

Once the software has been adjusted to allow for multithreading and multiprocessing, ATICS would benefit from running on high performing hardware. For CPUs, core count should be prioritized over raw speed in order to achieve best results, and we recommend looking into ways to offload work onto a graphics card for even better performance. Graphics cards are designed to accelerate the very math intensive process of rendering images on computer screens, which means they have a specialized processor with many cores. Because ATICS uses similar math to analyze frames, if the correct multiprocessing library is chosen then a graphics card could be employed to do the bulk of the work for ATICS. This would produce the greatest decrease in runtime, although at the cost of hardware. A modern , high performance graphics card would be needed for best results - likely running $1,000 - $1,500 at time of writing. CPU bound multiprocessing would be far cheaper, as only a Xeon or Core i9 (or AMD equivalent) would be needed - again - the more cores the better.