1. Intro

This is a 2D real time object recognition system. Since it's 2D and not taking the depth and color, the system works best when the objects are somewhat distinguished in shapes.

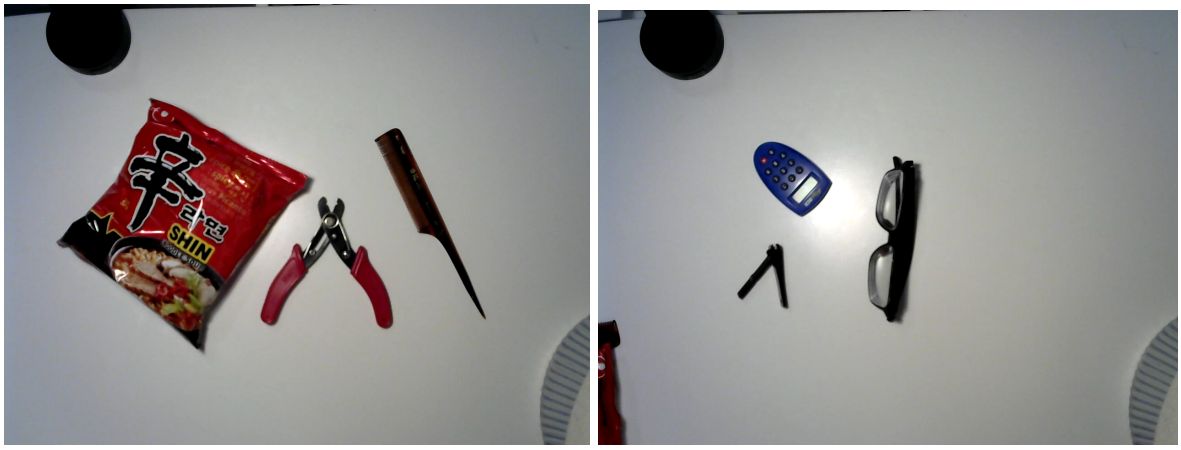
The process starts with separating a frame into background and foreground color depending on a constant threshold value. Then clean up noises and exclude regions that are not in the center of the image.

We then compute features such as fill percentage, height weight ratio and moments to get 8-9 features of the object. We can then start classifying using the nearest neighbor algorithm.

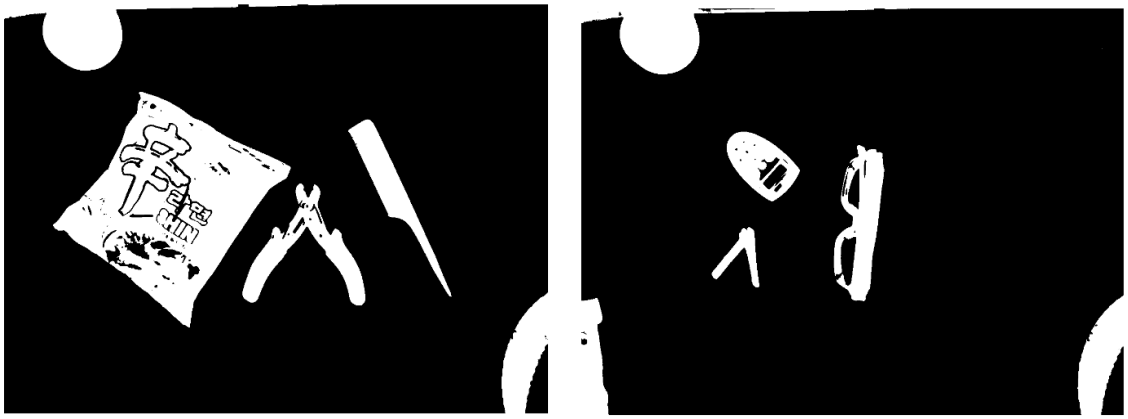
Task 1. Threshold the input video

Thresholding is a technique used in image processing and computer vision, including object recognition and classification, for separating regions of interest from the background. It is particularly useful when dealing with images that have a high contrast between the object and the background.

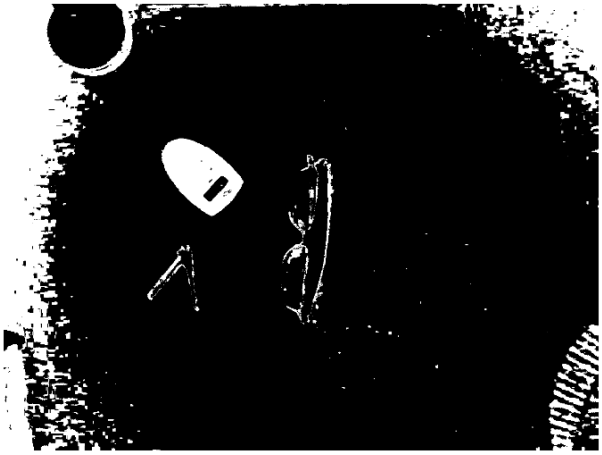
original images:



Thresholded images (RGB):



Thresholded images (HSV) using rgb to hsv own conversion:



Task 2 . Clean up the binary image

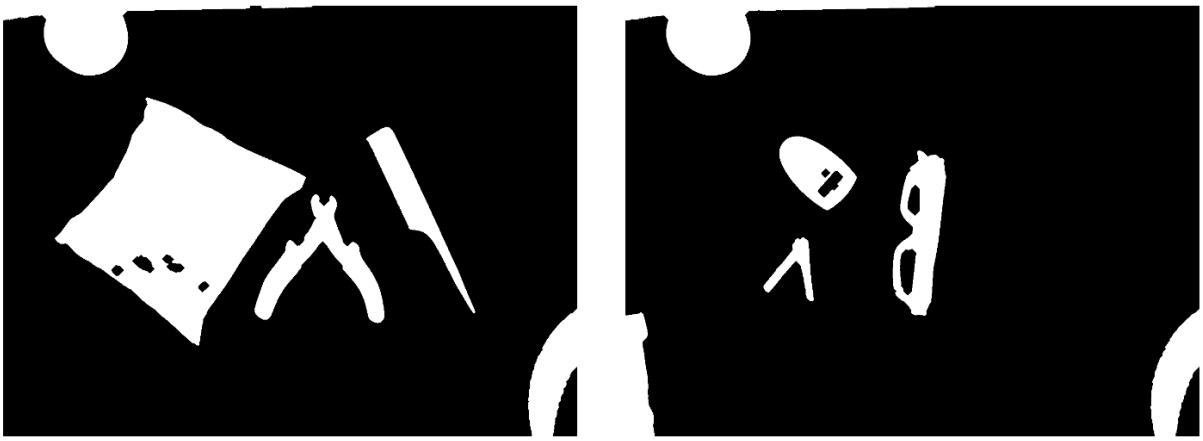
Using open cv morphologyEx (MORPH_CLOSE and MORPH_OPEN)

Clean the previous thresholded image using some type of morphological filtering. (growing/shrinking) to try to solve the issues. I use first closing then opening for the following reasons:

Closing (MORPH_CLOSE): Closing is a combination of dilation followed by erosion. It helps fill small holes, gaps, or dark regions like what we see on the graphic of the noodle package. We will close those small openings.

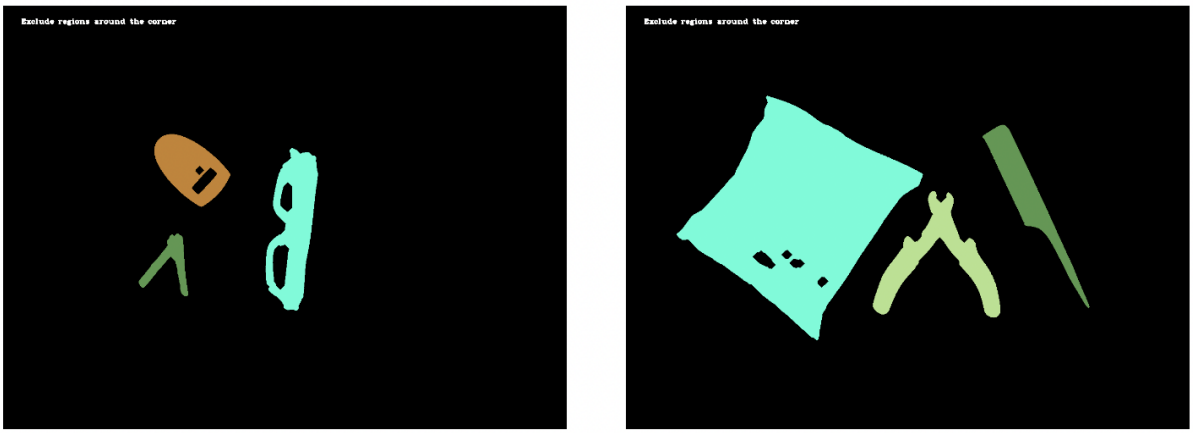
Opening (MORPH_OPEN): Opening is a combination of erosion followed by dilation. It helps remove small isolated noise or bright regions while preserving the overall shape. Applying

opening after closing can further clean up the image by eliminating noise and bright regions around the shape. Below is the result:



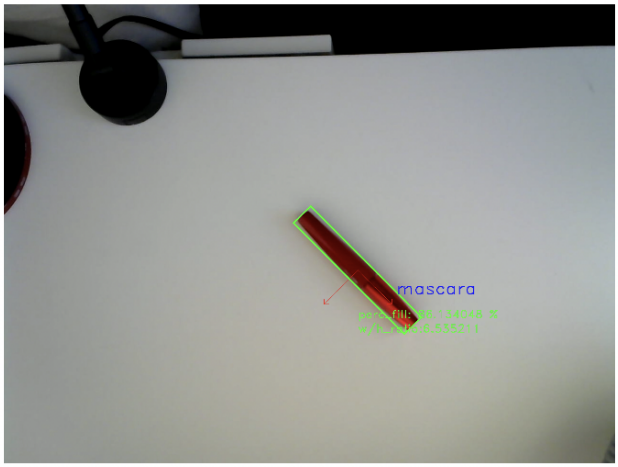
Task 3 . Segment the image into regions

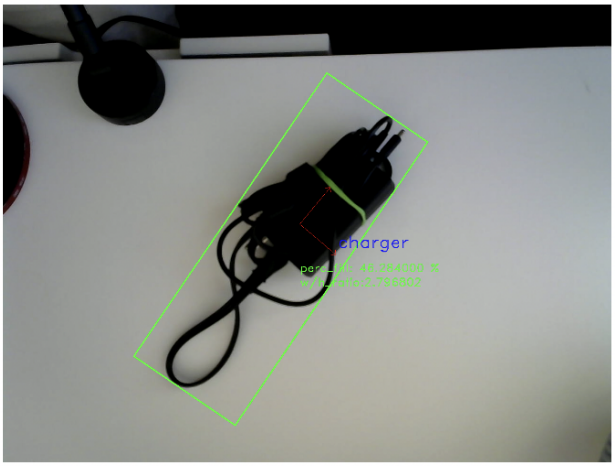
Using openCV connectedComponentsWIthStats we can obtain various statistics for each connected component, such as the area, centroid, and bounding box. By analyzing these statistics, you can further refine the image by removing unwanted regions. We will use this to remove all the regions around the corner like the vase on the bottom right and the black blob on the top left hand corner



Task 4 . Compute Features for each major region

We will extract a region of interest from an input image, and compute several features (Hu moments, percentage of area filled, and height-to-width ratio) to describe the region, and display the features on the output image.





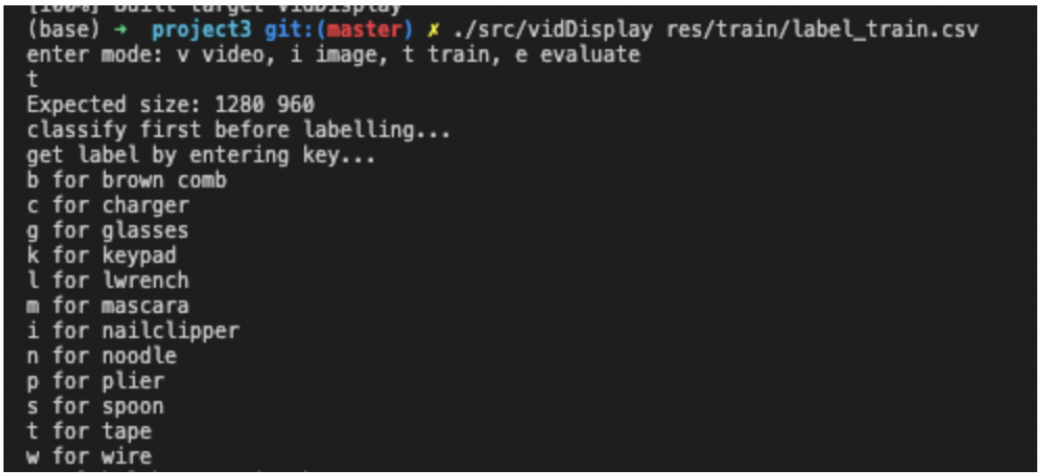
Task 5. Collect training data

Users can enter a different mode. To collect training data, enter 't'. Then enter 'a' to start labeling the target object. It will show a target object without any label on. the console will prompt the user for a key. Since we are working with a known number of objects, the user can just press the key while on the video so they don't need to type the object name.Once it's labeled, the info of vectors and image name will be appended to the csv file.
Below is a screenshot that shows the training images name, object names, and its feature vector

project3 > res > owntrial > labeltrial.csv

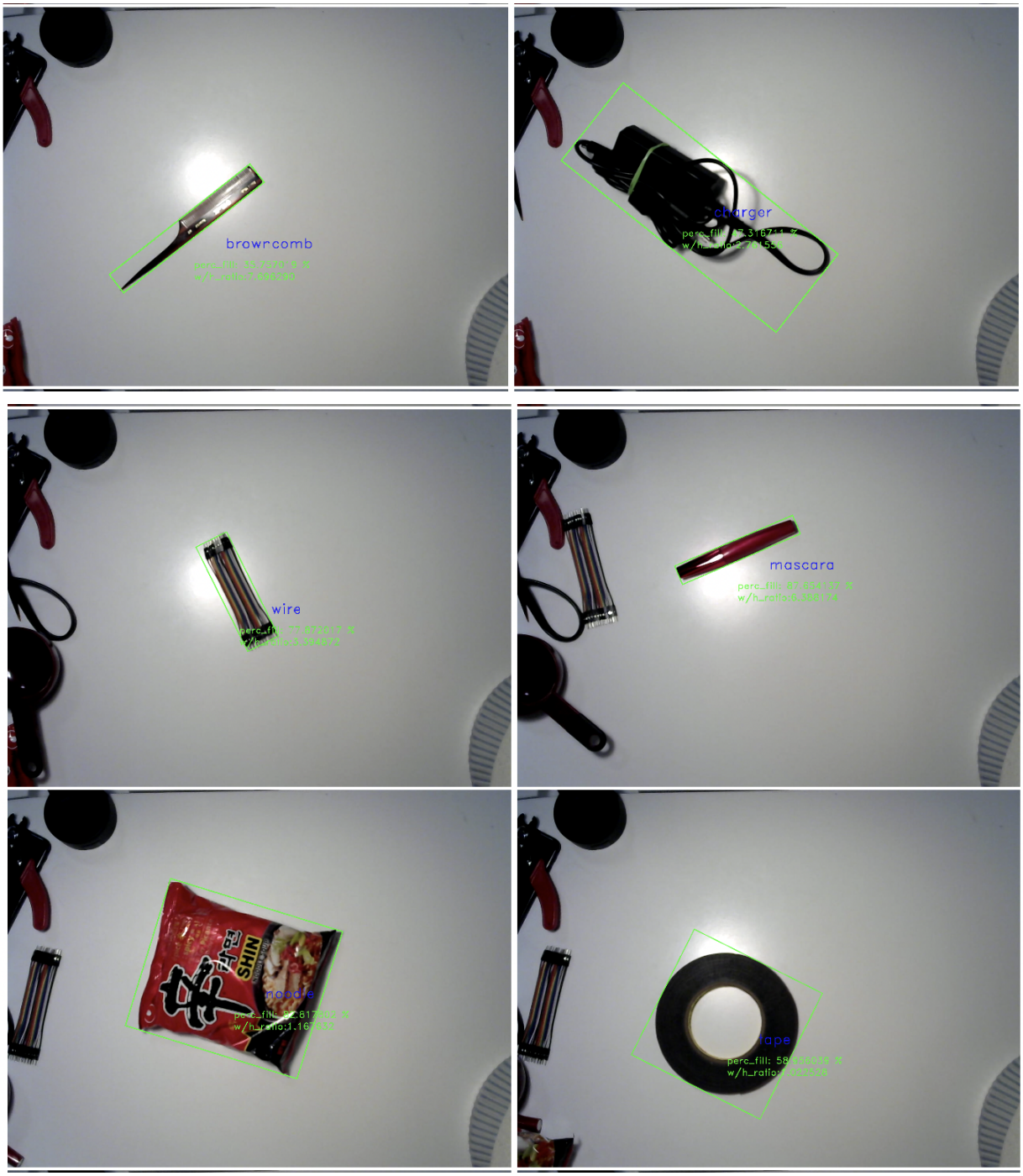
```
own0.png,glasses,0.5407,1.2475,3.4440,3.8011,7.4247,4.4308,8.5867,61.9910,3.2409
own1.png,glasses,0.4873,1.1005,3.1737,3.5235,6.8769,4.1038,7.7011,62.8695,0.2699
own2.png,glasses,0.4997,1.1349,3.1729,3.5293,6.8850,4.1204,7.7214,63.7535,0.2766
own3.png,glasses,0.4844,1.1112,3.3333,3.7354,7.3115,4.4575,7.6479,57.7354,3.3623
own4.png,lwrench,-0.0655,0.1979,0.0066,0.9384,-1.4455,-1.0375,-1.8275,19.1007,2.0986
own5.png,lwrench,-0.0877,0.1684,-0.1739,0.4959,0.9708,1.2363,0.7152,18.4478,0.4653
own6.png,lwrench,-0.0377,0.2566,0.0469,0.9217,-1.7008,-1.3599,1.4705,19.8873,0.4725
own7.png,lwrench,-0.0386,0.2631,0.0164,0.8362,-2.0573,-1.5555,-1.2681,19.5868,2.1078
own8.png,mascara,0.2248,0.4824,3.6081,3.6678,7.3058,3.9093,-9.8975,85.4184,0.1506
own9.png,mascara,0.3413,0.7408,4.0421,4.2089,8.3350,4.5965,-9.6215,87.7299,5.1944
own10.png,mascara,0.2765,0.5949,3.6045,3.7427,7.4166,4.0501,8.8565,85.7798,0.1640
own11.png,mascara,0.3328,0.7214,4.3518,4.9796,9.8007,5.9229,9.7909,87.9576,5.3378
own12.png,noodle,0.7485,2.5118,4.8404,5.9043,12.0323,-7.2988,-11.2834,85.4856,0.8210
own13.png,noodle,0.7683,3.2813,4.6687,6.2803,-12.7883,7.9307,-11.7567,87.7259,0.8912
own14.png,noodle,0.7601,2.7931,4.7174,5.4816,10.7690,-7.7019,10.6998,83.6938,0.8305
own15.png,noodle,0.7659,3.0617,4.5456,6.1305,12.0821,-8.0408,11.4818,84.7106,1.1395
own16.png,plier,0.3543,3.9312,1.1468,2.4207,-4.2050,4.5351,5.5076,31.2070,0.8580
own17.png,plier,0.3964,4.2506,1.2901,2.5770,-4.5424,-4.7031,-4.9427,34.3395,1.1209
own18.png,plier,0.3803,3.5558,1.2398,2.5279,-4.4586,-4.6024,-4.7679,32.8334,0.8881
own19.png,plier,0.3835,4.2741,1.2330,2.4948,-4.4625,-7.0686,4.5688,33.1823,1.1116
own20.png,wire,0.3946,0.8857,3.3749,4.1243,8.3192,5.2216,-7.9038,68.2825,3.0333
own21.png,wire,0.4470,1.0164,3.8738,4.6555,9.9953,-6.3939,-8.9217,71.9700,2.8180
own22.png,wire,0.4400,1.0007,4.1908,5.0026,-9.8383,-5.6384,9.6871,74.1137,0.3417
own23.png,wire,0.4692,1.0763,3.9276,4.4986,8.9810,5.4723,-8.7860,73.6132,2.7319
```

These are all the types of objects we will be training on:

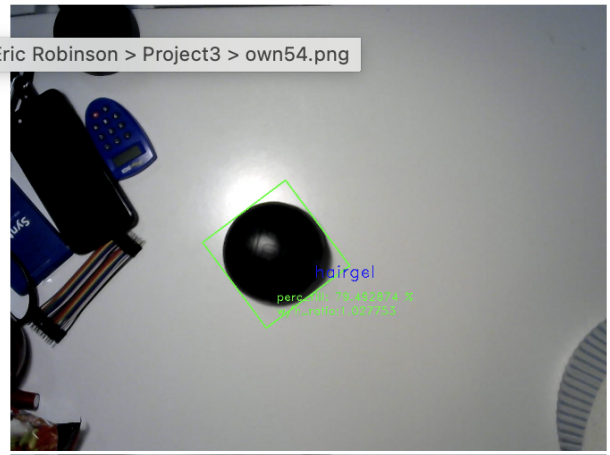


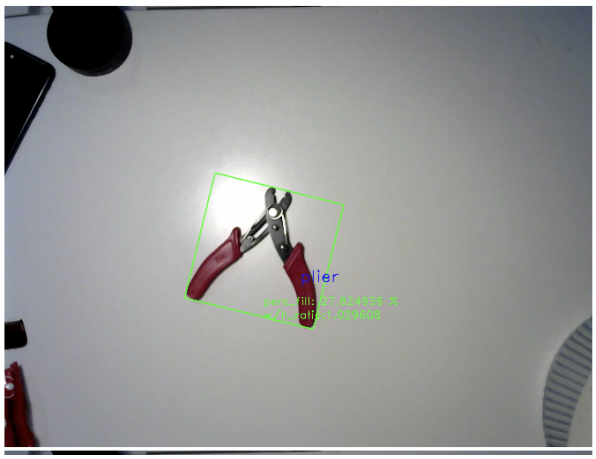
Task 6. Classify new images

We will use a scaled Euclidean distance metric [(x_1 - x_2) / stdev_x] to classify a new feature vector. We can see the results below


browrcomb
charger
wire
mascara
noodle
tape

Eric Robinson > Project3 > own54.png





Task 7 and 8. Implement a KNN classifier and evaluate the performance

KNN fares slightly better than Euclidean distance. It only made 1 wrong prediction whether or not I included the threshold for unknown objects. Below is the confusion matrix result of our of KNN

x	browncomb	charger	glasses	hairgel	keypad	lwrench	mascara	nailclipper	noodle	plier	spoon	tape	wire
browncomb	4	0	0	0	0	0	0	0	0	0	0	0	0
charger	0	4	0	0	0	0	0	0	0	0	0	0	0
glasses	0	0	4	0	0	0	0	0	0	0	0	0	0
hairgel	0	0	0	3	0	0	0	0	0	0	0	0	0
keypad	0	0	0	0	4	0	0	0	0	0	0	0	0
lwrench	0	0	0	0	0	4	0	0	0	0	0	0	0
mascara	0	0	0	0	0	0	4	0	0	0	0	0	0
nailclipper	0	0	0	0	0	0	0	4	0	0	0	0	0
noodle	0	0	0	1	0	0	0	0	4	0	0	0	0
plier	0	0	0	0	0	0	0	0	0	4	0	0	0
spoon	0	0	0	0	0	0	0	0	0	0	4	0	0
tape	0	0	0	0	0	0	0	0	0	0	0	4	0
wire	0	0	0	0	0	0	0	0	0	0	0	0	4