Vision

Tracy Chou, Chin Lung Fong, Shiwei Song {TYCHOU, CLFONG, SHIWEIS}@STANFORD.EDU

Abstract

Our project uses several classifiers to identify each of five types of objects in frames of a test video. The baseline program constructed decision trees based on Haar features for each object type. As one extension, we incorporated different features for image windows, including features based on texture, Hough transforms, and Fourier transforms. We also replaced two of the Haar decision trees, one with boosted decision stumps, and another with a logistic regression classifier. For a single window, we combine the weighted output of the five classifiers with a decision tree strategy, and across multiple windows we suppress overlapping detections. Lastly, we use optical flow to compute the detection windows for every other frame, saving computational time.

1. Overview

Our program is structured as a collection of one-against-many classifiers, the output of which is combined to produce the final classification of an image window.

There are five types of objects that we detect – mugs, staplers, keyboards, clocks, and scissors – so we have five separate classifiers. For mugs, staplers, and clocks, we use decision trees on Haar features; for mugs and clocks we additionally use circles detected by the Hough transform as a feature. For keyboards, we use logistic regression (essentially a decision stump) on Fourier transform features. For scissors, we use boosted decision stumps on texture features.

To combine the output of the five classifiers on a single window, we first consider the output for the keyboards and scissors detectors, since both are fairly accurate and produce few false positives. If neither predicts an object detection, then we compare the normalized confidences of the Haar decision trees against each other and a threshold for positive detection.

Since our program uses a sliding window over each frame, there are obviously many overlapping detections. We suppress any substantially overlapping detections of the same label, by selecting the larger ones. Any detections that are entirely contained within other detections are also rejected. There are a few more object type specific rules as well, to remove or preserve overlapping detections.

For speed, we only run detection over every other frame. In between, we use optical flow to shift the detection windows of the previous frame and use those as our detections.

Section 2 discusses the features used by our program in more detail. Section 3 discusses the different classifier types. Section 4 describes how we combine the output of our different classifiers and the output over different windows in a single frame. Section 5 describes how we perform various speedups, including choosing thresholds, optical flow and resizing of frames. Experimental results and conclusions are in Section 6 and 7.

2. Features

2.1. Haar

Haar features are the features used in the milestone and described in the final project handout. A Haar feature consists of a Haar template positioned at offset (x, y) from the top left corner of image patch. The templates are made up of arrangements of black and white polygonal regions. The response of an image patch with respect to a Haar feature is the difference between intensity values on the image patch between areas in the black region and areas in the white region.

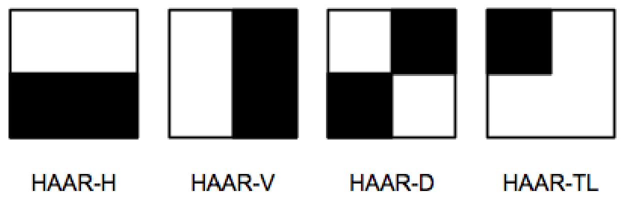

Figure 1. Examples of Haar features (borrowed from final project handout).

For objects with consistent patterns in intensity variation, such as a human face where the eye region is usually darker than forehead region, Haar features can work well for object detection (see [8]). However, it does not always work well on objects with high variation in color intensity and texture within the same object class. For our project, the Haar features used in the milestone are too simple to detect the five different kinds of objects we want.

2.2. Texture

A texture feature is a randomly generated square region cropped from a positive training image. Texture features were successfully used in [7]. Textures are more complex than Haar feature templates, and can capture small, unique details such as the fulcrum of a pair of scissors.

Texture features can be encoded in a similar fashion as Haar features. Each extracted texture feature includes the following information: an square kernel image $K^{s \times s}$ of size s representing the cropped subregion of the original (grayscale) positive training image; the (x, y) coordinates of the offset of the top left corner of the kernel, relative to the top left corner of original image; the width W and height H of the original image; and the mean intensity value m of the original image, used for intensity normalization.

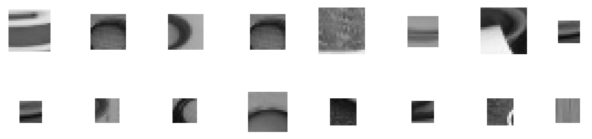

Figure 2. Examples of generated texture features for scissors.

Let $I^{W \times H}$ be the resized image region of interest for which we want to calculate the response with respect to a texture feature. The response is the simply normalized dot product of the kernel of the texture feature and the corresponding subregion of I:

$$R = \frac{\sum_{0 \le i,j \le s} \left(I'(x+i, y+j) \times K'(i,j)\right)}{\sqrt{\sum_{0 \le i,j \le s} I'(x+i,y+j)^2 \times \sum_{0 \le i,j \le s} K'(i,j)^2}}$$

where $I'(i,j) = I(i,j) - m_I$, $K'(i,j) = K(i,j) - m$, and m_I is the mean value of I. This operation normalizes the intensity values so that the matching is more invariant to lighting changes.

For our project, since all original positive training images have $W \ge H$, we randomly picked s, x, y such that $H/5 \le s \le H/2$, $0 \le x + s \le W$, and $0 \le y + s \le H$. Because these parameters are picked randomly, multiple features can be generated from a single positive training image. In our implementation, we generate three template features per training image. A classifier such as boosted stumps can pick the most useful features and discard the rest.

Texture features do not work well for all object types, because they depend on qualities such as color and light patterns which do not always characterize an object type.

2.3. Gradient

Gradient features are simply features extracted from a gradient image. The gradient image emphasizes the outlines and strong textures of an original image, rather than its color and minor textures [2].

Operations involved in generating gradient features are very similar to those of texture features, but instead of using the original grayscale positive training images as the source, the gradient image calculated using a normalized Sobel filter is used. Each feature is again represented by $K^{s \times s}$, (x, y), and W, H as in described in the texture features section. For each gradient image from the positive training set, we again generate 3 gradient features.

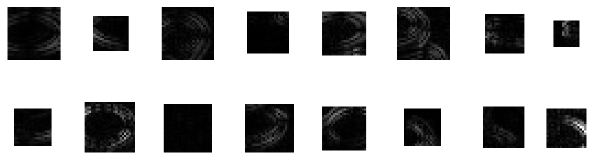

Figure 3. Examples of generated gradient features for scissors.

Let $I^{W \times H}$ be the gradient resized image region of interest for which we want to calculate the response with respect to a gradient feature. The response equation is:

$$R = \frac{\sum_{0 \le i,j \le s} \left(I(x+i, y+j) \times K(i,j)\right)}{\sqrt{\sum_{0 \le i,j \le s} I(x+i,y+j)^2 \times \sum_{0 \le i,j \le s} K(i,j)^2}}$$

Note that there is no normalization operation with respect to the mean of the image.

2.4. Fast Fourier Transform

Because keyboards are laid out as rows of keys (see Figure 4), images of them are correspondingly periodic. To use this domain knowledge, we designed the following feature based on the Fast Fourier Transform

(FFT). Classification using Fourier Transform was inspired by [6].

Figure 4. A sample keyboard image.

Given an input image I, let I_{ij} denote the i,jth pixel and I_i be the ith row of the image. We compute the FFT of each row which is denoted by $\mathcal{F}(I_i)$. The absolute value of the result from the FFT are taken and summed over all rows.

$$\tilde{I}_j = \sum_i |(\mathcal{F}(I_i))_j|$$

The result of computing $\tilde{I}$ for keyboard and a non-keyboard negative image is shown in Figure 5. From the graph, we see that the signal-to-noise ratio of keyboard images is much higher than that of negative images.

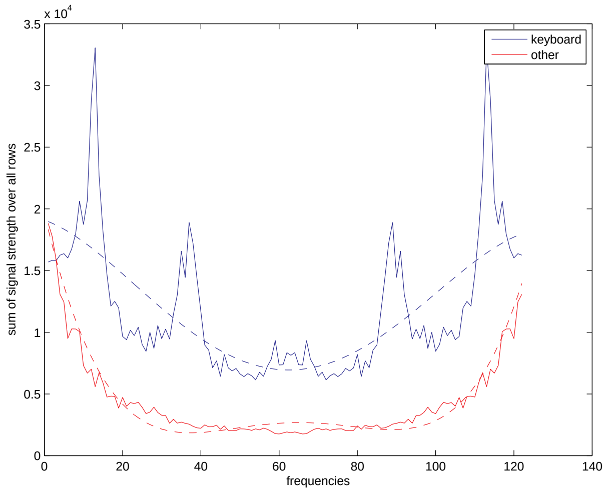

Figure 5. FFT frequency response of two images.

To compute an estimate of the signal to noise ratio, we fit a fourth degree polynomial to $\tilde{I}$. For $\tilde{I}$ which is an 1-by-m vector, we construct the matrix A with $A_{ij} = i^{j-1}$. Then we solve the normal equation for the coefficient vector b.

$$A^T Ab = A^T I^T$$

The result of polynomial fitting is shown in Figure 5. The residue of the polynomial fit is normalized by $\tilde{I}_j$. We take the absolute value of the mean of the normalized residues as our feature reponse R for keyboards.

$$R = \left| \frac{1}{m} \sum_{j=1}^{m} \left(1 - \frac{1}{\tilde{I}_j} \sum_{i=0}^{4} b_i j^i \right) \right|$$

In training, we compute this FFT feature response for all keyboard and non-keyboard images. The probability distribution of keyboards and other images with respect to this FFT feature is show in Figure 6. This result shows clear distinction between keyboard and negative images, providing solid evidence for FFT as a feature for detecting keyboards.

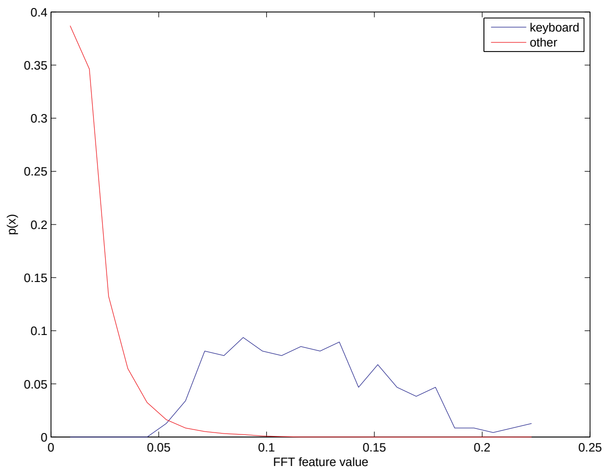

Figure 6. Probability distribution of our FFT feature value.

2.5. Hough Transform

Because Haar features do not describe clocks well, we apply the Hough transform [5] to find circles as an additional identifying feature of clocks. Intuitively, analog clocks like those in our training set are round in shape, whereas non-clock images do not have large, approximately centered circles.

We use the OpenCV implementation of the Hough circle transform, which first runs edge detection and then accumulates points based on the local gradient.

On the training set, the presence or absence of a circle is a fairly strong indicator for clocks. There are no circles detected in the images of square clocks, but the transform does find circles for the octagonal clocks. There are a number of false circles detected in mug images, due to designs on the mugs and the circular handles. A couple of clock and mug images with Hough circles superimposed are shown in Figure 7.

	Hough transform circles		
	#found	#total	#found/total
mug	54	294	0.18
stapler	0	392	0.00
keyboard	5	84	0.06
clock	**43**	**48**	**0.90**
scissors	2	306	0.01

Table 1. Counts of training images in which circles were detected, counts of total training images, and the fraction of training images in which circles were detected.

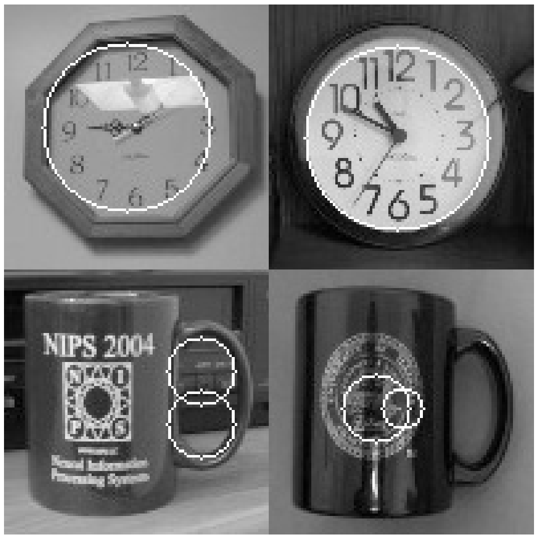

Figure 7. True circles detected on clocks, including the octagonal clock. False circles detected on mugs.

3. Classifiers

3.1. Decision Trees

We use standard decision trees, with modifications to the positive confidence measure.

3.1.1. Weighting by Number of Samples

The most straightforward confidence output for a particular decision leaf is the number of positive training samples that end up at the leaf, divided by the total number of samples at the leaf. Thus, the confidence c_i for leaf i is bounded, $0 \leq c_i \leq 1$.

A problem with this approach is that a leaf with 100 positive samples out of 100 total samples gives the same confidence as a leaf with one positive sample and no other samples. Clearly, the latter should be less confident.

To compensate for this, we weighted each confidence c_i with the number of positive samples p_i at that leaf, divided by a weighted average of the confidences of the entire tree.

$$c'_i = \frac{p_i c_i}{\frac{\sum_i p_i c_i}{\sum_i p_i}}$$

One drawback to this strategy is that the confidences are no longer bounded between 0 and 1. However, the confidences of different trees can still be measured relatively.

3.1.2. Weighting by Systematic Examination

The stapler decision tree has high precision but low recall. That is, when it labels a stapler detection, it is usually right, but it misses out on detections. This suggests that the confidence is too low or the threshold for a positive detection is too high. Since we want to use the same threshold for all decision trees, we weight the confidence before comparison.

3.1.3. Weighting by Hough Circle Transform Output

For mug and circle candidate detections, we also run the Hough circle transform to give additional information about a window. As seen above, circles are found in clock images with much higher probability than in mug images. Specifically,

$$P(\mathsf{circle}|\mathsf{clock}) = 0.90$$

$$P(\mathsf{circle}|\mathsf{mug}) = 0.18$$

However, we are interested in the posterior probabilities, $P(\mathsf{clock}|\mathsf{circle})$ and $P(\mathsf{mug}|\mathsf{circle})$. Bayes' theorem relates the conditional probabilities by the following equations:

$$P(\mathsf{clock}|\mathsf{circle}) = \frac{P(\mathsf{circle}|\mathsf{clock})P(\mathsf{clock})}{P(\mathsf{circle})}$$

$$P(\mathsf{mug}|\mathsf{circle}) = \frac{P(\mathsf{circle}|\mathsf{mug})P(\mathsf{mug})}{P(\mathsf{circle})}$$

If we assume that the prior probabilities of object classes are the same, that is, $P(\mathsf{clock}) = P(\mathsf{mug})$, then given that a circle is detected and no other information, it is five times more likely that an object is a clock than a mug.

$$\frac{P(\mathsf{clock}|\mathsf{circle})}{P(\mathsf{mug}|\mathsf{circle})} = \frac{0.90}{0.15} = 5$$

Similarly, given that a circle is not detected and no other information, it is eight times more likely that an object is a mug than a clock.

$$\frac{P(\text{clock}|\neg\text{circle})}{P(\text{mug}|\neg\text{circle})} = \frac{0.82}{0.10} = 8.2$$

Thus, if a circle is detected, we increase the clock confidence and decrease the mug confidence; if a circle is not detected, we decrease the clock confidence and increase the mug confidence.

3.2. Boosted Decision Stumps

As shown in lecture, using boosting to create an ensemble of decision trees or stumps can do better than a single decision tree, as boosting systematically picks the "hardest" examples to train new tree or stump. In this project, we use boosted decision stumps constructed with the AdaBoost algorithm as described in [3].

Each learned decision stump stores the threshold it splits on and a reference to the feature this threshold response corresponds to. The feature can be a texture or gradient feature; the AdaBoost algorithm picks the best feature to use in the learning process.

Figures 2 and 3 shows the top 16 scissors features picked by the algorithm. We can see that the features picked tend to be regions with corners or other complex textures. The features not picked by AdaBoost algorithm when constructing decision stumps are discarded and only the features actually used are stored.

Up to 50 stumps are constructed for each type of object we are trying to detect. Another stopping criterion is $\alpha_i^2 \leq 0.0001$. Since α_i is the weight for the ith stump, a small weight means that little progress is being made, and we consider the algorithm to have converged.

3.3. Logistic Regression

Given a set of features $\mathbf{x}$, we use logistic regression [4] to pick the best threshold vector to separating the positive and negative training set, where the classifer is in the form of $h_\theta(\mathbf{x}) = g(\theta^T\mathbf{x})$ with $g(x) = \frac{1}{1+e^{-z}}$. The threshold vector is computed by batch gradient descent. In particular, given a single feature, such as our FFT feature, we use logistic regression to find the separating threshold for the feature response.

4. Combining Detection Candidates

4.1. Over a Single Window

The difficulty with using different types of classifiers per object type is that the individual confidences are not easily normalized against a standard metric, so they cannot be directly compared. To combine the results of the different classifiers over the same window, we use a hand-tuned decision tree that splits on the confidences.

Since our scissors classifier is very conservative and rarely has false positives, we first consider the output of that classifier. If the confidence is greater than a certain threshold, then we label the window a scissors detection. If it is not, then we keep traversing down the tree.

Next is our keyboard classifier, which is also fairly accurate. In the absence of classifiers for other objects, the classifer performs reasonably well for detecting keyboards, as shown in Figure 8. If the keyboard classifier output is positive, then we label the window a keyboard detection. If it is not, then we consider the other classifiers.

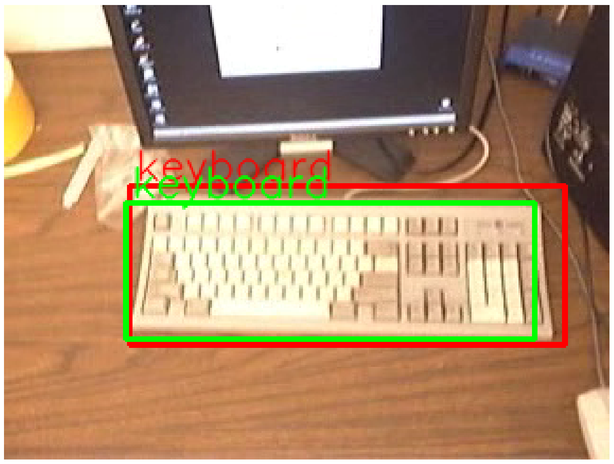

Figure 8. Keyboard Detection using FFT feature

The mug, stapler, and clock classifiers are all decision trees, so their confidences are more readily comparable. We pick the highest confidence output by any of the three, and if that confidence is higher than our threshold, than we label the window a detection of whichever object the most confident tree corresponds to.

4.2. Over One Frame

Since we use a sliding window over a frame, there are many overlapping detections for the same object.

If two detections of the same label overlap, we usually pick the larger one, since smaller windows typically correspond to portions of an object and we want a bounding box around the entire object. For scissors, however, the windows detected by our classifier are never too small. In fact, the smallest windows are the tightest boxes, and the most accurate, so we suppress the others.

If two detections of different labels overlap substantially, we need more sophisticated logic. If one detection window is entirely contained within another, we reject the small one. If two windows each cover a large portion of the other (in practice, this often occurs for mugs and clocks), then we pick the window with higher confidence.

5. Speedups

5.1. Choosing Threshold for Decision Trees

Suppose we have n training samples at a leaf that we want to split. If we choose the threshold by naively going through all possible thresholds and computing the entropy by counting the number of positive and negative samples in the left leaf, this would take $O(n^2)$ operations. Instead, we sort the samples by their feature response and incrementally updating the number of positive samples in left leaf, this would take $O(n \log n$ to sort and $O(n)$ to find the threshold with minimum entropy.

5.2. Optical Flow

We compute the average optical flow [1] of the entire video frame from the last frame, and use the resulting vector to shift our detection windows. Therefore, we only need to run our detection code once every several frames, greatly speeding up testing. The result is shown in Figure 9.

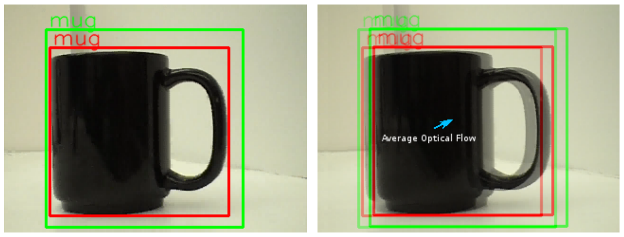

Figure 9. Using optical flow to shift detections

5.3. Resizing Frames

Instead of performing sliding window detection using windows of different sizes, and then resizing each window to 64×64 pixels, we resize each frame of video to recursively smaller sizes and run sliding windows of fixed size on them. This saves a fair amount of computation time for resizing images.

6. Experimental Results

6.1. Default Case

Our default baseline score is from using one-against-many Haar decision trees running on `easy.avi`. This is simply an extension to the milestone with a decision tree for each of the five objects we are trying to detect.

Although we do fairly well in this case detecting presence of mugs (recall of 0.452685) and staplers (recall of 0.306306), because of the large number of overlapping detections, we end up with F1 scores of 0.127292 and 0.359788 for mugs and staplers respectively. There are no scissors and very few clocks detected so the overall F1 score is 0.119418. In general, this case has too many overlapping regions, causing false positives.

6.2. Overlap Suppression vs. Default

Overlap suppression helps reduce the problem of false positives by preventing most overlapping detections from being added to the objects list. With overlap suppression turned on, we get similar number of true positives as the default case, but many fewer false positives, as expected. The F1 scores for mugs and staplers are now 0.426471 and 0.468966. The overall F1 score with suppression is 0.348148.

6.3. Combined Detector

We see from the default case that scissors are not detected using decision trees alone. Using decisions stumps to detect scissors, we get an F1 score of 0.229299 for scissors. Combined with weighted confidences for better detection of mugs, clocks, and staplers, we get an overall F1 score of 0.379005.

6.4. Hough Transform for Clocks and Mugs

As discussed earlier, using the Hough circle transform to distinguish clocks and mugs can potentially improve the detection of each. By turning on circle detection for cases when a region can be either a clock or mug, we get an overall F1 score of 0.383713. The improvement comes primarily from reduced number of false positives for mugs.

6.5. Keyboard Detection

Without the FFT keyboard detector, we cannot locate any keyboard objects in our video. In the `hard2.avi` video, we actually detect nothing besides keyboards.

6.6. Optical Flow

Under heavy machine load, the combined detector performs at around one frame per 1.5 seconds. To speed up the process, optical flow is turned on to track detected windows. The detection is only performed every three frames while optical flow tracks the two frame in between. With this, we get a 2x speedup. Overall F1 score is 0.368349 with optical flow turned on.

7. Conclusions

We investigated many different extensions with theoretical or intuitive motivations, but in the end the most successful classifier was the original, one-against-many Haar decision trees, merged together based on unweighted confidences, with some basic overlap suppression.

We identify the major problem with our project as not combining the different features well.

References

[1] S. Beuchemin and J. Barron. The computation of optical flow. *ACM Computing Surveys*, 27:433–467, 1997.

[2] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, August 2002.

[3] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[4] D. W. Hosmer and S. Lemeshow. *Applied logistic regression (Wiley Series in probability and statistics)*. Wiley-Interscience Publication, September 2000.

[5] P. Hough. Machine analysis of bubble chamber pictures. In *Proc. Int. Conf. High Energy Accelerators and Instrumentation*, 1959.

[6] S. Tiwari, S. Ramachandran, A. Bhattacharya, S. Bhattacharya, and R. Ramaswamy. Prediction of probable genes by fourier analysis of genomic sequences. *Comp. Appl. BioSci.*, 13(3):263–270, June 1997.

[7] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *Computer Vision and Pattern Recogntion*, pages 762–769, 2004.

[8] P. Viola and M. Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.

A. Evaluation

```
** Default Case
Confusion matrix:
        mug     scis.   clock   keyb.   stap.   other
mug     177     0       0       0       0       214
scis.   0       0       0       0       0       138
clock   0       0       27      0       0       157
keyb.   0       0       0       0       0       0
stap.   0       0       0       0       34      77
other   2213    13      654     0       44      0

Score on each of the individual classes:
mug:
        true positives: 177
        false positives: 2213
        false negatives: 214
        precision: 0.0740586
        recall: 0.452685
F1 score: 0.127292
scissors:
        true positives: 0
        false positives: 13
        false negatives: 138
        precision: 0
        recall: 0
F1 score: 0
clock:
        true positives: 27
        false positives: 654
        false negatives: 157
        precision: 0.0396476
        recall: 0.146739
F1 score: 0.0624277
stapler:
        true positives: 34
        false positives: 44
        false negatives: 77
        precision: 0.435897
        recall: 0.306306
F1 score: 0.359788
Overall scoring information:
        true positives: 238
        false positives: 2924
        false negatives: 586
        precision: 0.0752688
        recall: 0.288835
        FINAL F1-SCORE: 0.119418

** Supression
Confusion matrix:
        mug     scis.   clock   keyb.   stap.   other
mug     174     0       0       0       0       217
scis.   0       0       0       0       0       138
clock   0       0       27      0       0       157
keyb.   0       0       0       0       0       0
stap.   0       0       0       0       34      77
other   251     11      29      0       0       0

Score on each of the individual classes:
mug:
        true positives: 174
        false positives: 251
        false negatives: 217
        precision: 0.409412
        recall: 0.445013
F1 score: 0.426471
scissors:
        true positives: 0
        false positives: 11
        false negatives: 138
        precision: 0
        recall: 0
F1 score: 0
clock:
        true positives: 27
        false positives: 29
        false negatives: 157
        precision: 0.482143
        recall: 0.146739
F1 score: 0.225
stapler:
        true positives: 34
        false positives: 0
        false negatives: 77
        precision: 1
        recall: 0.306306
F1 score: 0.468966
Overall scoring information:
        true positives: 235
        false positives: 291
        false negatives: 589
        precision: 0.446768
        recall: 0.285194
        FINAL F1-SCORE: 0.348148

** Combined Detector
Confusion matrix:
        mug     scis.   clock   keyb.   stap.   other
mug     176     0       0       0       0       215
scis.   0       18      0       0       0       120
clock   0       0       49      0       0       135
keyb.   0       0       0       0       0       0
stap.   0       0       0       0       35      76
other   283     1       18      0       63      0

Score on each of the individual classes:
mug:
        true positives: 176
        false positives: 283
        false negatives: 215
        precision: 0.383442
        recall: 0.450128
F1 score: 0.414118
scissors:
        true positives: 18
        false positives: 1
        false negatives: 120
        precision: 0.947368
        recall: 0.130435
F1 score: 0.229299
clock:
        true positives: 49
        false positives: 18
        false negatives: 135
        precision: 0.731343
        recall: 0.266304
F1 score: 0.390438
stapler:
        true positives: 35
        false positives: 63
        false negatives: 76
        precision: 0.357143
        recall: 0.315315
F1 score: 0.334928
Overall scoring information:
        true positives: 278
        false positives: 365
        false negatives: 546
        precision: 0.432348
        recall: 0.337379
        FINAL F1-SCORE: 0.379005

** Optical Flow
Confusion matrix:
        mug     scis.   clock   keyb.   stap.   other
mug     177     0       0       0       0       214
scis.   0       21      0       0       0       117
clock   0       0       36      0       0       148
keyb.   0       0       0       0       0       0
stap.   0       0       0       0       36      75
other   285     3       21      0       63      0

Score on each of the individual classes:
mug:
        true positives: 177
        false positives: 285
        false negatives: 214
        precision: 0.383117
        recall: 0.452685
F1 score: 0.415006
scissors:
        true positives: 21
        false positives: 3
        false negatives: 117
        precision: 0.875
        recall: 0.152174
F1 score: 0.259259
clock:
        true positives: 36
        false positives: 21
        false negatives: 148
        precision: 0.631579
        recall: 0.195652
F1 score: 0.298755
stapler:
        true positives: 36
        false positives: 63
        false negatives: 75
        precision: 0.363636
        recall: 0.324324
F1 score: 0.342857
Overall scoring information:
        true positives: 270
        false positives: 372
        false negatives: 554
        precision: 0.420561
        recall: 0.32767
        FINAL F1-SCORE: 0.368349

** Optical Flow
Confusion matrix:
        mug     scis.   clock   keyb.   stap.   other
mug     176     0       0       0       0       215
scis.   0       18      0       0       0       120
clock   0       0       49      0       0       135
keyb.   0       0       0       0       0       0
stap.   0       0       0       0       35      76
other   265     1       18      0       63      0

Score on each of the individual classes:
mug:
        true positives: 176
        false positives: 265
        false negatives: 215
        precision: 0.399093
        recall: 0.450128
F1 score: 0.423077
scissors:
        true positives: 18
        false positives: 1
        false negatives: 120
        precision: 0.947368
        recall: 0.130435
F1 score: 0.229299
clock:
        true positives: 49
        false positives: 18
        false negatives: 135
        precision: 0.731343
        recall: 0.266304
F1 score: 0.390438
stapler:
        true positives: 35
        false positives: 63
        false negatives: 76
        precision: 0.357143
        recall: 0.315315
F1 score: 0.334928
Overall scoring information:
        true positives: 278
        false positives: 347
        false negatives: 546
        precision: 0.4448
        recall: 0.337379
        FINAL F1-SCORE: 0.383713
```