

Respectful Cameras Overview

Input Frame



Output Frame



Our input is a 320x240-pixel mJPEG video stream from a Panasonic HCM280A webcam. The mJPEG stream is a series of jpeg images, transmitted at 20 frames per second (FPS). Each pixel has a Red, Green, and Blue (RGB) component and 8 bits describe each component. Our application, written in C++, processes this input and handles each image independently. The objective is to detect the yellow hat, and then overlay a circular disk to conceal the identity of the person wearing the hat. Our output is a new stream of JPEGs, with overlaid disks.

To determine the hat's location in each image, we identify which colors corresponds to the hat and then find sufficiently large spatial clusters of hat pixels. We first train the system by manually selecting a set of 200 hat pixels, and a set of 200 non-hat pixels. A statistical learning and classification technique, AdaBoost[1], is used on these training pixels with a set of weak hypotheses to generate a final classifier described as a linear function of these hypotheses. The weak hypotheses used to classify the pixels are axis-aligned hyper-planes. Our final classifier uses 30 weak hypotheses. An advantage of AdaBoost is that it selects the "best classifying" hypotheses from our much larger set (in our case 4590 features).

Let X be a feature space, $Y \in \{0,1\}$ be an observation space and $H = \{h : X \rightarrow Y\}$ be a set of weak hypotheses. The objective of AdaBoost is to define a function $f : X \rightarrow Y$ by constructing a linear function of elements from H that accurately predicts Y given X . To determine this function, we use a set of training data: $\{(x_i, y_i) \mid x_i \in X, y_i \in Y\}$. In AdaBoost, the function f is defined to minimize: $\mathbf{E}[l(f(X), Y)]$ where

$$l(\hat{Y}, Y) = \begin{cases} 1 & \text{if } \hat{Y} \neq Y \\ 0 & \text{otherwise} \end{cases}. \text{ To achieve this goal, AdaBoost}$$

uses the greedy strategy whereby at each iteration, it selects a classifier that minimizes the number of incorrectly labeled pixels according to a distribution. It uses this new classifier to update the distribution to reflect new correctly classified pixels, where the incorrectly weighted pixels are made more significant in the distribution, implicitly making correctly weighted pixels less significant.

Because we perform color tracking, we define our feature space as a 9-tuple of a pixel's RGB values, followed by the projection of that pixel into the HSV and LAB spaces. HSV is a description of color in terms of hue, saturation, and value, which performs well with colors varying intensities that occur during lighting changes. LAB is a color description where L corresponds to luminosity (between white and black), A transitions a color between magenta and green, and B transitions a color between yellow and blue. LAB was designed to model how humans see color, being more perceptually linear, and is particularly well suited for determining specularities. An advantage of AdaBoost is that we can use very simple features, which will allow us to classify pixels very rapidly, as we require for real-time processing. We use the union of two sets of very simple weak hypothesis, which describe all possible axis-aligned hyper-planes:

$$h_{d,i}(X) = \begin{cases} 1 & \text{if } X[d] \leq i \\ 0 & \text{otherwise} \end{cases}, \bar{h}_{d,i}(X) = \begin{cases} 1 & \text{if } X[d] > i \\ 0 & \text{otherwise} \end{cases}$$

$$H = \left\{ h_{d,i} \cup \bar{h}_{d,i} \right\} \forall d \in \{1, \dots, 9\}, \forall i \in \{0, \dots, 255\}.$$

After training AdaBoost on a set of manually chosen good and bad colors, we train it to determine which pixels correspond to our feature, in this case the yellow construction hat. Because there will always be noise in the video, we then determine connected components of the labeled good pixels to remove false positives. For connected components, good pixels a and b are in the same group if there is a path from a to b through adjacent good pixels. We use the minimum area of each group to determine which of these color clusters correspond with a hat. Finally, we overlay a colored dot on the bottom edge of each group's bounding box. The dot's size is proportional to the area of the bounding box. Our system uses produces a new video stream with these overlaid dots, thereby preserving privacy.

This is our preliminary approach, our next steps will include temporal modeling and adaptive feature tracking.

[1] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory: Eurocolt 1995*, pages 23-37. Springer-Verlag, 1995.