Real time power toggling from camera enabled devices: EyeSwitch Pro

Michael Johnson
University of Washington, mjj47@cs.washington.edu

Ryan McMahon
University of Washington, ryanm35@cs.washington.edu

INTRODUCTION

EyeSwitch Pro is a new mechanism for interacting with home electronics. EyeSwitch Pro is a system that allows users to toggle power in real time using any camera enabled device. This project aims to create an ecosystem that makes the use of home electronics more accessible to those with physical and mobile impairments. In the United States, there are 35.2 million individuals that have difficulty with physical activity, 250,000 with spinal cord injuries, and 30,000 with ALS. These individuals could potentially benefit from EyeSwitch Pro by having a new, non-physically demanding, way of interacting with the home. The full EyeSwitch Pro ecosystem was implemented including hardware switches, a registration server, a correlation server, and an EyeSwitch Pro compatible device. For this project, an Android app is used as an example of an EyeSwitch Pro device. The eventual motivation is to move the client application to the Pupil Pro eye tracker to increase the size of the potential benefactors. The Pupil Pro tracks a user's pupil and projects the pupil focal point onto a 2D image of the surroundings. After the introduction, an overview of the EyeSwitch Pro Ecosystem is given, followed by a look at improvements from EyeSwitch to EyeSwitch Pro, as well as a conclusion and future works.

FIGURE 1
PRODUCTION MODEL OF HARDWARE SWITCH

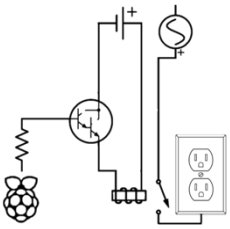

FIGURE 2
CIRCUIT DIAGRAM OF SWITCH

OVERVIEW

The EyeSwitch ecosystem allows a user to toggle power to household electronics by simply looking at the electronic with a Pupil Pro while blinking, or by capturing a photo of the electronic with a compatible EyeSwitch device. The electronic needs to be connected to an EyeSwitch hardware unit. The user takes an initial video of the electronic along with its configuration identifier and then subsequent photos of electronic toggle power.

EYESWITCH ECOSYSTEM

1. EyeSwitch Hardware

The EyeSwitch hardware includes the physical switches that toggle power to household electronics. A Raspberry Pi 3.0 is used as the brains; all the logic and computing in the hardware switch is done in the Raspberry Pi. The Raspberry Pi is connected to an electrical circuit that enables power to be toggled by flipping a pin on the output board. Please see Figure 2 for a diagram of the electrical circuit. The hardware switch uses an industrial grade relay switch that supports 120 volts at 60 hertz and up to 16 amps of current. The box contains two electrical outlets that can be used to power household electronics. Please see Figure 2 for an example mock-up of a production hardware switch.

The hardware switch is IPv4 addressable. The unit broadcasts its IPv4 address to the registration server discussed in the next section. A hardware switch also has a statically preconfigured configuration identifier that is broadcasted with the IPv4 address. The hardware switch contains a server that accepts requests using the EyeSwitch power state change protocol. When the hardware switch is plugged into the wall, it automatically connects to the registration server and begins broadcasting its information.

2. Registration Server

The EyeSwitch registration server contains information about hardware switches and their IPv4 addresses and configuration identifiers. The registration server sits at a well-known IPv4 address that is statically configured in the hardware switches. The registration server contains two sets of functionality. The server supports a *put* operation that inserts a mapping of an IPv4 address to a configuration identifier. The server also supports a *get* operation that fetches an IPv4 address based on a configuration identifier.

3. Correlation Server

At a high level, the correlation server is dedicated to resolving image queries to see if they map to any known configured device identifiers. The configuration server handles two operations: Append Configuration and Query Database. The following describes the two operations as well as other correlation server algorithms.

3.1 Append Configuration:

When users want to add a new household electronic to the EyeSwitch ecosystem, they upload a configuration from an EyeSwitch Device to this server. A configuration consists of a configuration identifier from an EyeSwitch hardware device and an RGB video of the household electronic. The server will then take the video and extract key frames from it sampled at a fixed rate at 3Hz. Feature extraction is run on each frame and stored in association with the configured identifier. Using all the features from the video, the correlation server builds a model for the device.

3.2 Query Database:

The other operation the Correlation Server supports is querying. A query request contains an RGB image and metadata. Metadata includes focal point, GPS location, and the requestor's IP address. The server processes this query by finding a candidate model subspace from all images, then runs the Object Matching algorithm against these candidate images. Both algorithms are discussed in the following sections labeled "Model Subspace Reduction" and "Object Recognition." If there is a match found, the server will send a request to the registration server with the matched identifier to toggle the power to the matched hardware.

3.3 Correlation Server Algorithms:

3.3.1 Model Subspace Reduction:

A single Correlation Server is used to process all configuration uploads and query requests in the ecosystem; however, there could many configuration models in the database. As the size of potential models grow, issues arise with both accuracy and performance. To combat this, for every query, a pruned list of potential models is computed based on metadata. To do this, two pieces of metadata are relied on from the user query: IP address and GPS location.

IP Address Pruning: This assumes that the EyeSwitch Pro Hardware and EyeSwitch Pro Device will be on the same network when attempting to interact. Using this assumption, the prefix of the IP addresses are compared. If they match, the image is considered and passed on to the GPS pruning stage.

GPS Pruning: The model subspace is further pruned based on GPS location. This assumes configured devices will not be moving over time. With this, the GPS location and confidence of the query image is compared to the locations of the configured images. If the confidences are high enough, and the distances are outside a threshold, the model is excluded; otherwise, the model is considered as a candidate.

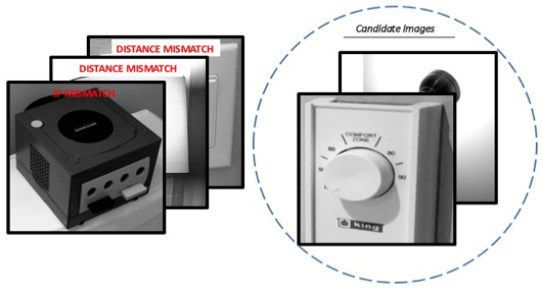

FIGURE 3
VISUALIZATION OF IMAGE SUBSPACE REDUCTION

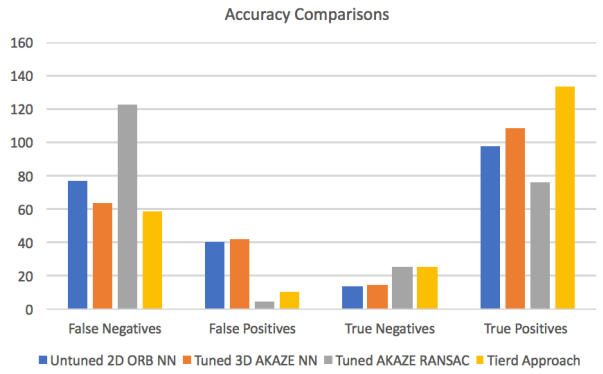

FIGURE 4

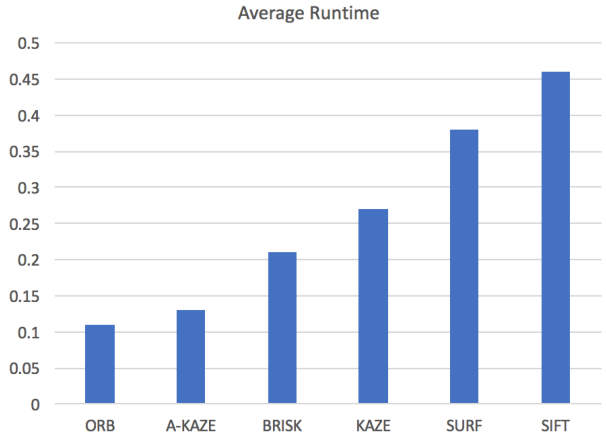

FIGURE 5

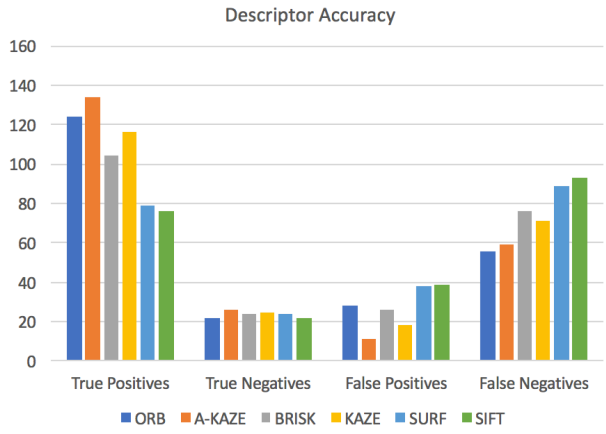

FIGURE 6

3.3.2 Object Recognition:

The EyeSwitch Pro object recognition algorithm is run for all queries to the Correlation Server against the candidate model subspace.

A wide variety different object matching techniques were implemented and evaluated including 2D to 3D image projection, 2D video to stereo maps, calculating homographies, and nearest neighbor. The final solution was chosen to provide the best accuracy while still providing real time performance. The algorithm runs a two-tiered matching system where no-match queries from the first tier are passed to the second for more lenient matching. A two-tiered system is used due to the nature of the two matcher abilities, the first tier minimizes false positives while sacrificing some true positives, the second tier minimizes false negatives while allowing more false positives. Successes of the tiered approach can be seen in Figure 4. Before matching is run, features are extracted from the query image as input to our matchers. From experimentation, Accelerated KAZE features give the best balance of accuracy and performance. See Figure 5 for details on the descriptor runtime performance.

Tier 1 - RANSAC Homography Calculations:

1. **Compute good feature matches against candidate models.** A KNN matcher finds the N best matched features from a query feature to model features. "Good" matches are defined as matches between candidates and models where the top match confidence is more than 0.7 times the next closest match as per Lowe's ratio test.
2. **Compute Homography if there are enough good matches.** From experimentation, it is found that if there are at least 6 "good" matches between a query image and a model, homography calculations prove highly successful. Therefore, if a query and model share less than 6 good matches, no homography match is reported for this model.
3. **Compute Homography.** Uses RANSAC to sample good match points. Given sampled points, computes a homography.
4. **Compute Percent Inliers from homography**. Given our homography, the good matches are checked to see what percentage of the matches agree with the homography. The higher the percent inliers, the higher the scoring function gives towards this query belonging to the model class.
5. **Accept or Reject Homography Scores.** If a single homography is found between the query and a model, the query is accepted as belonging that model. If multiple homographies are found, the query is accepted as belonging to the model with the highest percent inliers. If no homographies are found, the extracted features are passed to the second matching tier.

Tier 2 - Nearest Neighbor Query Features to Model Features:

1. **Compute feature matches against candidate images.** A brute force matcher is used to compare queries to models due to small model subspaces, however, as configured subspaces begin to grow, a Flann based matcher is used as a replacement for performance improvements. The matcher projects the features into *n* dimensional feature space and runs nearest neighbor.
2. **Weight feature matches based on the distance to the query focal point.** Because EyeSwitch is built with the Pupil Pro in mind, the location of the pupil is known in query images (and in our Android App, this is simulated by the click location). Knowing the interaction location allows the matching to focus more on feature matches that are closer to the pupil focal point. Matches are weighted using a Gaussian function giving a better score to those matches that are closer to the focal point.
3. **Find best scoring match.** The top *n*, where *n* is a decided threshold, weighted matches are then used to compute a score for the query image against each of the candidate images. The best scoring candidate model is then compared to a threshold. If the score is lower than the false positive threshold, it is considered a match. If the best score of any model to the query is above the threshold, the query image is said to not match any configured devices.

3.3.3 Threshold Learning

A significant challenge with object detection is having enough features to accurately represent a candidate objects. However, for EyeSwitch, only a single video of a household electronic is required to begin querying against. Because users only upload a single, potentially brief, video, there had to be a way of growing the model of the electronic. To enhance the feature representations, high scoring homography based query images are used to grow the representation of the model. This learning scheme helps allow for users to improve their query accuracy over time.

4. Tuning

The two-tiered approach used in the EyeSwitch Pro correlation server has a wide dimensionality of possible parameters to tune. Tuning the parameters to provide the optimal tradeoff between accuracy and latency was critical to maintain the real-time requirement of EyeSwitch Pro. A few examples include when to switch between a brute force matcher and a FLANN when calculating top matches of descriptors, false positive thresholds from nearest neighbor matching, and RANSAC acceptance parameters. Over forty thousand simulations over fifteen test sets were evaluated and analyzed to provide the optimal combination of parameters.

5. Comparison of EyeSwitch and EyeSwitch Pro

5.1 Latency

Query latency is a large concern for the EyeSwitch Pro ecosystem. Device power toggling more than 0.4 seconds from query time is deemed unacceptable as a user experience. While the EyeSwitch Pro matching algorithm runtime grew from EyeSwitch, overall latency, as seen in Figure 8, was reduced due to optimizations made in the EyeSwitch Pro protocol as well as the Android App. The protocol now supports compressed videos and image reducing transfer time. The Android application now can reuse TCP connection for queries that are near in time as well as improving memory buffering.

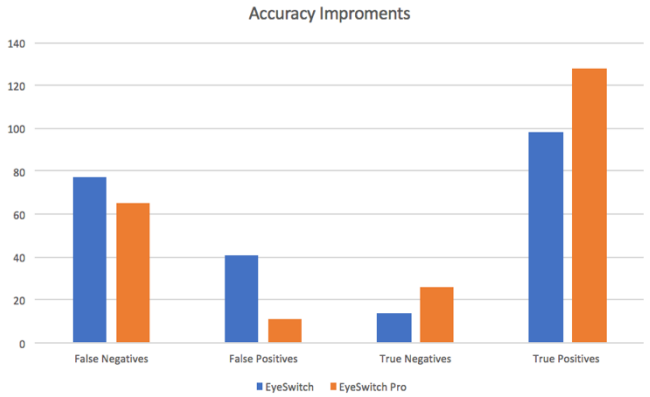

FIGURE 7

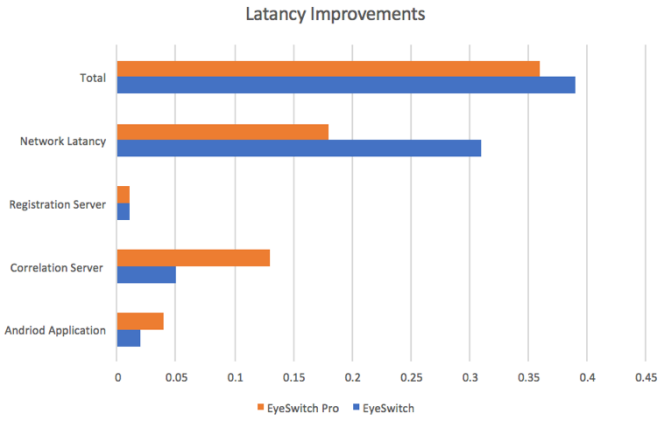

FIGURE 8

FIGURE 9
EYESWITCH MODE ON ANDRIOD APPLICATION

5.2 Correlation Server

The two-tiered system of EyeSwitch Pro correlation server vastly improves the accuracy versus the previous EyeSwitch correlation server. The previous version of the correlation server used a untuned nearest neighbor approach on a single image to match a query image to the candidate model subspace. The current approach discussed in the previous section improved the accuracy across all four metrics, as seen in Figure 6 and when combined with advances in compression and CPU utilization provides an overall better runtime. In Figure 6 the EyeSwitch Pro correlation server decreased the number of false positives. False positives are deemed to be far worse than false negatives. Accidently turning the wrong device on/off is an extremely detrimental user experience.

6. EyeSwitch Devices

The EyeSwitch Device is the main portal of interaction to the EyeSwitch ecosystem. The user interface of these devices is designed with the Pupil Pro (our target device) in mind; however, any device that has a camera and network access is supported. The EyeSwitch devices allow users to upload configuration videos as well as submit query images to potentially toggle power. The prototype for this project was built as an Android application. A screenshot of the app can be seen in Figure 8. The Android app simulates the pupil from the Pupil Pro via touches to the screen.

CONCLUSION

With the implementation of the hardware switch, registration server, correlation server, and EyeSwitch device, a new mechanism has been created for interacting with electronics. EyeSwitch improves the quality of life for those with difficulties with physical actions or mobility.

FUTURE WORKS

Further work could be done to improve the ease of configuration. The configuration process could also be improved by using NFC or Bluetooth to correlate a household electronic with a photo instead of a static configuration identifier. The EyeSwitch ecosystem could also be modified to leverage its real-time object recognition to track objects. While the current implementation simply returns a single correlation match and score, future works could extend this to sequential camera frames to track objects. The ability to track many pre-configured objects has many implications including further improvements of EyeSwitch Pro, robotics, smart home electronics, and others.