Facial Detection and Tracking by Autonomous Drone

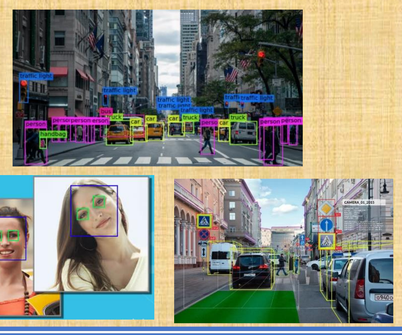
Austin Philips
Student
Utah Valley University
Austin.Philips@uvu.edu

Angel Rodriguez
Student
Utah Valley University
Angel.Rodriguez@uvu.edu

***Mohammad Shekaramiz**
Faculty
Utah Valley University
mshekaramiz@uvu.edu

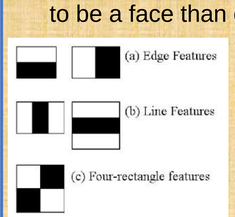
Problem Statement

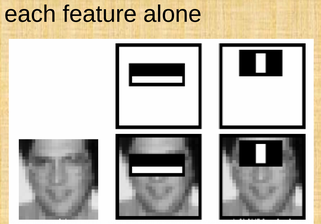
- Computer vision is a strongly growing presence in military, security, and entertainment
- Images and live video feed can be used for tracking persons or objects
- Cascade Classifiers can be used to detect faces of the general population or faces of authorized specific users (i.e. face ID)



Face Detection using Haar Cascades

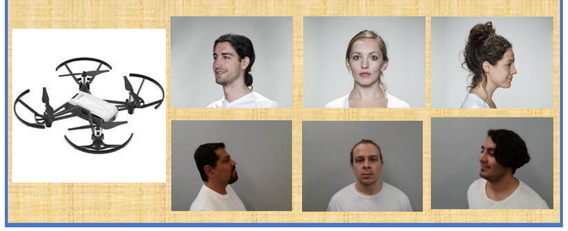
- Machine Learning (ML) models must utilize many positive (face in frame) and negative (face not in frame) images to learn what features to extract to determine what makes a face a face
- Each feature is a single value obtained by subtracting the sum of pixels under the white rectangle from the sum of pixels under the black rectangle.
- Consider the image below right; Top row shows two good features, based on supervised training: eyes region being darker than nose/cheeks and eyes being darker than nose bridge
- Features as such are used by the ML model to determine likelihood of face in image
- Feature of eyes with nose/cheeks more likely to be a face than each feature alone





Data Set Collections

- Supervised ML algorithms rely on being trained with very large data sets that included positive and negative images (they cannot know which features are important without false images to show the difference)
- Existing database collection London Face Lab set was utilized for training as a negative set.
- Positive set included varying photos of Austin, Angel, and Dr. Shekaramiz taken in similar fashion to database set to train the model; photos taken with Tello drone
- Good practice is to have negative set closely related to positive set, in this case, facial photos



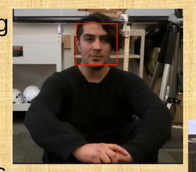
User Verification through ML

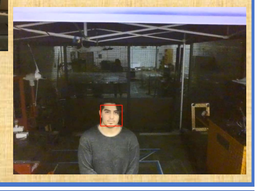
- Convolutional Neural Networks (CNN) ResNet50V2 and VGG19 and the linear Support Vector Machine (SVM) were tested to see if they could verify the user using the above dataset.
- The perfect accuracy found in all three models suggest underlying issues. The dataset was partitioned correctly, suggesting the photos inside the dataset were too similar to each other or the dataset has some other fundamental issue.

	ResNet50V2	VGG19	SVM - Linear
Testing Accuracy	1.0	1.0	1.0

Using Haar Cascade with Drone

- Haar Cascade models produce .xml files that can be used for image processing/computer vision with faces using open-cv library
- Python files draw bounding boxes around detected faces in drone's camera stream using open-cv and djitellopy libraries
- The drone will detect faces and output live video feed of bounding boxes using Python threads

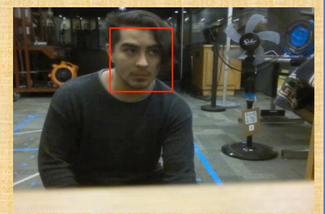




Calculating Distance

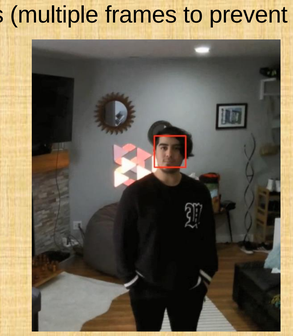
- **A bounding box can tell the distance of a face by its size, larger box means further away while smaller box means closer distance to drone**
- Testing was done to ensure that the calculated distance was within a few cm of actual distance

```
distance: 87
distance: 85
distance: 83
distance: 82
distance: 81
distance: 86
```



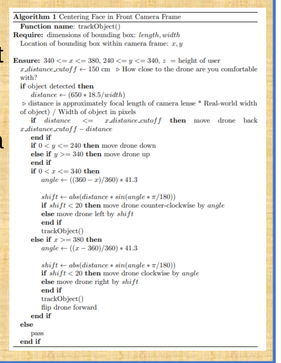
Facial Detection Initialized Takeoff

- **Like devices which require voice input to activate, the drone is programmed to require facial recognition to takeoff**
- The drone will remain on standby until a face is recognized for a duration of 10 frames (multiple frames to prevent error)
- One takeoff is initialized, the drone will enter a state of continuous facial tracking
- Figures on the right are about the same time frame from different views



Tracking Face Algorithm

- **Recursive code used for centering face within camera frame**
- Recursive algorithm used is designed to shift the drone laterally and adjust height to keep face in center of camera frame
- Every return of the function results in the drone doing a flip to let the user acknowledge it is finished with a call of the function



Algorithm 1 Centering Face in Front Camera Frame

Function name: trackObject()
Require: dimensions of bounding box: *length, width*
Location of bounding box within camera frame: x, y

Ensure: $340 <= x <= 380$, $240 <= y <= 340$, z = height of user
$x\_distance\_cutoff \leftarrow 150$ cm ▷ How close to the drone are you comfortable with?
if object detected **then**
distance $\leftarrow (650 * 18.5/width)$
▷ distance is approximately focal length of camera lense * Real-world width of object) / Width of object in pixels
if *distance <= x_distance_cutoff* **then** move drone back *x_distance_cutoff − distance*
end if
if $0 < y <= 240$ **then** move drone down
else if $y >= 340$ **then** move drone up
end if
if $0 < x <= 340$ **then**
$angle \leftarrow ((360 - x)/360) * 41.3$
$shift \leftarrow abs(distance * sin(angle * \pi/180))$
if $shift < 20$ **then** move drone counter-clockwise by *angle*
else move drone left by *shift*
end if
trackObject()
else if $x >= 380$ **then**
$angle \leftarrow ((x - 360)/360) * 41.3$
$shift \leftarrow abs(distance * sin(angle * \pi/180))$
if $shift < 20$ **then** move drone clockwise by *angle*
else move drone right by *shift*
end if
trackObject()
flip drone forward
end if
else
pass
end if

Conclusion/Future Work

- The ML models used to perform verification of the user performed too well on the testing dataset but performed poorly in the field, suggesting issues with the dataset.
- Future work will include tracking of hand gestures for additional drone commands and maneuvers.

Acknowledgments

- L. DeBruine and B. Jones, "Face research lab london set figshare. dataset." 2017, https://doi.org/10.6084/m9.figshare.5047666.v5.
- The authors would like to thank Hunter Dalsing (ECE student) from Utah Valley University for his contributions in providing database set.