# CLPS 0950 Project Presentation

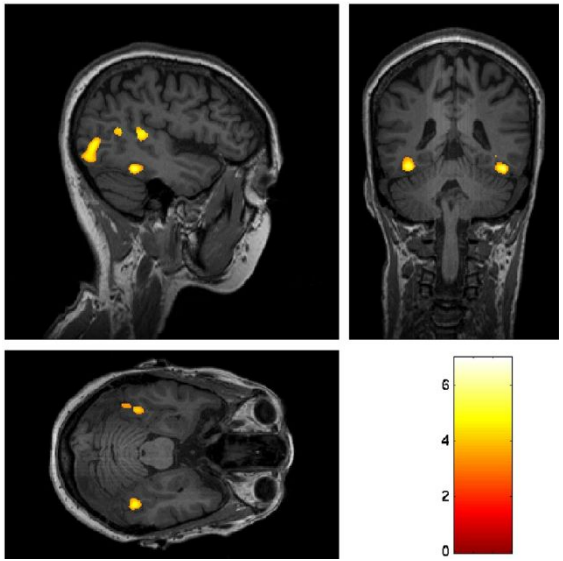
Mira White and Priya Bhanot

# Our Project: Face Generator + Face Finder

- Project Goal: Create a basic model emulating the cognitive process for facial recognition
- Face Generator:
  - Function that is able to create images that depict a face (or can be edited to depict NOT a face)
    - Not a face may contain some, but not all facial features and/or facial features incorrectly oriented on the face
- Face Finder
  - Takes matrix input (1s & 0s) and determines whether the face contains the given facial features (two eyes, nose, and mouth) and whether they are correctly oriented
    - If all features are present and oriented correctly → a face is detected

# Our Motivation

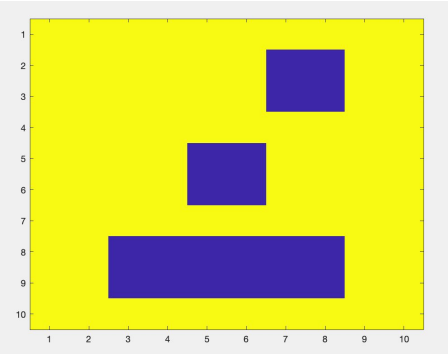
- Brain Model
  - Feature dependent: individual features of face are perceived → sent to higher cortical areas for facial recognition/analysis
  - Fusiform face area/IT: dominant higher-order brain regions responsible for facial recognition
- Our Model
  - Detects whether four basic facial features (two eyes, nose, mouth) are present and oriented correctly with respect to each other → final facial detection
  - We chose to program a facial recognition generator that would be able to determine if what was shown was representative of a face or not



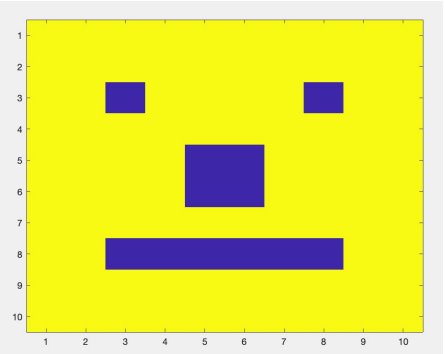

Activated parts of the brain when looking at a face/recognizing features

# Our Approach

- Define face / facial features / facial orientation
  - Face contains 4 basic facial features: 2 eyes, 1 nose, 1 mouth
  - Face contains these features oriented with two eyes (set symmetrically) above the nose (centered on face) above the mouth (same width as eye distance)
  - Approach: created numerous faces/not faces matrices manually to mathematically define a facial feature/orientation
- Created basic face Finder function that could detect faces/not faces for simple cases
- Created face Generator to more efficiently create tailored inputs to test face Finder function
- Broadened faceFinder + faceGenerator to apply to matrices of any size (best results for n >= 9)
- Added conditional statements to return different outputs depending on what facial features were present even in NOT face inputs



NOT a Face Example



Face Example

# Skills Learned

- Learned how to properly code more complex functions
- Determine mathematical relationships → translate into code to create functions
- How to broaden the scope of a function to accept more varied inputs
- Create functions that could be easily edited to customize outputs (face vs. not face outputs)
- Use conditional statements to analyze different inputs
- Test functions using a main script
- Debugging/Optimization

```
if mod(n,2) ~= 0 %if n is odd
    center = round(n/2);
    eye_local = round(center/2);
    unit = center - eye_local;
    mouth_local = center + unit;
    mouth_width = floor(.75*n);
    %create nose
    image(center,center) = 0;
    %create eyes
        %to only remove one eye, delete one element of the index array in
        %the column placement of image
    image(eye_local,[eye_local, center+unit]) = 0;
    %create mouth
    image(mouth_local, eye_local:center+unit) = 0;
```

```
    %mouth detection
    if image(even_center + even_eyelocal, even_eyelocal:even_center+even_center - even_eyelocal+1) ==0
        mouth_detected = 1;
    else
        mouth_detected = 0;
    end
end

if ((nose_detected == 1) & (eyes_detected ==1) & (mouth_detected ==1))
    face_detected = 1;
    disp ('Face detected')
elseif (nose detected == 0)
    face_detected = 0;
    disp ('There is no nose, no face detected')
elseif (eyes detected == 0)
    face_detected == 0;
    disp ('There are no eyes, no face detected')
elseif (mouth_detected == 0)
    face_detected = 0;
    disp ('There is no mouth, no face detected')
end
end
```

ive feedback