Is all your tech **distracting** you?

Even if cell **phones** are turned off and turned face down, their mere **presence reduces** people's **cognitive capacity**

Brain Drain: The Mere Presence of One's Own Smartphone Reduces Available Cognitive Capacity

Adrian F. Ward, Kristen Duke, Ayelet Gneezy, and Maarten W. Bos,

Journal of the Association for Consumer Research 2017 2:2, 140-154

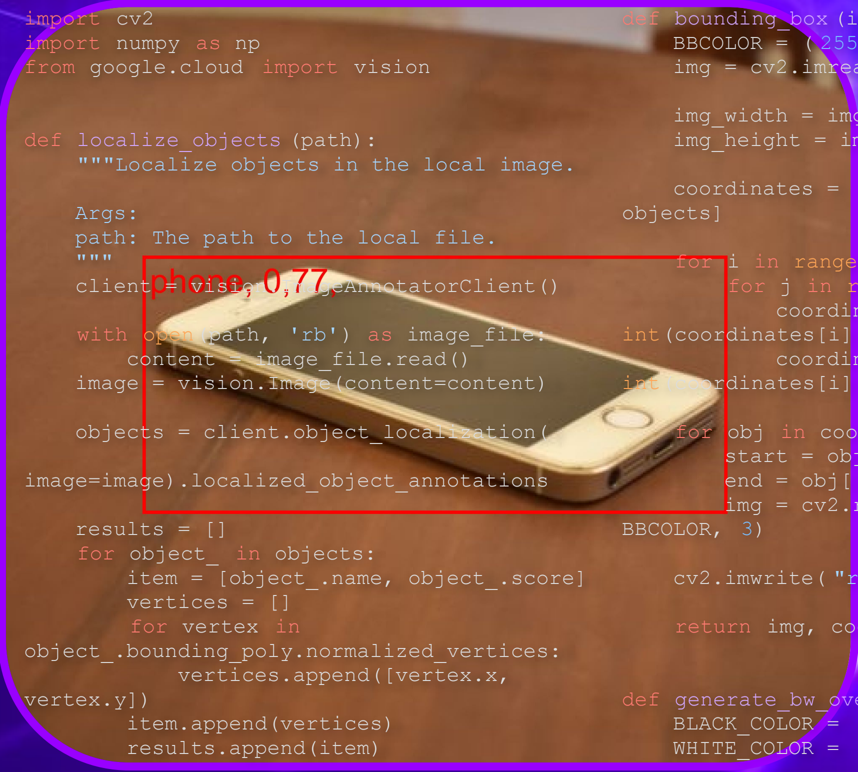
Add **more tech** to **eliminate tech** distractions!

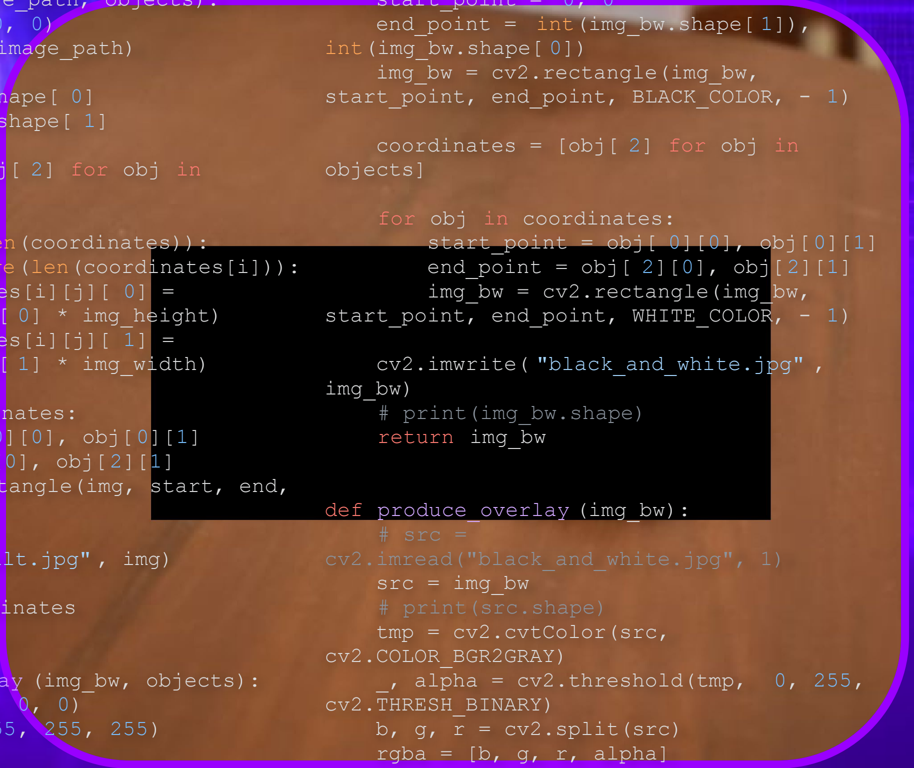
Trust us, it works!

DIMINISHED REALITY

With your headset look and decide what to eliminate

And after some behind the scenes magic





```
import cv2
import numpy as np
from google.cloud import vision


def localize_objects(path):
    """Localize objects in the local image.

    Args:
    path: The path to the local file.
    """
    client = vision.ImageAnnotatorClient()

    with open(path, 'rb') as image_file:
        content = image_file.read()
    image = vision.Image(content=content)

    objects = client.object_localization(

image=image).localized_object_annotations

    results = []
    for object_ in objects:
        item = [object_.name, object_.score]
        vertices = []
        for vertex in
object_.bounding_poly.normalized_vertices:
            vertices.append([vertex.x,
vertex.y])
        item.append(vertices)
        results.append(item)

    return results
```

```
def bounding_box(image_path, objects):
    BBCOLOR = ( 255, 0, 0)
    img = cv2.imread(image_path)

    img_width = img.shape[ 0]
    img_height = img.shape[ 1]

    coordinates = [obj[ 2] for obj in
objects]

    for i in range(len(coordinates)):
        for j in range(len(coordinates[i])):
            coordinates[i][j][ 0] =
int(coordinates[i][j][ 0] * img_height)
            coordinates[i][j][ 1] =
int(coordinates[i][j][ 1] * img_width)

    for obj in coordinates:
        start = obj[ 0][0], obj[0][1]
        end = obj[ 2][0], obj[2][1]
        img = cv2.rectangle(img, start, end,
BBCOLOR, 3)

    cv2.imwrite( "result.jpg" , img)

    return img, coordinates

def generate_bw_overlay(img_bw, objects):
    BLACK_COLOR = ( 0, 0, 0)
    WHITE_COLOR = ( 255, 255, 255)

    start_point = 0, 0
    end_point = int(img_bw.shape[ 1]),
```

```
    start_point = 0, 0
    end_point = int(img_bw.shape[ 1]),
int(img_bw.shape[ 0])
    img_bw = cv2.rectangle(img_bw,
start_point, end_point, BLACK_COLOR, - 1)

    coordinates = [obj[ 2] for obj in
objects]

    for obj in coordinates:
        start_point = obj[ 0][0], obj[0][1]
        end_point = obj[ 2][0], obj[2][1]
        img_bw = cv2.rectangle(img_bw,
start_point, end_point, WHITE_COLOR, - 1)

    cv2.imwrite( "black_and_white.jpg" ,
img_bw)
    # print(img_bw.shape)
    return img_bw

def produce_overlay(img_bw):
    # src =
cv2.imread("black_and_white.jpg", 1)
    src = img_bw
    # print(src.shape)
    tmp = cv2.cvtColor(src,
cv2.COLOR_BGR2GRAY)
    _, alpha = cv2.threshold(tmp, 0, 255,
cv2.THRESH_BINARY)
    b, g, r = cv2.split(src)
    rgba = [b, g, r, alpha]
    dst = cv2.merge(rgba, 4)
    cv2.imwrite( "overlay.png" , dst)
```

it’s gone!

Add more, to get less

The **Future** is **Present**

DIMINISHED
REALITY

Tech stack

- Computer Vision
- Figma
- HTC Vive XR Elite
- Machine Learning
- OpenCV
- Python
- Stable Diffusion
- Unity

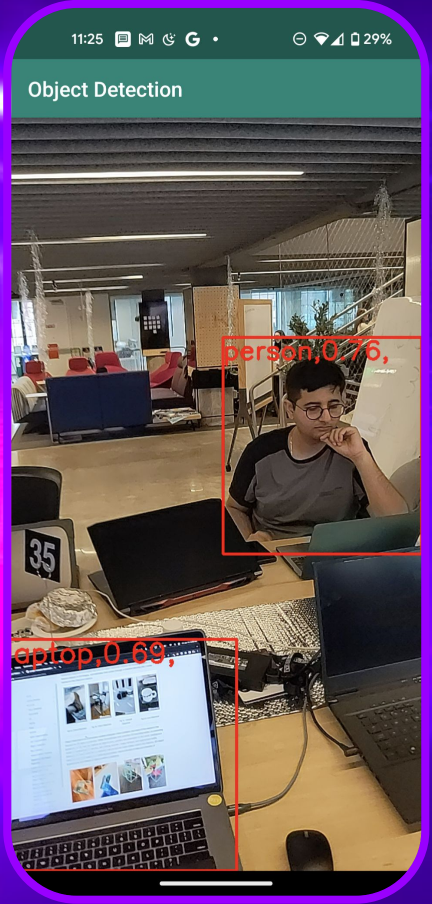
Brian McDonald, Grace Ng, Jash Rathod, Lisa Szolovits, Steph Ng

Appendix

Appendix - Ethics

Q: What about ethics? Do you allow rich people to block from ever seeing homeless people?

A: We only allow people to block objects - people and animals are not blocked



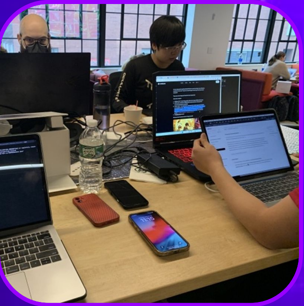
Appendix - Tech Approach Current

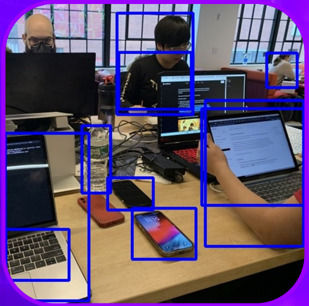
1. Get a screenshot
2. Run object detection and generate the mask
3. Inpaint using OpenCV or Stable Diffusion
4. Return overlay image to server
5. Imprint it on the anchored plane

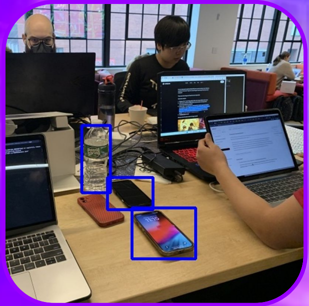
Appendix - Tech Approach Alternative

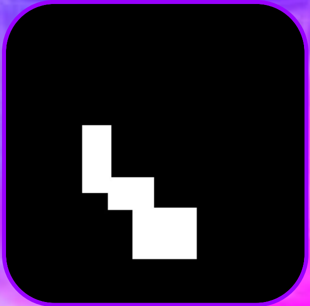
1. Get a screenshot every second
2. Run object detection and generate the mask
3. Inpaint using OpenCV or Stable Diffusion
4. Return overlay image to server
5. Resize and distort it to fit perfectly on the Mixed Reality feed for all the frames of the entire second (no anchor needed)

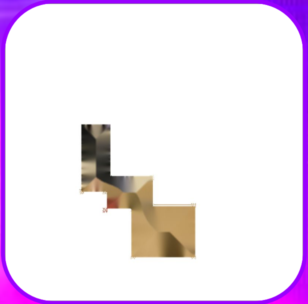
Appendix - Tech Current Approach

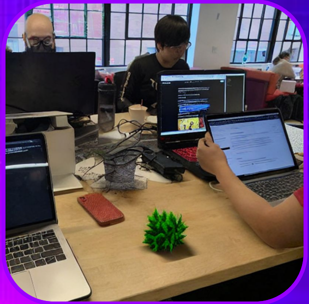

Initial Image


All Objects


Filtering

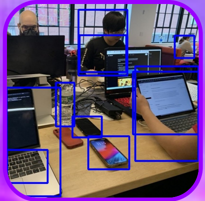

Mask

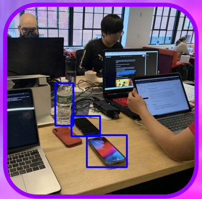

Overlay (OpenCV)

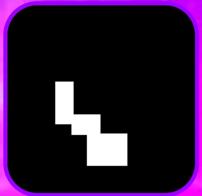

Stable Diffusion

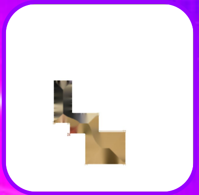
Appendix - Tech Current Approach

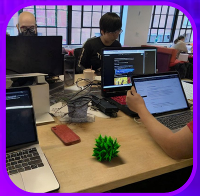
Initial Image


All Objects


Filtering

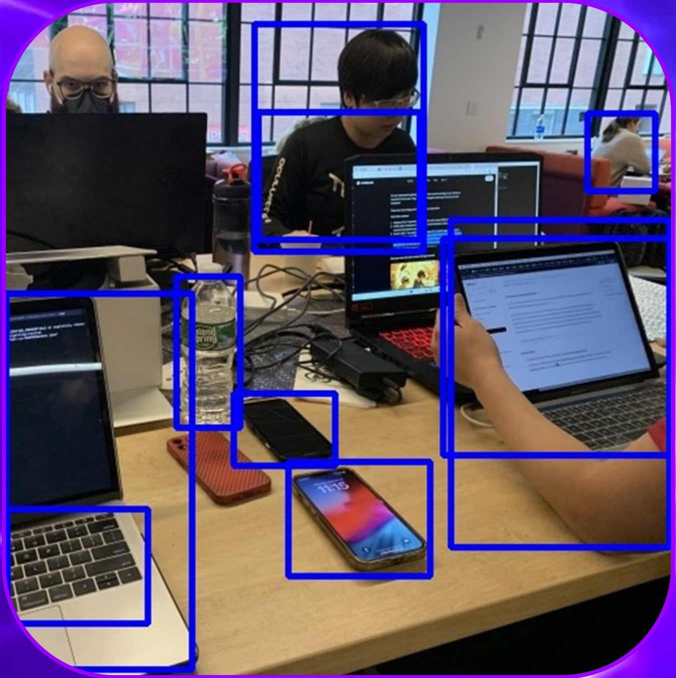

Mask

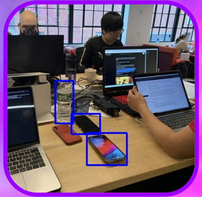

Overlay (OpenCV)

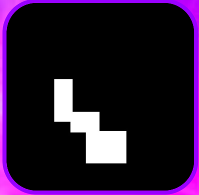

Stable Diffusion

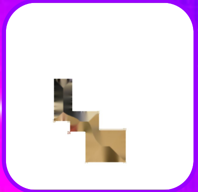
Appendix - Tech Current Approach

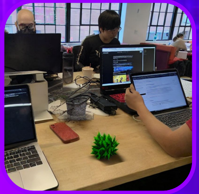
Initial Image


All Objects


Filtering

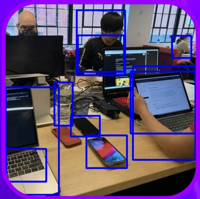

Mask

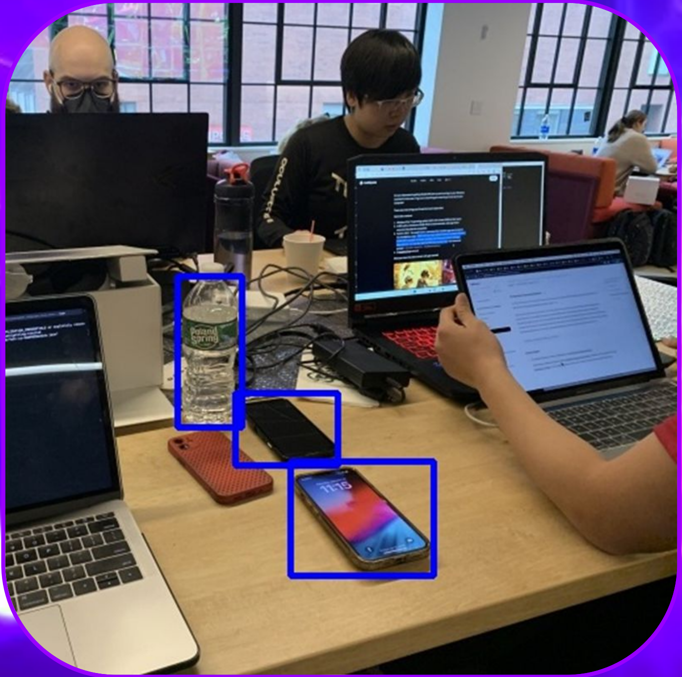

Overlay (OpenCV)

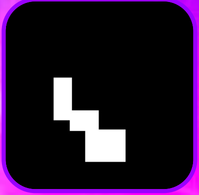

Stable Diffusion

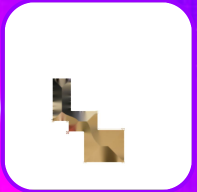
Appendix - Tech Current Approach

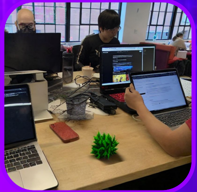
Initial Image


All Objects


Filtering

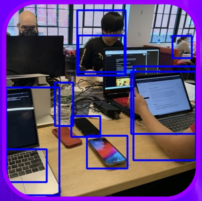

Mask

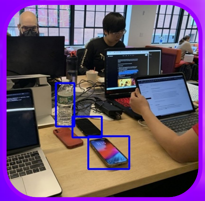

Overlay (OpenCV)

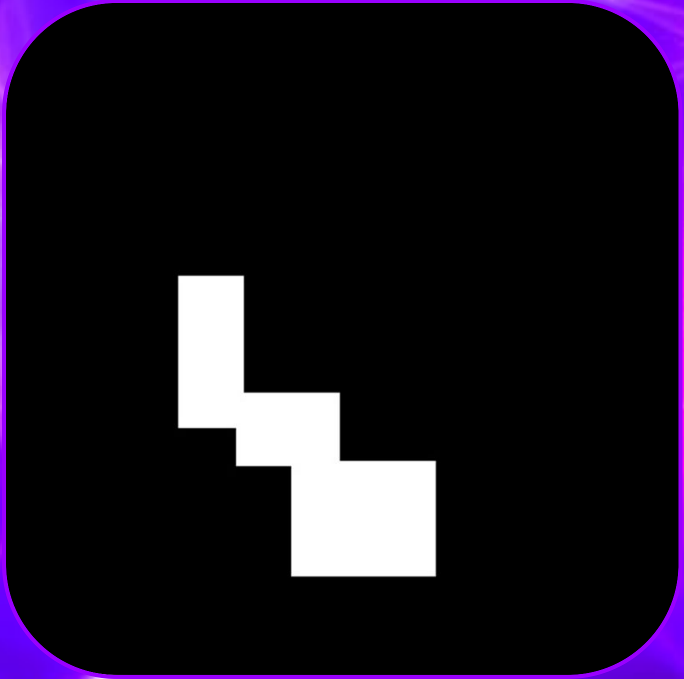

Stable Diffusion

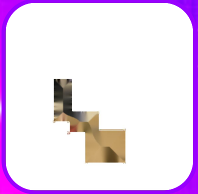
Appendix - Tech Current Approach

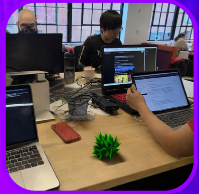
Initial Image


All Objects


Filtering

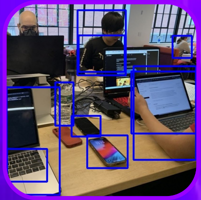

Mask

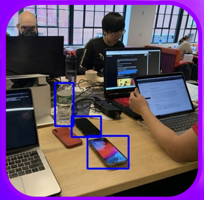

Overlay (OpenCV)

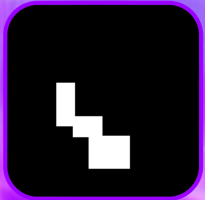

Stable Diffusion

Appendix - Tech Current Approach

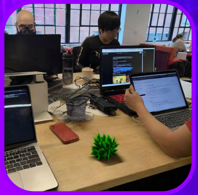
Initial Image


All Objects


Filtering

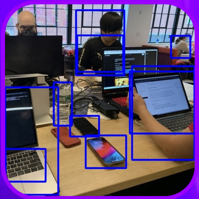

Mask

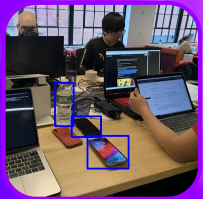
Overlay (OpenCV)

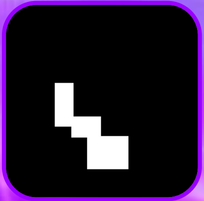

Stable Diffusion

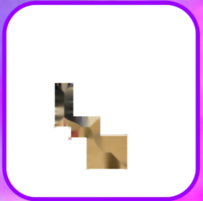
Appendix - Tech Current Approach

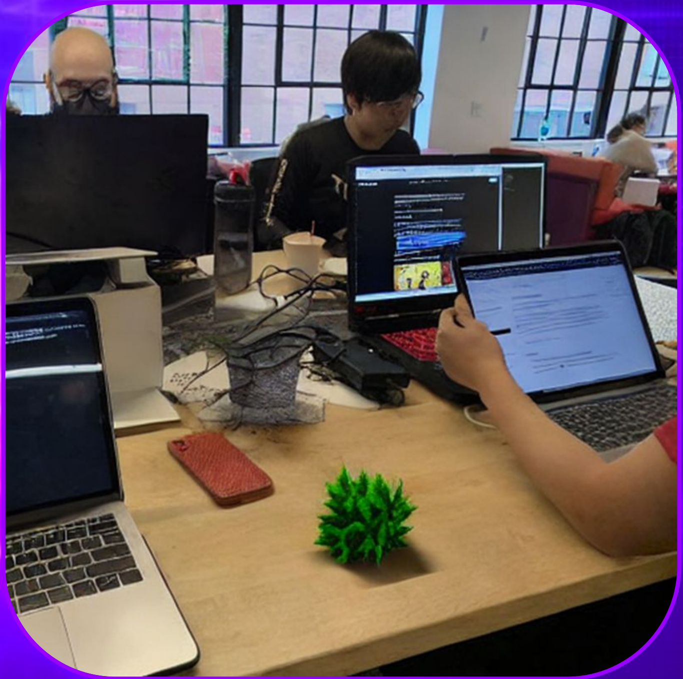
Initial Image


All Objects


Filtering


Mask


Overlay (OpenCV)


Stable Diffusion