RESEARCH

What is the problem we are trying to solve?

Human Interface Devices (HIDs)...

Spread Germs, Contaminants. 💉🦠

In **public places**, objects that are **touched frequently**, like **elevator buttons**, **point of sale terminals**, shared keyboards, and **kiosks** at stores can be **hot spots for germs**.

In a **laboratory** or factory, the **unintentional mixing** and **polluting** of materials can **compromise the accuracy** and **quality** of the product. **Worker interaction with HIDs** are the **source** of cross-contamination.

📵 Restrict freedom to use hands.

In places like a kitchen or restaurant, it is **inconvenient to mount or carry a HID**. Current HIDs **restrict** the use of **hands** and work **surfaces**. and **cannot** be used with **soiled** or **occupied** hands.

Are difficult to use by the disabled. 💻

Those with **disabilities,** natural or **Repetitive-Stress- Injuries**, are **unable** to **maneuver** physical HID devices.

Existing Solutions

LEAP Motion Controller

- Detects individual ligaments/ joints + projects virtual hands
- **Tiring** for arm, **not portable**
- Only works on PC + Stereo/optical
- Mounted **underneath** → can get soiled (ex. blood dripping down)

AirType by ippinka, Tap Strap 2

- Two bands which wrap around users' hands → defeats purpose as there's **still physical contact**!
- Problem: **cannot** be implemented in **public places** or devices with **multiple** users (**user-specific**!)

Microsoft XBOX 360 Kinect

- AI for Gesture Recognition; **RGB** cameras w/ structured light
- Stereo/optical cameras
- **Not portable** (**large**)
- **Lacks precision,** body-detection

Intel RealSense Developer Kit

- Demonstrates keyboard mapping
- However, **VERY compute intensive** → utilizes GPUs
- **Cannot** detect **multiple** fingertips + keypresses **simultaneously**

The existing solutions are **not practical, portable, or applicable to the touchless HID that is needed.** The use of **optical/stereo camera** severely **limits** their number of **use cases** (ex. no **low-light** conditions) and **accuracy**.

Introducing Toffi (Time of Flight For Interfacing)

Works everywhere, including low light or low color contrast environments

Customizable, natural, intuitive, easy-to-use solution

Portable, requires much less compute, and power (Battery-Powered Embedded System)

More accurate gesture-recognition than the current stereo/optical camera solution

Our solution will accurately track the **movement** and **depth change of a fingertip** and **translate** that to mouse movement and clicks **on a computer display**.

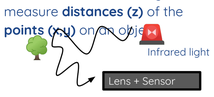
ToF + OpenCV

What are some of the resources we are using?

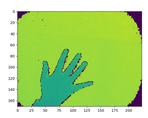
About the Time of Flight Sensor

What is a Time of Flight (ToF) Sensor?

- Uses *reflected infrared light* to measure **distances (z)** of the **points (x,y)** on an object



2D Array of Z-Values stored as numpy array

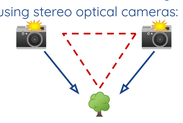


ToF produces 2D Array of Z-Values directly from sensor (graphically represented below)

Currently, ToF sensors are being used in a variety of applications, including:

- Phone facial recognition
- Self driving cars (LIDAR)
- Advanced AR experiences

Current method of 3D recognition using stereo optical cameras:



The fixed distance between the cameras and the relative image shift is used to calculate distances

Feature Comparison	ToF	Stereo
Can be useful in object ID & position	✔	✔
Not reliant on RGB (works in low light)	✔	
Only needs 1 sensor to object detect	✔	
Latency	↓	↑
O(n) to caculate z-array	↓	↑

About OpenCV

OpenCV

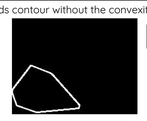
- **OpenCV** - Open Source Computer Vision
  - Free, open source library optimized for real time computer vision
  - Supports variety of languages, operating systems, and processors
  - Library includes over 2500 algorithms

💡 We **mapped** raw ToF sensor data to grayscale .bmp format in order to use OpenCV and Python **visualization** libraries

Key OpenCV Functions in Processing Pipeline

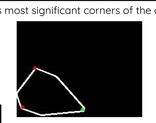
```
cv2.convexHull(maxContour)
```

Finds contour without the convexity dips



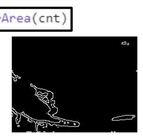
```
cv2.goodFeaturesToTrack(im,8,0.01,10)
```

Finds most significant corners of the contour



```
cv2.contourArea(cnt)
```

Finds area of a contour



💡 We experimented to find **compute-efficient** OpenCV algorithms

Using OpenCV, we **filter and process a frame** (camera input) to **find the exact coordinate location** of a fingertip.

HOW IT WORKS

How did we create, improve, and speed-up our algorithm?

Toffi Software Architecture

main.py

- Uses **multi-processing**
- main **starts** pipeline & mouse processes
- Uses **queue.Queue()** between processes

What is Multiprocessing?

- Uses **multiple cores**
- Allows **parallel processing**/execution

main
Gets camera frames

- Turns **camera on** and **sets frame** rate (45fps)
- Puts newly generated **camera frames into the queue** q
  - Loops through **process_event_queue**
- Continuously **updates** frames queue

```
def process_event_queue (q, painter, frames, seconds):
    # create a loop that will run for the given amount of time
    t_end = time.time() + seconds
    while time.time() < t_end:
        try:
            # try to retrieve an item from the queue.
            # this will block until an item can be retrieved
            # or the timeout of 1 second is hit
            item = q.get(True, 1)
        except queue.Empty:
            # this will be thrown when the timeout is hit
            break
        else:
            #painter.paint (item)
            #print("Main    :    average(item))
            frames.put(item, True, seconds)
```

1.

1.

Earliest to latest frames

1.[224][171]	2.[224][171]	3.[224][171]	4.[224][171]

queue is a FIFO in the shared memory
q is a queue between main and camera
frames is a queue between main and pipeline

pipeline
Extracts fingertip data

- **Spawned** python process
- Gets **most recent frame** from frames
- Uses openCV for **image processing**
- **Appends** location and click status (x, y, click) values to queue mouseCoords

mouse
Updates the cursor

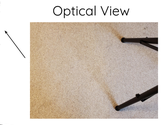
- **Spawned** python process
- Gets most **recent coordinates** from **mouseCoords**
- Uses **pyautogui** for current cursor position, **moving cursor** and **clicking**

Get background (During Initialization Only)

- Save first frame as background

START


ToF View

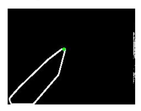

Optical View

2. Find Edges

- Finds the edges of the **hand**

4. Find Fingertip

- Finds **eight** corners on hull, choose the one that is **furthest from center**
- Use **rate of change** of **z** to **detect click**



- **Mouse Movement**
- dX and dY within respective thresholds
- **Smoothing** function
  - **Averaging** current and previous values to **reduce noise**
  - Common graphics algorithms
- **Mouse Click**
- When dZ > start **threshold**, turn on tracking of click
- Keep **track** of distance moved down
- If movement downward > click threshold
  - **Register** as **click**
  - Turn off click tracking
- If dZ < start threshold, **turn off** click tracking

dZ = **lastZ** - **currentZ** between two **successive** frames

Current Frame
224x171 numpy array

IMAGE PROCESSING PIPELINE

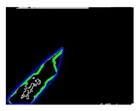
Finger Coords
Tuple of x, y, click values

1

1. Find and Contrast Hand

- **Subtracts** current frame from background frame
- **White** hand on **black** background

3



3. Find Contours and Convex Hull

- Uses **perimeter** and enclosed **area** for **max** contour

Key Design Decisions + Modifications

Library used to recognize fingertip

Tensorflow vs OpenCV

OpenCV
Simpler, faster, more robust

- Both can be **used to find fingertip** for the algorithm
- Because **OpenCV is non-neural network based**, it can run faster
- We are **not doing complex recognition**

Added to improve accuracy

Background Subtraction

Start of Program
Calibrates with background

- **Takes capture** at **beginning** of program
- Subtract **current frame** from background frame **remove background noise** in frames

Reduced cursor movement jitter

Smoothing Mouse Movement

Common Graphics Tool
More natural movement

- **Averages** current value with previous value to **smoothen movement**
- Goal is **less sharp movement** without **visual lag time** or **accuracy loss**

Improved fingertip detection

Determining Correct Tip from Contour

Center on Max Contour
Finds farthest corner from center

- After finding the **largest area** with **max contour**, finds **center**
- After finding corners, **corner farthest from center** is determined as **the fingertip**

Improving cursor "jumps"

Deque vs Queue

Queue
Looks through all frames generated

- **Deque (as LIFO), not thread-safe**
  - Gave **corrupted** frames
  - **Non-smooth** movement due to skipped frames
- **Queue (as FIFO), thread-safe**

Through extensive tuning and testing, we have implemented carefully thought out improvements to our program to improve its accuracy, smoothness, and performance.

Speeding up + Achieving 45 FPS

Debug Mode

- **Toggle** passed into pipeline
- **Debug ON** enables intermediate step of **saving images** as BMP files + **text logging**
- **Debug OFF** passes "image" as **numpy array** between openCV methods without saving files or producing logs

FIFO Instead of LIFO

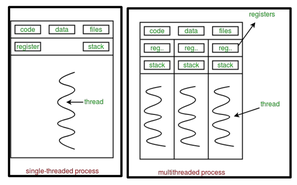
- FIFO - **First** in First out
- LIFO - **Last** in First out (like a stack)
- LIFO was **initially** used → rationale: process the **latest** frames that the ToF has gotten
- Problem: **skipped** frames caused incorrect movement, like moving "backward"

Pipeline Speedup

- Removal of "for" **loops** + intermediaries
  - **Vectorization,** optimized computations
- **Conversion** of **numpy** array **directly** to openCV format (skip disk read/write)
- Background subtraction → less compute time required to extract the hand
- Cython → **interpreted** vs. **precompiled**
  - Precompiled language that uses Python-like syntax
  - Uses terminal command to translate Python code into Cython code

Threading vs. Multiprocessing

- **Toffi uses multiprocessing**
- Uses **multiple cores**
- Allows **parallel processing**/execution
- Main spawns two other processes: **mouse** + **pipeline**
- **Camera** itself is a **thread** (does not need entire process)
  - All it does is query frames from roypy suite
- Multiprocessing sidesteps **Global-Interpreter-Lock** by using **sub-processes** instead of threads




```
frames = multiprocessing.Queue()
mouseCoords = multiprocessing.Queue()
pPipeline = multiprocessing.Process(target=procPip, args=(frames,mouseCoords,options,))
pMouse = multiprocessing.Process(target=smoothMove, args=(mouseCoords,options,))
pMouse.start()
pPipeline.start()
```

pPipeline
pMouse

```
pPool = multiprocessing.Pool(globals.threadPoolSize)
curMouseCArray = pPool.starmap(pipeline, listOfArgs)
for curMouseC in curMouseCArray:
    """
    Go through each (x, y, z), perform necessary
    click/movmement calculations, and put in mouseCoords
    """
```

Creates a pool, executes, and loops through the resulting fingertip coords

Pipeline Pooling

- Pool **distributes** the **tasks** to the **available processors** using FIFO **scheduling** and returns output as an **array**
- Ensures maximum CPU usage → **efficiency** + speed
  - No processors sitting idle
- **Pooled** every **four** frames for pipeline to process
- Usage of **pool.starmap**
  - For functions with more than one argument
- Provided a considerable **speedup**
- **PC Stick** has **4 cores**

Toffi Hardware

PMD Pico flexx TOF Camera

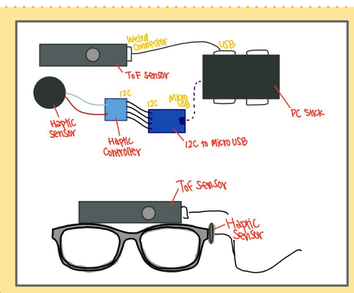
- 0.1 to 4m range
- Laser Class 1
- Dimensions: 68mm X 17mm X 7.35mm
- Resolution: 224 X 171 (38k) pixels
- Power: 300 mW
- Runs on Windows, Linux/ARM, Ubuntu Linux, macOS, Android/ARM

AZULLE Access3 Mini PC Stick

- Fanless embedded computer
- 4 GB RAM
- 32 GB storage
- Intel Celeron
- USB 3.0 Connection
- Allows portability

Future: Toffi Smartglasses

- Embedded processor connected to ToF sensor + Haptic sensor
- Gyroscopic sensor to detect head movement
  - Often built into embedded processor
  - Used to compensate for head tilts, twists
- Small, portable
  - Heads-up display with AR
  - Wireless connection with 5G Tech
  - Powered by rechargeable battery
  - Haptic feedback to improve user experience



Future ToF Technology

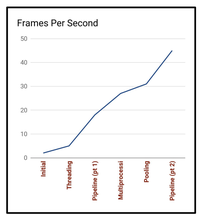
- Future: Usage of Pico monstar (TOF)
  - Smaller, 6m measurement, 352 X 287
- Many smartphone companies such as Samsung, Google, and Apple are using ToF sensors in their new smartphones (ex. Galaxy S10 5G)

RESULTS

How did we test Toffi?

Tracking Frames Per Second

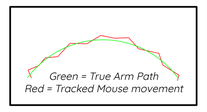
- Went from 2 FPS to 45 FPS
- Chart below shows the increment with each addition



Improvement in FPS with each major modification/addition

Tracking Testing

- Built a **swinging arm** using LEGO Mindstorms EV3
- Preprogrammed to swing left and right (hand move)
  - **Swung 120 degree arc**
- Tracked hand position **detected** by Toffi **compared** to the **actual** movements
- Used difference to smoothen



Our prototype was able to **track motion seamlessly** at **45 FPS and** was **significantly accurate** in detecting **fingertip position!**

CONCLUSION

What is next for Toffi?

Cost + Feasibility

*It is important to note that Toffi, in its debut incarnation, is intended to be **used by embedded systems and computer manufacturers to add touchless interaction with their devices.***

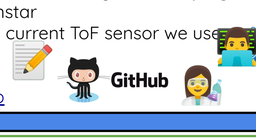
The ToF sensor we are using, **CamBoard Pico Flexx, is $389**. Once a high-enough volume of production is achieved, the per-unit costs **will significantly decrease**. As a proof-point, smartphones, which ship in huge volumes, are able to **absorb** the **cost** of ToF sensors with almost no additional price markup. Additionally, newer **innovations** and **competition** is already in place to decrease the cost of ToF sensors.

Toffi is a **portable, embedded system** that is **power and compute efficient**. There is **zero wear and tear** in comparison to the constant physical presses needed to operate a keyboard and mouse, **resulting** in a **longer life, providing a greener and hassle-free experience.**

Future Improvements

- Detect **additional hand-gestures**
  - Ex. closed fist for click and drag

- **Speed-up** processing
  - From interpreted to **compiled** (Python → C++)
- **Keyboard Typing Capability**
  - Detect **10 fingers**, project positions onto virtual keyboard, track x, y, z to determine **keypresses**
- **Handwriting Capability**
  - Use depth and x, y coordinates to train a light-weight **handwriting AI recognition** model
- **Remote Equipment Maneuvering**
  - Using ToF Camera in a **smartphone**, recognize gestures to **send** controls to internet-connected **remote** computer (ex. surgical robot) ← **5G**
- Create Wearable Prototype
  - Smartglasses attached to haptic sensor, ToF, and connection to **remote** or **local device**
  - Use **AR** to **project** hand/fingers on **display**
- Test PMD Pico Monstar
  - 4x smaller than current ToF sensor we use
- Share/Publicize
  - File a **Patent**
  - Promote Github



Conclusion

Toffi **is revolutionary** 💡 in the way we **interact** with computers 💻. Our solution uses a **time of flight sensor** and **OpenCV** to accurately track the **movement** and **depth change of a fingertip overtime** and **translates** that to mouse movement and clicks 🖱️ **on a computer display**. This working, portable **embedded system** runs in **real time**.

In the future, we plan to **integrate** our idea into a smart-glass or other type of **wearable** 👓, implement **typing** 💻**and handwriting** ✏ **in air,** and add **additional, customizable gestures** for **ease-of-use.**

The **ability to control** a computer **without physical touch** can **prevent the spread of germs** 🦠 and **contamination**, gives **universal accessibility**, and allows people with **disabilities** to **interact** with **devices**📲. **Hospitals** 🏥 , **laboratories** 🔬 , **public** 🏙️**spaces**, offices, and homes -- all **can benefit from Toffi**.

SOURCES

What are the inspirations behind Toffi?

- Health Informatics J. 2019 Dec;25(4):1325-1342. doi: 10.1177/1460458217748342. Epub 2018 Feb 10.
- Forbes, Elliot. "Python Multiprocessing Tutorial." TutorialEdge, 17 Apr. 2015, tutorialedge.net/python/python-multiprocessing-tutorial/.
- "3D Camera Survey - ROS-Industrial." ROS-Industrial, ROS-Industrial, rosindustrial.org/3d-camera-survey. - Accessed December 24, 2019
- Vision Campus, director. What Is Time-of-Flight? YouTube, Basler AG, 31 May 2016, www.youtube.com/watch?v=OMDfQC0m4i4&vl=en. - Accessed October 3, 2019
- Astojilj. "Tutorial: Typing in the Air Using Depth Camera, Chrome*, JavaScript*, and WebGL* Transform Feedback." *Intel® Software*, Intel, 26 June 2017,software.intel.com/en-us/blogs/2017/06/22/tutorial-typing-in-the-air-using-depth-camera-chrome-javascript-and-webgl-transform.
- Fingers Detection Using OpenCV and Python - Izane github.com/Izane/Fingers-Detection-using-OpenCV-and-Python
  - *Used as an introduction to image- extracting OpenCV functions*
- Kolb, Andreas & Barth, Erhardt & Koch, Reinhard & Larsen, Rasmus. (2008). Time-of-Flight Sensors in Computer Graphics. Proc. Eurographics (State Art Re.). 2009. - Accessed August 25, 2019

Python Packages/Libraries

- Pyautogui, numpy, argparse, time, queue, matplotlib, sys, multiprocessing, PIL, cv2, os
- Roypy package
  - ToF interfacing