Real-time responsive input gestures using a standard rgb camera for computer systems.

Noah King
Dept. Computer Science
University of Canterbury
Christchurch, New Zealand
nki38@uclvie.ac.nz

Prof Richard Green
Computer Science and Software Engineering
University of Canterbury
Christchurch, New Zealand
line 4: City, Country
richard.green@canterbury.ac.nz

***Abstract*—This paper proposes an approach to implementing 'natural' computer input gestures in 3d space through real time tracking gestures and recognition. Google's Media Pipe Hands model is used for real time palm tracking, including a single shot detector. A neural network model categorizes gestures and a Kalman filter is used to mitigate failed tracking. This facilitates some common multitouch gestures to be read against a webcam and for a computer system to respond to them accordingly. An output frame showing the users hands is created usingcontour extraction and thresholding. The gesture recognition achieves an overall 86% accuracy.**

Keywords—Gestures, Vision, style, styling, insert (key words)

I. Introduction

Capacitive input devices now augment traditional mouse and keyboard inputs with ubiquity. Both Apple MacOS and Microsoft Windows include support for non-linear input gestures and for kinetic multitouch inputs in their respective environments. Offerings like Meta Horizon OS and Apple Vision OS now incorporate gestures in 3d space for interacting with UI in augmented reality and virtual reality. In both instances, numerous cameras are required, including depth sensing camreas and IR emitters. [2] [1]

This paper therefore proposes a means to implement real time tracking and gesture categorization which can be used to implement real time 1:1 kinetic input gestures as well as static input gestures using only commodity a single commodity RGB camera like those built into most laptops, tablets and mobile computers.

II. Background

There have been multiple approaches to facilitate this model of interaction:

A) Custom Sensors

In 1990, Quam[4] proposed a 'DataGlove' instrument with an embedded microcontroller which communicated over a serial link with a desktop computer. Quam explored computer inputs by translating American Sign Language from inputs from the worn gloves. Quam used the angles between each landmark[1], collected by fibre-optic joint angle sensors as datapoints for categorizing gestures.

B) Segmentation

Wang[5] proposes a Method which uses a coloured worn glove and a commodity RGB camera. Colours are separated via colour segmentation from the background then compared against a database of hand poses. The 10 nearest neighbours are combined through a Gaussian Kernel co create the final representation of the hand.

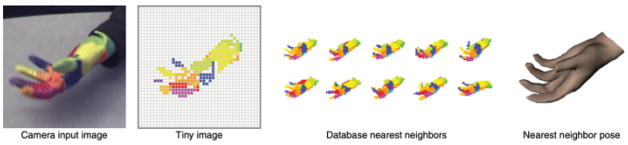

Figure 1 Coloured Glove, Segmentation, Neighbours, Interpolated Form

C) Image Subtraction

Hickman[6] proposes a method for determining the presence of a hand through background subtraction, Hickman determines if the hand exists and the state of the hand by counting the intersections of the thresholded edges of the binary subtraction of a calibrated frame and the frame containing the hand. This approach has the limitation of requiring a background image to be calibrated and is intolerant of noise. It is also not suitable for gathering depth information about the hands.

D) Neural Networks

Yu[7] proposed a method which used a trained convolutional neural network to detect five unique gestures for use in public displays. This method produced real time recognition but was susceptible to false positive readings due to clashes with skin tone and background colours. This method required substantial pre-processing and references a global threshold, so is not adaptable to all lighting conditions.

Lowe[8] proposes a Convolutional neural network approach to categorizing gesture (and therefore the presence of hands) based on InceptionV3; where the input dataset is a set of images of hands with no morphological transformations applied. With 500 images per gesture, Lowe achieved accuracies ranging between 0.5-0.85. This approach requires substantial training data and is susceptible to background noise and features.

InceptionV3 is widely used in image recognition[9] as an image classification model; however it's application without substantial training data means that it is ill suited to non-static gestures.

Zhang Proposes a method for vision based hand pose estimation through the use of a hand landmark model and palm detector [11] collectively called *Media Pipe.*

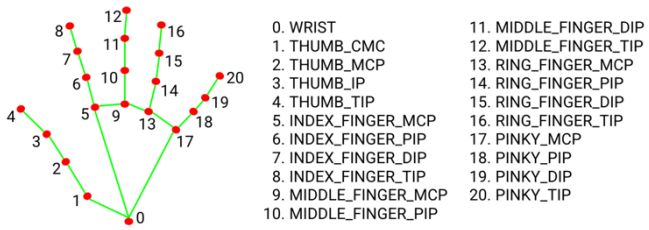

Figure 2: Media Pipe's 21 Key points (Landmarks) [10]

Which uses deep learning models and a single shot detector to identify and map hands and their key landmarks.[10]

Hurst [12] proposes a Neural Network for classification using a pretrained model for nine different categorical gestures using landmark information from the MediaPipe Hands model[10] for landmark 3vectors. Hursts' approach achieves overall accuracies of 92% for gesture categorization. However, this approach is limited by the binary state of individual gestures and as the training data is the set of 21 3 vectors it is intolerant of unique hand shapes and requires significant training data to be applied to a general audience.

E) Custom Hardware and Depth Maps

Hoshino proposes a method for generating smoothed hand models from depth cameras[[13]. Due to the nature of the depth cameras, occluded 3d information is not available, this solution passes three unique filters; 2 instances of Gaussian 5x5 Filters follow pixel median filters and moving average filters to remove noise. This approach has significant computational overhead [13]

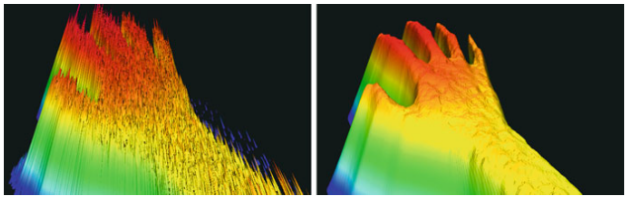

Figure 3 Depth Map Hand Model [13]

Hoshino notes that this method restricts gestures to be observable in the viewable plane.

These solutions have largely become overshadowed by the uniquity of RGB cameras and their relative computational overhead.

III. Solution / Method

A. Overview

The proposed method uses four main components to achieve 'smooth' computer inputs from a solitary RGB camera. The method generates points in 2d space which can be later used for 1:1 input gestures. The method also facilitates ancillary static gestures.

The data points, similar to Hurst[12] and Gelin[14], are landmarks provided by the Media Pipe hands model [10]

There are four core components to this proposal.

- The Media Pipe Hands palm detector and landmark model
- A Neural Network which categorizes gestures.

- A state machine which defines input sequences.
- An interface model for a specific client.

The method targets 60hz updates with a 30hz polling rate.

The hardware which was used for the development and evaluation of performance is as follows:

Table 1: Development Hardware and Library

OS	MacOS 14.2.1
Processor	Apple M2 8-core
GPU	Apple M2 8-core
NPU	16- core Apple Neural Engine
RAM	8gb (unified)
Camera	1080p FaceTime HD @ 60fps
Language	Python 3.9.6
Libraries	OpenCV-py 4.9.0.80, tensorflow 2.16.1 Pyautogui 0.9.54

B. *PreProcessing*

The live webcam feed is first scaled to 640x480 to improve the speed of the MediaPipe landmark model. The image is flipped about the vertical axis to mirror the input. Per the requirements for Media Pipe Hands detection[12]; the frame is converted from RGB to BGR.

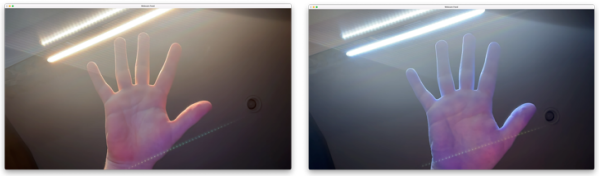

Figure 4 Colour Channel Conversion RGB to BGR

C. *Hand Point Detection*

Firstly, the relative location of a single palm is detected and a bounding box defined using a sub-millisecond single shot similar to BlazeFace [15] which is able to achieve 200-300 fps on a mobile processor[10]. Estimating bounding boxes for rigid objects like palms or fists are trivial in comparison to detecting hands with articulated digits. When BlazePalm does detect a palm, a flag is set. No predictions are made by other models where no palm is detected.

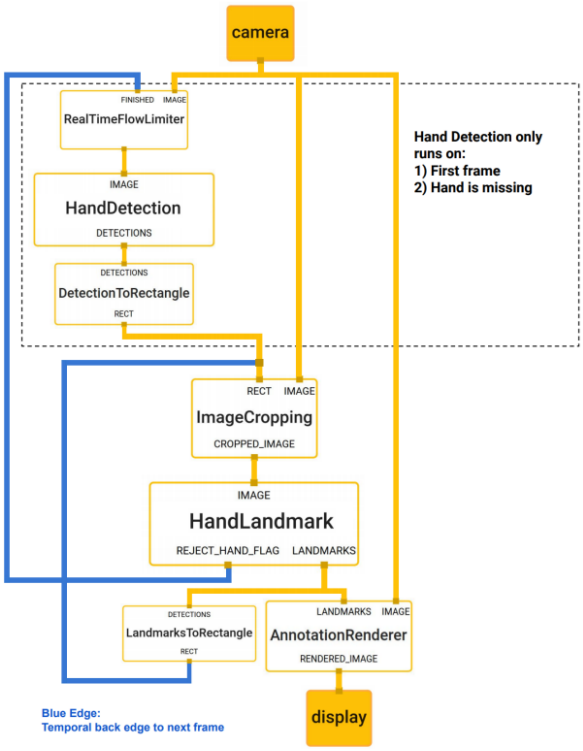

Figure 5 MediaPipe Hand Landmark Pipeline [10]

If palm detection has generated a bounding box; the Hand Landmark Model is run over the entire image, providing 21 landmarks in *relative* 3d space; that is x,y and relative depth [13] The pipeline is seen in *Figure 6.*

The Hand Landmark Model was trained using the following labelled input images[10]

- *6000 diverse in-the-wild images*
- *10'000 images from 30 people*
- *A broad synthetic dataset*

Media Pipe does track points that are obscured or offscreen.

D. *Reliability of Tracking*

As Media Pipe Landmarks is not able to provide cohesive or perfect tracking; an unscented Kalman filter is applied to each frame, based on a dot drawn on the centre of the originating gesture, not displayed to the user. Short losses in Landmark tracking are therefore not noticed.

E. *Gestures*

The state machine dictates the current gesture and thereby the current input. Transitions between states are defined by the gesture detected through the recognizer model in *G.*

There are 6 key gestures, in the order in *Figure 6* they are:

1. Idle
2. Pinch Without Click
3. Pinch With Click
4. Two Finger
5. Point
6. Spread

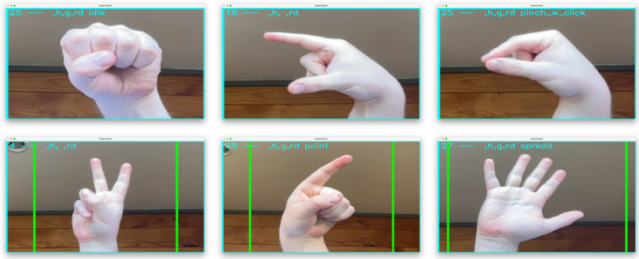

Figure 6 The Six Gestures

A pretrained Neural Network categorizes 5 of these; however the distinction of click in *pinch* is defined by the relative Euclidean distance between the tip of the user's index finger and tip of their thumb over a subset of the averaged frames to provide quick response and mitigate unintended tracking when transitioning between gestures.

F. *State Machine*

The State Machine for the device interface allows multi-input and restricts switching to only valid flows of gestures. A small sample of the state machine is viewable in *Figure 3,* Only showing one particular set of transitions from the idle state.

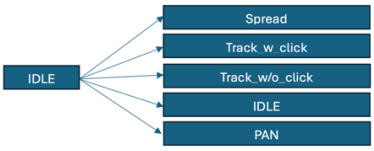

Figure 7 StateMachine Sample, Without Transition Labels

The state machine is an incomplete graph whose reduced state contains 34 transitions.

The machine is used with the interface model to relay common inputs to the computer. Interfaces are implemented for macOS and Windows Operating systems.

G. *Gesture Recognition*

Similar to Quam[2] recognition of gestures is performed by analysing the angles between landmarks. However, similar to Hurst[12] and Gelin[14] a NN model is trained to provide categorization.

The model is trained to recognize 5 key gestures using a dataset of 300-1000 rows of 120 datapoints per gesture and 1 column for the output.

The model's hidden layer comprises of 64 neurons using the ReLU activation function, and an output layer employing softmax activation for categorization across each of the five gestures.

ReLU mitigates the likely issue of vanishing gradient in our data as the differences in many of our angles across 120d is small *relatively*.[16][17]

For optimization, the model utilizes the *Adam optimizer* [18] , along with categorical cross-entropy loss as they are both *relatively* efficient for higher dimension datasets, in our case, 120 column on a model that is trained on a laptop.[16][18]

Standard relative accuracy is used as the evaluation metric during training iterations.

The model is trained over 10 epochs.

H. *Training*

Training data was created in two ways:
1. A live webcam feed recorded each of the gestures on keypress, Allowing hundreds of datapoints to be generated per minute.

2 A live webcam feed with a random stretch transformation applied was used to generate datapoints in real time and create pseudo hand-shapes.

In each instance, Media Pipe Hands generated a list of *landmarks* as 3vectors and the relative radian angles between each of 15 of the 21 landmarks was recorded in 120 feature rows and an output row denoting the gesture.

I. *Optimization for 60hz input*

To achieve the 60hz perceptible update rate the main thread for the mouse position update occurs at 60hz however positioning is updated at up to 30 times per second. Multithreading is used for the three core functions; Media Pipe Hands is run on the main thread.

J. *Smoothing*

The 3space vectors of locations for each of the 15 landmarks measured are also abstracted as their average over 8 samples. Pinch and pan translation are based on this average as is cursor acceleration.
This reduces jitter significantly at the expense of introducing latency of the 8 multiples of the update time for recognizing a new gesture.

By recording separately with a high speed setting, we obtain the following rough estimations of time These are within the 200ms[19] upper bound and therefore acceptable.

Table 2: Observed Response Times

Deque Cycle Time	164ms	8 samples
Recognize Click	130ms	8 samples

To reduce cursor jitter a sigmoid function is used as a an acceleration curve.

An unscented Kalman filter is used to detect the movement of the finger; that is; the location of the tip of the user's finger in 2space is obtained from MediaPipe's [10] and drawn onto the frame. An unscented Kalman filter is then used to track the point.
Tracking in this way allows partial movements offscreen; such as; when a movement begins in the viewable area and returns to the viewable area, through obstruction or to account for lost tracking

K. *Gestures to HCI Inputs*

This same approach is used for calculating the distance traversed in the pan and drag gestures; in these instances points are given locations based on the average of the *index tip, ring tip* and *thumb tip and index tip* respectively.

L. *Interface*

Point translates the cursor using one of two user defined methods:

1. The average position of the tip of the index finger is calculated on each frame and the gradient over those observations is applied as an acceleration in XY space to the cursor model
2. A vector is drawn from the average of the of the landmarks along the index finger and scaled to the resolution of the client display.

Pinch creates a gradient and relative positions of the tip of the index finger and thumb.
Spread calls the relevant window overview
2 Finger works in the same way as approach 1 of cursor translation, using the average location of the two fingers for panning the image in x,y.
Idle reverts all states
Both options are available to the user on to toggle on a keystroke

M. *User Interface*

Finally; the user interface is displayed: A silhouette of the users hand is created applying in order, a 5x5 Gaussian blur, binary thresholding and contour detection.

Figure 8 Sample Cropped Output frame to indicate Hand presenc

The output frame is cropped to a buffered bounding box around the *landmarks*;

If there is no hand present, the entire frame is displayed to indicate to the user that no gesture is detected.

RESULTS

A. *Test Results*

These results show that the proposed approach can accurately categorize gestures using a pretrained neural network within acceptable accuracy boundaries. The proposed approach can also smoothly interface with modern computer operating systems and does produce data points in real time that could be mapped to 1:1 kinetic gesture inputs.

The following accuracy results of the trained model are made against 3x 15s recordings of only one gesture and do not include transitions between gestures.

Table 3: Summary of Recognition Accuracy%

Gesture	Average Certainty %
Pinch	0.5059232
Point	0.92377746
Idle	0.5048256
2-Finger	0.9964953
Spread	0.99726105

The confidence of the *Pinch* gesture is quite poor,

as is idle, the overall accuracy is below 80% for pinch.

Table 4: Classification Accuracy

Gesture	Classification Accuracy
Pinch	77.27
Point	81.02
Idle	93.22
2-Finger	81.25
Spread	99.15
Average	86.382

B. *Comparison with other solutions*

Comparing to Hurst[12] and Yu[7] for

Table 5: Comparison to Previous Results

	Yu (<1m Detection)	Hurst	Result
Accuracy	90%	80%	86.362%

The volume and dimensionality of the training data is likely responsible for the improvement in gesture recognition over Hurst and the single shot detector is likely responsible for the overall accuracy improvement over YU, but the small volume of training data is likely responsible for YU's better <1m results.

C. *Further Research*

To further extend the practicality of this an on display overlay would be greatly practical. A core limitation of translating cursor inputs is the precision differential between a finger and a cursor. Apple Inc include a cursor with dynamic shapes in iPadOS [22] to mitigate this, for example.

The datapoints created by the *pinch* and *zoom* gestures can be used by an updated interface model to accurately input those gestures with a 1:1 response, requiring implementation of a custom driver or abstraction onto a generic driver on either major system.

REFERENCES

[1] Meta. "Introducing Our Open Mixed Reality Ecosystem" Internet: https://about.fb.com/news/2024/04/introducing-our-open-mixed-reality-ecosystem/, Apr, 24, 2024 [Apr, 25, 2024].

[2] Apple. "Documentation/visionOS" Internet: https://developer.apple.com/documentation/visionos [Apr,23,2024]

[3] UltraLeap "TouchFree User Manual" Internet https://docs.ultraleap.com/TouchFree/touchfree-user-manual/index.html [Apr 20,2024]

[4] x Quam, D.L.: Gesture Recognition with a DataGlove. Proc. 1990 IEEE National Aerospace and Electronics Conf., 2, (1990) 755–760

[5] R. Y. Wang and J. Popovic, "Real-Time Hand-Trackingwith a Color Glove," ACM Transactions on Graphics28, Massachusetts, 2009.

[6] Henry Hickman and Richard Green, "Playing Tetris Using Simple Hand Gestures", Computer Vision Lab, University of Canterbury, Tech. Rep., May 2021

[7] Shun Lyu and Richard Green, "Real-time Low-cost Hand Detection System for Interaction with Public Displays", Computer Vision Lab, University of Canterbury, Tech. Rep., May 2019

[8] Joshua Lowe and Richard Green, "Human-technology interaction hrough hand gestures and interpreting camera inputusing labelled images and a pre-trained neural network", Computer Vision Lab, University of Canterbury, Tech. Rep., May 2019

[9] Google: "Advanced Guide to Inception v3" Internet: https://cloud.google.com/tpu/docs/inception-v3-advanced [Apr 1 2024]

[10] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L.Chang and M. Grundmann, "MediaPipe Hands: On-device Real-timeHand Tracking," Google Research, Mountain View, 2020

[11] Valentin Bazarevsky, Yury Kartynnik, Andrey Vakunov, KarthikRaveendran, and Matthias Grundmann. Blazeface: Sub-millisecondneural face detection on mobile gpus July 2019

[12] Josh Hurst and Richard Green, "SmartApplianceControl Using Hand GestureRecognition", Computer Vision Lab, University of Canterbury, Tech. Rep., May 2023

[13] Hoshino, K. (2015). Hand Gesture Interface for Entertainment Games. In: Nakatsu, R., Rauterberg, M., Ciancarini, P. (eds) Handbook of Digital Games and Entertainment Technologies. Springer, Singapore. https://doi.org/10.1007/978-981-4560-52-8_47-1

[14] Hoshino, K. (2015). Hand Gesture Interface for Entertainment Games. In: Nakatsu, R., Rauterberg, M., Ciancarini, P. (eds) Handbook of Digital Games and Entertainment Technologies. Springer, Singapore. https://doi.org/10.1007/978-981-4560-52-8_47-1

[15] Vakunov, Karthik Raveendran, and M. Grundmann. Blazeface: Sub-millisecondneural face detection on mobile gpus, 2019 https://arxiv.org/abs/1907.05047

[16] A.Agarap Deep Learning using Rectified Linear Units (ReLU) 2018 https://arxiv.org/abs/1803.08375

[17] Google 'TensorFlow Documentation tf.keras.layers.ReLU' https://www.tensorflow.org/api_docs/python/tf/keras/layers/ReLU [Apr. 20, 2024]

[18] Diederik P.Kingma J.ba Adam: A Method for Stochastic Optimization, 30. Jane 2017 [Apr. 11, 2024]

[19] Miller,R.B Response time in man-computer conversational transactions. In: Proceedings of the AFIPS 1968, December 9-11, 1968, pp. 267-277. ACM, New York (1968)

[20] Apple Inc. 'Pointing Devices' https://developer.apple.com/design/human-interface-guidelines/pointing-devices [May. 01, 24]