4. J. Wang, Y. Yang, T. Wang, R.S. Sherratt. Big data service architecture: a survey // Journal of Internet. – 2020. – V. 21. – №. 2. – PP. 393–405. – URL: https://jit.ndhu.edu.tw/article/viewFile/2261/2274.

5. E. Mehmood, T. Anees. Challenges and solutions for processing real-time big data stream: a systematic literature review // IEEE Access. – 2020. – V. 8. – PP. 123–143. – URL: https://ieeexplore.ieee.org/iel7/6287639/ 6514899/09126812.pdf.

УДК 681.178.9

Н.В. Гурько, студ.; П.А. Лазарева, доц.
(КНИТУ-КАИ им. А. Н. Туполева, г. Казань, Россия)

СИСТЕМА БЕСКОНТАКТНОГО УПРАВЛЕНИЯ ДИСПЛЕЕМ НА ОСНОВЕ КАМЕРЫ ГЛУБИННОГО ИЗОБРАЖЕНИЯ

В современном обществе всё большую популярность приобретают системы, использующие в основе принципа своей работы методы обработки изображений при помощи нейросетей. Частным случаем подобных систем считается определение положения рук на снимке, дающее возможность частично контролировать функционал любого устройства за счёт заранее указанных жестов. На основе подобного определения, при использовании камеры глубины можно реализовать механизм, схожий с привычным управлением любого сенсорного устройства, но без необходимости в касаниях.

Камеры глубины (depth cameras) – разновидность камер, позволяющих определить расстояние от устройства до наблюдаемых объектов сцены и сопоставить карту глубины с цветным RGB изображением (перенос точки из depth изображения на RGB). Реализация системы была осуществлена при помощи сенсора Kinect v1 и библиотеки OpenNI2, использовавшейся для получения и передачи данных с камеры, на языке программирования Python. Данная камера уже использовалась для создания бесконтактного управления консолями Xbox, однако механизм имел ряд недостатков, в том числе необходимость нахождения человека в полный рост на расстоянии не менее 3м от самой камеры и невозможность удобного использования её вне экосистемы самой игровой консоли.

Распознавание положения рук и отслеживание движений было реализовано с использованием библиотек OpenCV и MediaPipe, дающих возможность получить “скелет” руки через опорные точки, даже если часть точек будет закрыта или спрятана (рис. 1).

Рисунок 1 – Отображение скелета конечности при использовании MediaPipe

Каждая характерная точка руки на данном изображении имеет свой номер (рис. 2).

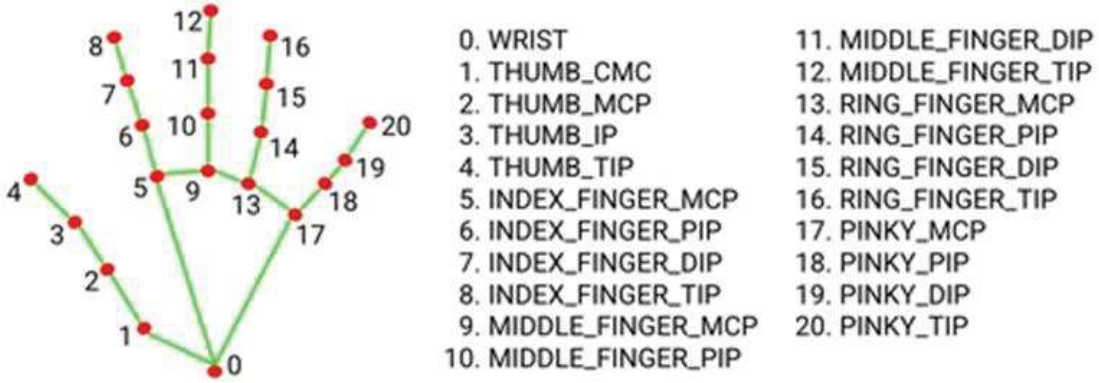

Рисунок 2 – Расположение ключевых точек в MediaPipe при обработке жестов рук

Минусом использования данной библиотеки считается то, что первоначальный стабильный видеопоток в 20 кадров в секунду может снижаться до 15. Также, хотя MediaPipe и содержит встроенное определение 7 базовых жестов, но использование данной функции дополнительно снижает производительность на 1-2 кадра, поэтому для определения конкретных жестов была создана обособленная нейросеть с использованием библиотеки PyTorch, состоящая из трёх линейных слоёв и двух функций активации LeakyRelu между ними. MediaPipe обрабатывает изображение, полученное от OpenCV, и определяет координаты каждой точки в виде массива из двух элементов [x, y], каждый из которых определяется как отношение координаты данной точки в соответствующей оси на самом изображении к общему количеству точек на ней.

Если соединить некоторые заданные точки и провести отрезки 0-1, 0-4, 0-5, 0-8, 0-9, 0-12,0-13, 0-16, 0-17, 0-20, 48-, 8-12, 12-16, 16-20, то можно узнать длины данных отрезков и создать массив из 14 элементов, куда будут вноситься эти длины.

Данный массив подаётся на нейронную сеть, заранее обученную на распознавание некоторого набора жестов, и в результате можно с высокой точностью получить класс заданного жеста.

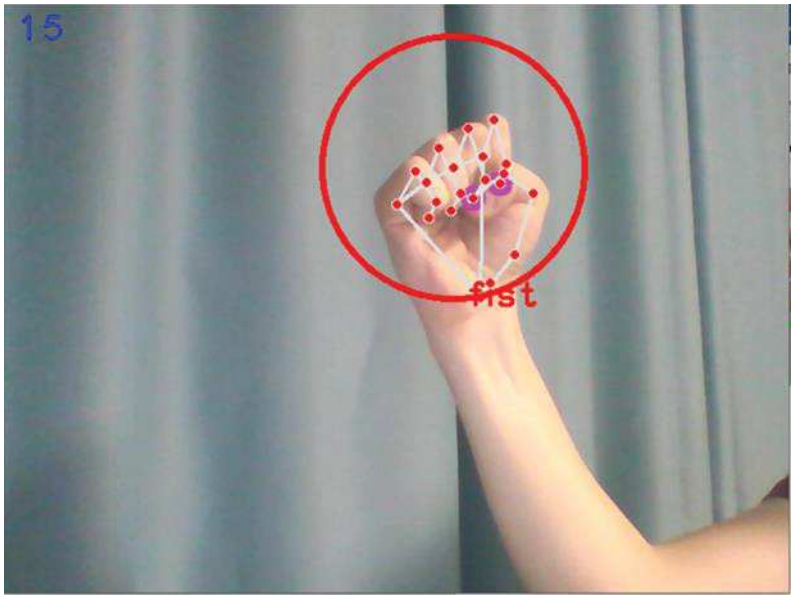

Рисунок 3 – Пример распознавания жестов рук

Учитывая искажения связанные с тем, что не существует универсального метода переноса пикселей с RGB на Depth изображение, можно определить трёхмерные координаты центра руки (точки 9) в пространстве при помощи метода openni2.convert_depth_to_world(), где ось Z сонаправлена с камерой, ось X направлена по горизонтали вправо, а ось Y вертикально вверх.

Если расположить Kinect под дисплеем, на котором идёт отображение картинки, таким образом, чтобы камера охватывала весь экран своим полем зрения и находилась не менее чем на 0,5м от его самой нижней точки, а вектор X сенсора был коллинеарным к вектору на линии ширины экрана, тогда вектор Z будет противонаправленным к вектору, имеющему начало в левой верхней точке экрана и направленному вниз по высоте.

Получив заранее разрешение экрана и указав рукой над камерой две диагонально противоположные точки экрана, можно “симулировать” движение, аналогичное движению компьютерной мыши по экрану, где она находится на пикселе, соответствующем координатам:

$$X_{\text{экран}} = \frac{X_1 - X_{\text{тек.}}}{X_1 - X_2} * \text{width};$$

$$Y_{\text{экран}} = \frac{Z_1 - Z_{\text{тек.}}}{Z_1 - Z_2} * \text{height}.$$

В данных формулах X_1 и Z_1 – координаты левой верхней точки,

X_2 и Z_2 – координаты правой нижней точки, $X_{тек}$ и $Z_{тек}$ – координаты точки на руке в данный момент времени. Координата Y в системе камеры может характеризовать расстояние от точки на руке до самого дисплея. Назначим некоторое пороговое значение Y, при переходе через которое будет активироваться некоторая команда. При помощи модуля *mouse* можно задать при данном действии выполнение команды "Клик левой кнопкой мыши", а через нейронную модель на PyTorch – другие действия, выполняемые мышью.

Помимо всего перечисленного стоит отметить дополнительные моменты по настройке системы для бесконтактного управления:

1) камера может располагаться не только под дисплеем, но и над ним, и в стороне, параллельно краю дисплея, но для этого будет необходимо поменять местами переменные или умножить на –1 правые части в формулах для $X_{экран}$ и $Y_{экран}$.

2) расположение сенсора на некотором расстоянии параллельно экрану и перед ним не являются обязательными условиями, но стоит учитывать, что при любых условиях управление дисплеем происходит относительно камеры глубины. Чем меньше угол между осью X сенсора и шириной экрана, тем точнее процесс контроля.

3) расположение крайних точек прямо на концах сенсора также не является обязательным условием. Пользователь имеет возможность настроить под себя своё рабочее пространство, выбрав лишь частичную область на дисплее, через которую будет вестись контроль над всем устройством, или же область, не находящуюся рядом с дисплеем.

Таким образом, в данной работе получена работоспособная система бесконтактного управления дисплеем на основе камеры глубины Kinect и нейросетевых алгоритмов.

ЛИТЕРАТУРА

1. Официальный сайт разработчиков библиотеки MediaPipe – URL: https://ai.google.dev/edge/mediapipe/solutions/studio (дата обращения 18.01.2025).

2. Официальный сайт разработчиков библиотеки OpenNI2 – URL: https://structure.io/openni/ (дата обращения 18.01.2025).

3. Официальный сайт разработчиков библиотеки OpenCV – URL: https://opencv.org (дата обращения 18.01.2025).