Вестник КРАУНЦ. Физ.-мат. науки. 2019. Т. 27. № 2. С. 55-73. ISSN 2079-6641

DOI: 10.26117/2079-6641-2019-27-2-55-73

ИНФОРМАЦИОННЫЕ И ВЫЧИСЛИТЕЛЬНЫЕ ТЕХНОЛОГИИ

УДК 004.42

# АППАРАТНО-ПРОГРАММНЫЙ КОМПЛЕКС АУГМЕНТАТИВНОЙ СИСТЕМЫ КОММУНИКАЦИИ НА ОСНОВЕ ТЕХНОЛОГИИ EYETRACKING

**Ю. В. Грушко**

Камчатский государственный университет имени Витуса Беринга, 683032,
г. Петропавловск-Камчатский, ул. Пограничная, 4
E-mail: thekidsshow96@gmail.com

В работе представлено описание разработанного АПК и реализованного алгоритма айтрекинга (с использованием сверточных нейронных сетей и методов математической морфологии), позволяющего людям, страдающим нарушениями двигательных функций и нервнопаралитическими синдромами, осуществлять процесс коммуникации с внешним миром, посредством управления компьютером и печати текста движением глаз.

*Ключевые слова: айтрекинг, окулография, алгоритм, АПК, сверточные нейронные сети, морфологический анализ, контурный анализ, каскадные классификаторы, Хаар-каскад, метод Виолы-Джонса, машинное зрение, OpenCV, dnn, deep neural network, микроконтроллеры, эрозия, дилатация, сверточное ядро, бинаризация, алгоритмы построения выпуклых оболочек, алгоритм Джарвиса, YOLO.*



## Введение

Ежегодно инсульт переносят более 500 тыс. россиян, причем каждые 1.5 минуты он поражает новую жертву. Инсульт занимает второе место в структуре общей смертности населения. Он является и лидирующей причиной инвалидности населения. По данным на 2018-2019 г. Национального регистра инсульта, трети пациентов, перенесших это заболевание, необходима посторонняя помощь в уходе за собой, 20% не могут самостоятельно ходить и лишь каждый пятый из выживших может вернуться к прежней работе [1].

После перенесения инсульта наиболее характерным является парез или паралич конечностей, приступы эпилепсии, нарушение координации. В тяжелых случаях у больных (в дальнейшем Пациент) наблюдается:

- Идеомоторная апраксия (МКБ - 10: код - R48.2) – расстройство способности выполнять последовательные действия при сохранении необходимого объема сенсорных и двигательных функций.

- Синдром запертого человека (синдром деафферентации) он же полный паралич (МКБ – 10: код G83.5) проявляется полной потерей речи (афазия), параличом (за исключением мышц глаз) при полной сохранности сознания и чувствительности.

- Миастения Гравис (МКБ – 10: код G70.2) – аутоиммунное нервно-мышечное заболевание, характеризующееся патологически быстрой утомляемостью поперечнополосатых мышц.

При длительном сохранении вышеперечисленных синдромов одной из основных проблем является сложность или невозможность коммуникации больных с окружающим миром. Например, в случае синдрома «запертого человека» пациенты понимают обращенную к ним речь, однако коммуникация возможна только с помощью движений глаз или моргания.

При детальном исследовании вышеперечисленных синдромов, выяснилась возможность использования альтернативной аугментативной коммуникации (ААК) – в частности использование методов окулографии и айтрекинга.

**Альтернативная аугментативная коммуникация** – процесс установления через посредство разнообразных знаков связей с людьми, не способными полноценно общаться на вербальном уровне по причине невозможности полноценного воспроизведения или же восприятия речи[2].

Разработанный аппаратно-программный комплекс, позволит увеличить качество коммуникации пациентов с внешним миром (родственники, врачи), а именно:

- Затрачивать минимальные физические усилия.

- Увеличить скорость коммуникации.

- Давать информативный/развернутый ответ медицинскому персоналу.

- Задавать свои вопросы.

Аппаратно-программный комплекс "ELIXIR"позволит людям, страдающим нарушениями двигательных функций, нервнопаралитическими синдромами и другими болезнями ЦНС, препятствующими вербальным и невербальным методам коммуникации:

- управлять компьютером движением глаз (управление курсором осуществляется с помощью движения глаз, щелчок производится морганием);

- печатать текст движением глаз с помощью разработанной виртуальной клавиатуры, с возможностью создания пользовательских текстов-пресетов для быстрого набора сообщений.

- отправлять экстренные сообщения, набранные на виртуальной клавиатуре, родственникам или медицинскому персоналу (CRM модуль);

- озвучивать напечатанный текст в чистую речь, используя модули синтеза речи (Text-To-Speech модули).

## Разработка алгоритма трекинга зрачка

Для разраотки АПК аугментативной системы коммуникации необходимо было изучить основные методы окулографии.

**Окулография (отслеживание глаз, трекинг глаз, айтрекинг)** - определение координат взора («точки пересечения оптической оси глазного яблока и плоскости наблюдаемого объекта или экрана, на котором предъявляется некоторый зрительный раздражитель»).

**Отслеживатель глаз** – устройство, используемое для определения ориентации оптической оси глазного яблока в пространстве (то есть для отслеживания глаз). Отслеживатели глаз используются в исследованиях зрительной системы, психологии, когнитивной лингвистике. Для отслеживания глаз используют несколько методов. Самый популярный – покадровый анализ видеосъемки глаза, также используются контактные методы, такие как электроокулография [3].

В рамках разработки проекта был использован покадровый анализ видеосъемки – камера снимает один глаз и регистрирует его движения, пока испытуемый рассматривает визуальный стимул (например, экранную клавиатуру). Отслеживатель глаз использует контраст между зрачком и радужной оболочкой, который возникает при инфракрасной подсветке. Кроме того, анализируется положение блика инфракрасной подсветки, благодаря чему становится возможным определить ориентацию оптической оси глазного яблока (сравнивается положение зрачка пациента относительно блика, создаваемого на склере, если центр зрачка совпадает с бликом – оптическая ось глаза находится перпендикулярно зрительному раздражителю – монитору).

По использованию подсветки глазного яблока, айтрекеры делятся на:

- системы, основанные на методе яркого зрачка;
- системы, основанные на методе темного зрачка.

Их разница заключается в расположении источника подсветки относительно камеры:

- Метод темного зрачка - источник подсветки сдвинут относительно оптической оси камеры, зрачок становится черным, поскольку вторичное отражение от сетчатки не поступает в камеру.
- Метод яркого зрачка - подсветка расположена параллельно оптической оси камеры, глаз работает как вторичный отражатель света, который поступает от подсвети и отражается от сетчатки, создавая эффект яркого зрачка, аналогичный эффекту красных глаз в фотографии.

Эффект яркого зрачка позволяет вести айтрекинг в независимости от цвета радужной оболочки испытуемого. Это также способствует преодолению влияния тёмной глазной туши и ресниц, частично закрывающих зрачок. Это также позволяет проводить отслеживания глаз при световых условиях от полной темноты до высокой освещенности, однако техники яркого глаза не эффективны для отслеживания глаз в условиях улицы, вследствие наличия дополнительных источников инфракрасного излучения.

Разработанный айтрекер использует метод темного зрачка, так как инфракрасная подсветка, состоящая из 4х инфракрасных диодов, мощностью 0.5 Вт и длиной

излучения 960нм, сдвинута относительно оптической оси камеры и располагается по периметру камеры (рис. 1). Отслеживатель глаз, разработанный в рамках ВКРм, работает на скоростях 15 кадров в секунду. Это необходимо для обеспечения регистрации 100% движений глаз.

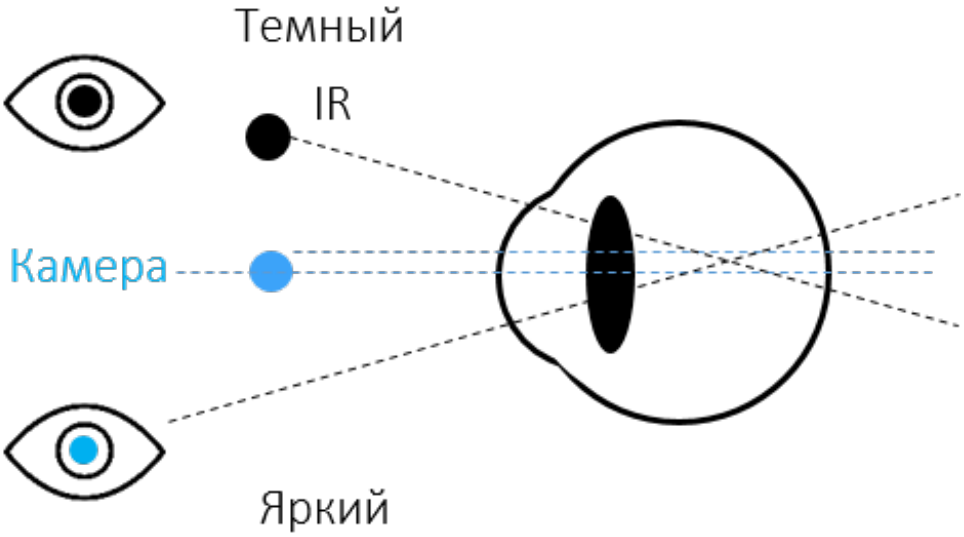


Рис. 1. Сравнение методов темного и яркого зрачка

В качестве устройства трекинга зрачка был выбран одноплатынй микрокомпьютер Raspberry Pi 3 model B.

Технические характеристики Raspberry Pi 3: процессор Broadcom ARM Cortex – A53 с тактовой частотой 1,4 ГГц, 4 ядра (+ графическое ядро с поддержкой OpenGL, аппаратного ускорения и FullHD-видео и DSP-ядро), оперативную память 1 Гб, HDMI, CSI, DSI разъёмы, аудиоразъем 3.5, 4 USB, Ethernet, Wi-Fi 802.11ac и Bluetooth 4.2.

Для разработки алгоритма трекинга зрачка использовалась библиотека машинного зрения OpenCV 4.1, модули глубинного обучения DNN и нейрофреймворк Darknet. В качестве основного языка был выбран Python 3.

**Ниже описан реализованный алгоритм айтрекинга, позволяющий определять позицию глаза и зрачка с точностью до 98-99%, погрешность алгоритма составляет 5-10 пикселей (при разрешении съемки 640 x 480)**.

**Этап №1**

Для использования камеры подключенной через CSI-интерфейс его необходимо включить, используя команду *sudonanoraspi – config* и в разделе interfaces отметить P1 Camera. Далее необходимо добавить камеру в список подключенных модулей к Raspberry командой: *sudonano/etc/modules*, после чего в конец файла дописать: *modprobebcm*2835 – *v*4*l*2. После чего получаем из видеопотока кадры (640 x 480) и переводим каж-дый полученный кадр в градации серого используя метод *cv*2.*cvtColor* :: *cvCV_BGR*2*GRAY*.

**Этап №2**

Дале необходимо было вычислить ROI (region of interests) глаза или его части в кадре (при этом получить кортеж из четырех элементов: $x, y, width, height$). На ранних этапах разработки алгоритма айтрекинга, в качестве детектора объектов (в нашем случае глаза), мы использовали Хаар-классификатор.

Так как классификатор потребляет большую вычислительную мощность, максимально количество кадров на выходе при запуске всего алгоритма (с последующими шагами) было равным 1-3fps (запуск на Raspberry Pi 3), что неприемлемо для данного рода задач. Также хотелось бы отметить, что точность распознавания с использованием классификатора существенно ниже, в сравнении с глубоким обучением (deep learning). Поэтому, было принято решение перейти на использование сверточных нейронных сетей.

Сверточная нейронная сеть (convolutional neural network, CNN) - использует некоторые особенности зрительной коры, в которой были открыты так называемые простые клетки, реагирующие на прямые линии под разными углами, и сложные клетки, реакция которых связана с активацией определённого набора простых клеток [4].

После сравнения и тестирования СНС сетей из набора: SSD mobilenet, YOLO, Faster R-CNN была выбрана архитектура YOLO. Тест производительности протестированных нейросетей с использованием фреймворка OpenCV 4.1 используя модуль глубинного обучения dnn (при обучении и тестировании использовалась машина: CPU: Intel Core i7, 6Гб, тактовая частота видеокарта NVIDIA GeForce GT 740M, 384 CUDA-ядер, скорость передачи данных 1.80 Гбит/с, тактовая частота 810 МГц не в boost, 2Гб DDR3) представлена в таблице 1.

*Таблица 1*

**Сравнение представленных алгоритмов СНС**

| Метод | Обучающая выборка | Средняя точность (mAP) | FPS |
|---|---|---|---|
| Fast R-CNN | COCO dataset | 69.7 | 0.6 |
| Tiny YOLOv3 | COCO dataset | 68.9 | 152 |
| YOLOv3 | COCO dataset | 80.9 | 68 |
| SSD mobilenet | COCO dataset | 77.5 | 19 |

Из данных таблицы 1 можно делать вывод, что наиболее оптимальным вариантом (Как в точности, так в скорости вычислений) является архитектура Tiny Yolov3, при условии ее использования в режиме реального времени.

Модель Tiny YOLO имеет 9 сверточных слоя и 3 полносвязных слоя (полносвязные многослойные перцептроны). За счет меньшего количества параметров нейросети сокращается количество используемой памяти GPU, что дает возможность использовать алгоритм на встраиваемых системах, например на Raspberry Pi 3, NanoPi, NVIDIA Jetson TX2. При этом точность mAP на VOC 2007 для Tiny YOLO составляет 68.9%.

Для обучения нейросети использовался нейрофремворк darknet написанный на языке C и имеющий поддержку технологии CUDA.

В таблице 2, представлен список параметров, используемых в процессе обучения нейросети.

*Таблица 2*

**Параметры, используемые при обучении СНС Tiny Yolo v3**

| Параметр | Описание | Значение |
|---|---|---|
| maxbatches | Максимальное число итераций/эпох на всю тенировку нейросети | 4000 |
| batch | Количество картинок, используемых в процессе тренировки на каждой итерации/эпохе | 24 |
| classes | Количество тренируемых классов | 1 |
| learning rate | Скорость обучения или размер шага в машинном обучении | 0.001 |
| train | Количество файлов в тренировочном наборе | 270 |
| valid | Количество файлов в тестовом датасете (тестовый датасет не участвует в процессе тренировки сети, на нем проверяют точность *mAP* обучения | 20% от тренировочного набора |

Процесс обучения нейросети представлен на рис. 2.

Ось *X* показывает количество итераций (шагов обучения нейросети, на каждый шаг в процессе обучения используется по 24 картинки из тренировочного сета). Ось *Y* определяет степень ошибки (*Lossaverage*) распознавания объекта в процессе обучения (на тренировочном сете), *mAP%* - показывает точность детектирования объектов относительно тестового сета, который не участвует в процессе обучения. Первоначально можно заметить, что средняя ошибка (*Loss*) остается константой до 501 итерации, к этому времени сеть уже должна пройти все картинки хотя бы один раз и обучиться методом обратного распространения ошибки. Как видно в процессе обучения средняя ошибка снижается, в то время как точность распознавания доходит до 97%.

Вычисленный ROI для глаза представлен на рис. 3.

**Этап №3**

Далее нам нужно выявить наиболее контрастную область, между зрачком и радужной оболочкой, возникающую при ИК-подсветке и увеличить резкость. Используем адаптивную эквализацию гистограмм с ограничением контраста CLAHE (Contrast Limited Adaptive Histogram Equaliza-tion) с сеткой 7 x 7 и порогом 2.0. Он отличается от обычного выравнивания гистограммы тем, что адаптивный метод вычисляет несколько гистограмм, каждая из которых соответствует отдельному участку изображения (мы разделили изображение на 49 тайлов), и использует их для перераспределения яркости изображения. Использование сетки 7 x 7 обусловлено положением глаза в кадре, границы сетки должны наложится на зрачок, для вычисления контраста между белой склерой и зрачком. Значение порога 2.0 подобрано экспериментальным путем, при больших значениях увеличивается время обработки одного кадра, а также появляется шум в виде пигментного эпителия радужки, стромы, ресниц. Полученный кадр представлен на рис. 4.

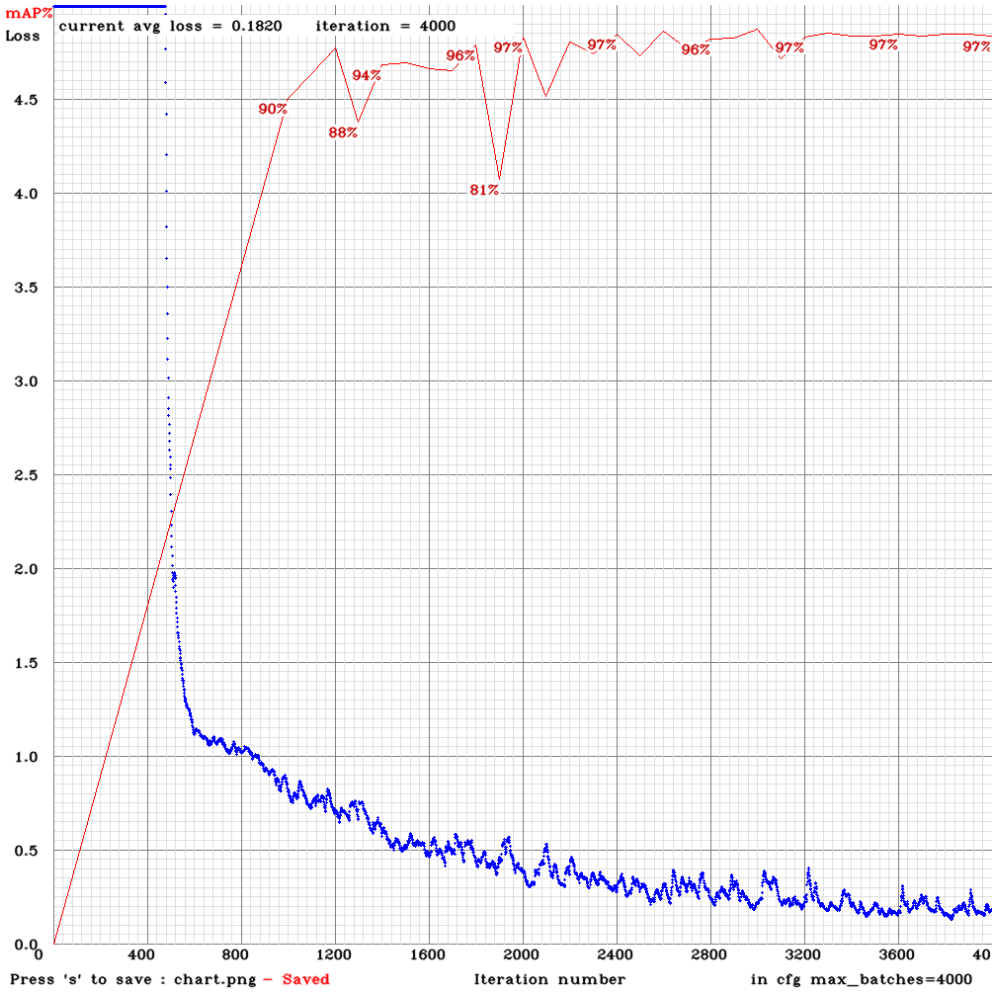


Рис. 2. График обучения нейронной сети

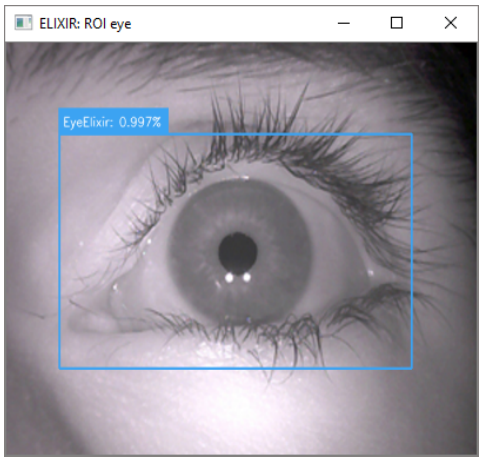


Рис. 3. ROI (Regions of interests) обнаруженного глаза (mAP = 97%)

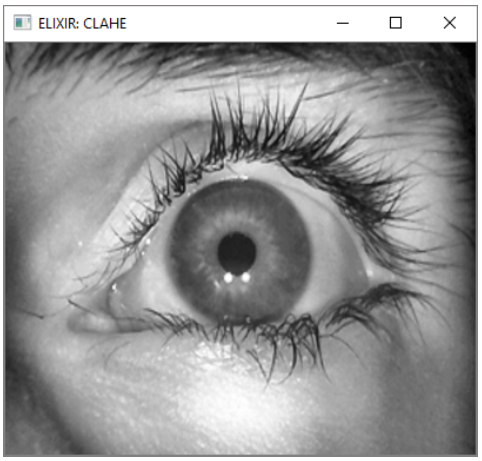


Рис. 4. Применение адаптивной эквализации CLAHE

**Этап №4**

Проведем бинаризацию изображения, для того чтобы радикально уменьшить количество информации, с которой придется работать, методом $cv2.threshold(CLAHEframe, max(0,(1.0-\sigma)*m),255,cv2.THRESH\_BINARY\_INV)$.

**Бинаризация изображений** - перевод полноцветного или в градациях серого изображения в монохромное, где присутствует два типа пикселей (темные и светлые). Главным параметром такого преобразования является порог $t$ – значение, с которым сравнивается яркость каждого пикселя. По результатам сравнения, пикселю присваивается значение 0 или 1 (если значение яркости пикселя не входит в интервал $[LOW\_THRESHOLD; HIGH\_THRESHOLD = 255]$ то 1, иначе 0 т.к. используем инвертор $cv2.THRESH\_BINARY\_INV$). Мы использовали глобальную бинаризацию (т.е. бинаризация осуществляется по всему изображению сразу) с адаптивным нижним порогом (для каждого кадра мы рассчитываем порог по формуле):

$$max(0, (1.0 - \sigma) * m)) \tag{1}$$

где, $\sigma$ – процент бинаризации изображения, для нашей задачи подходит значение 0.45, $m$ – медиана двумерной матрицы (для двумерной матрицы – это вектор-строка $M$ длиной $N$, содержащая значение срединных элементов каждого столбца, каждый из которых равен:

$$M(i) = \begin{cases} \frac{M[(\frac{N}{2})-1]+M[\frac{N}{2}]}{2} & \text{, для N mod } 2 = 0, \\ M[\text{N div } 2] & \text{, для N mod } 2 \neq 0 \end{cases}$$

и медиана от этого вектора). Бинаризованое изображение представлено на рис. 5.


Рис. 5. Бинаризация изображения с адаптивным порогом (по медиане)


Рис. 6. Методы мат. морфологии и медианный фильтр)

**Этап №5**

После бинаризации необходимо убрать шумы, нарастить нужные области (blob'ы), убрать зернистость по краям областей. Делается это с помощью методов математической морфологии.

Две базовые операции математической морфологии, на которых строится абсолютно все - это так называемые дилатация (delation) и эрозия (erosion), и две составные операции: размыкание (opening) и замыкание (closing), которые строятся как суперпозиция из первых двух базовых.

Эрозия – морфологическое сужение, истончает изображения используя матрицу свертки (ядро) определенной формы и размера. Результирующее изображение формируется из локальных минимумов – т.е. будут увеличиваться темные области. Математически эрозию можно описать как:

$$A \ominus B = \{z | B_z \subseteq A\}, \tag{2}$$

где $A$ – представление исходного изображения, $B$ – матрица свертки (ядро).

В алгоритме мы использовали функцию $morphology = cv2.morphologyEx$ $(threshold\_frame, cv2.MORPH\_ERODE, np.ones((3,3)), iterations = 1)$ с квадратной матрицей свертки 3x3 в один проход. Изображение, обработанное эрозией представлено на рис. 1.

Оставшиеся объекты необходимо нарастить. Будем использовать дилатацию. Дилатация – морфологическое расширение, наращивание, изображение формируется из локальных максимумов (наращиваем светлые области). Математически дилатацию можно описать как:

$$A \oplus B = \{z | B_z \cap A \neq \oslash\}, \tag{3}$$

В алгоритме мы использовали функцию $morphology = cv2.morphologyEx(morphology,$ $cv2.MORPH\_DILATE, kernel, iterations = 3)$ с круглой матрицей свертки 5 x 5 в количестве 5 итераций.

$$kernel = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Замыкание – позволяет удалить небольшие внутренние «дырки» и убрать зернистость по краям области. Дырки удаляются за счет начального применения дилатации, последующее применение эрозии обеспечивает обратное уменьшение границы [5].

В алгоритме мы использовали функцию $morphology = cv2.morphologyEx(morphology,$ $cv2.MORPH\_CLOSE, np.ones((3,3)), iterations = 1)$ с квадратной матрицей в один проход.

**Этап №6**

Следующий этап состоит в применении медианного фильтра для сглаживания изображения и краев (рис. 6), обнаруженных blob'ов. В OpenCV данный фильтр можно применить, используя функцию $cv2.medianBlur(morphology, 3)$.

На этом этапе мы закончили препроцессинг изображения.

**Этап №7**

Далее необходимо обнаружить контуры. Есть два варианта: использование детектора границ Канни, или использование функции $\_, contours, hierarchy =$ $cv2.findContours(blur\_frame, cv2.RETR_TREE, cv2.CHAIN\_APPROX\_SIMPLE)$.

$cv2.findContours$ - функция, которая находит контуры в бинарном изображении, при этом возвращает массив контуров, каждый из которых является вектором точек, составляющих этот контур. Первый параметр – кадр, в котором ведется поиск контуров, второй параметр отвечает за режим группировки, мы использовали

*CV_RETR_TREE* – группирует контуры в многоуровневую иерархию, третий параметр отвечает за метод упаковки *CV_CHAIN_APPROX_SIMPLE* – склеивает все горизонтальные, вертикальные и диагональные контуры. Обнаруженные контуры представлены на рис. 7.


Рис. 7. Поиск всех контуров в бинарном изображении и их отображение во фрейме

**Этап №8**

На этом шаге нам необходимо отсеить ненужные контуры, провести их замыкание и найти центр blob'а – отображения зрачка (будем считать, что blob, который находится в ROI-найденного глаза, имеющий наиболее подходящий размер и эллипсоидную форму является зрачком глаза).

Далее переходим к основной части алгоритма - отсеиванию ненужных контуров. Алгоритм отсеивания контуров в кадре выглядит следующим образом:

1) Задаем с помощью генератора списков или функции $map()$, список контуров, площадь которых находится в диапазоне от 300 до 3000 $forcntin[iforiincontoursif300 < cv2.contourArea(i) < 3000]$. Площадь, ограниченную контуром, находим функцией $cv2.contourArea()$;

2) Для отобранных контуров ищем экстремумы (диалект python3):

   ) $left\_extr = tuple(cnt[cnt[:,:,0].argmin()][0])$,

   ) $right\_extr = tuple(cnt[cnt[:,:,0].argmax()][0])$,

   ) $top\_extr = tuple(cnt[cnt[:,:,1].argmin()][0])$,

   ) $bottom\_extr = tuple(cnt[cnt[:,:,1].argmax()][0])$.

   После чего определяем, входят ли экстремумы контура в ROI найденного глаза $(x, y, w, h)$;

3) Если контур входит в ROI, используем алгоритм построения выпуклых оболочек Джарвиса (рис. 8), он определяет последовательность элементов множества, образующих выпуклую оболочку для этого множества. Для контура $P = \{p_1, p_2, \ldots, p_n\}$ в качестве начальной берется самая левая нижняя точка $p_1$ (ее можно найти за $O(n)$ обычным проходом по всем точкам или найти самую дальнюю точку от центроида множества $P$), она точно является вершиной выпуклой оболочки. Следующей точкой ($p_2$), берем такую точку, которая имеет наименьший положительный полярный угол относительно точки $p_1$ как начала координат. После этого для каждой точки $p_i (2 < i < |P|)$ против часовой стрелки ищется такая точка $p_{i+1}$ путём нахождения среди оставшихся точек (+ самая левая нижняя), в которой будет образовываться наибольший угол между прямыми $p_{i-1}p_i$ и $p_ip_{i+1}$. Она и будет следующей вершиной выпуклой оболочки. Сам угол при этом не ищется, а ищется только его косинус через скалярное произведение между лучами $p_{i-1}p_i$ и $p_ip'_{i+1}$, где $p_i$ — последний найденный минимум, $p_{i-1}$ — предыдущий минимум, а $p'_{i+1}$ — претендент на следующий минимум. Новым минимумом будет та точка, в которой косинус будет принимать наименьшее значение (чем меньше косинус, тем больше его угол). Нахождение вершин выпуклой оболочки продолжается до тех пор, пока $p_{i+1} \neq p_i$. В тот момент, когда следующая точка в выпуклой оболочке совпала с первой, алгоритм останавливается — выпуклая оболочка построена. Данный алгоритм реализуется функцией *cv2.convexHull*();

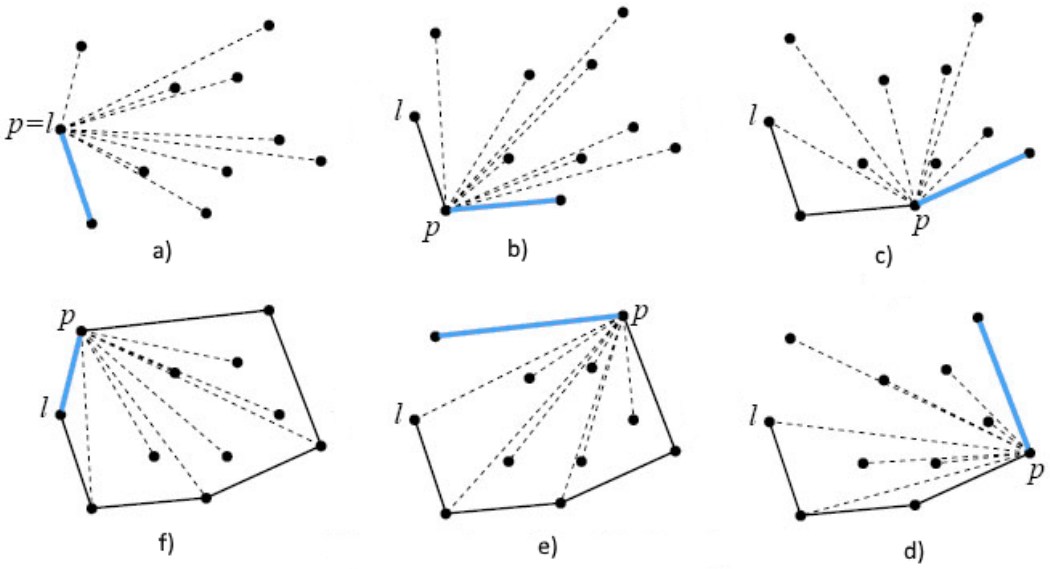


Рис. 8. Алгоритм Джарвиса построения выпуклой оболочки

4) Находим периметр контура функцией *cv2.arcLength*() и площадь поверхности, ограниченную данным контуром, после чего проверяем округлость контура формулой:

$$0.81 < \frac{(4 * \pi * area)}{perimeter^2} < 1.0. \quad (4)$$

Если контур не отсеивается найдем моменты контура для вычисления центра масс контура. Момент — это суммарная характеристика контура, рассчитанная суммиро-

ванием всех пикселей контура:

$$M(p,q) = \sum_{i=1}^{n} I(x,y)x^p y^q, \tag{5}$$

здесь $p$ и $q$ — порядок возведения в степень соответствующего параметра при суммировании, $n$ — число пикселей контура.

Для подсчета центра масс контура (центроид) воспользуемся формулой:

$$Centroid = [\frac{\sum_{i=1}^{n} x_i}{n}; \frac{\sum_{i=1}^{n} y_i}{n}], \tag{6}$$

либо прибегнем к помощи OpenCV (диалект python3):

$$cnt_x = int(\frac{M["m10"]}{M["m00"]}), cnt_y = int(\frac{M["m01"]}{M["m00"]}). \tag{7}$$

Отфильтрованное изображение представлено на рис. 9.


**Рис. 9.** Поиск всех контуров в бинарном изображении и их отображение во фрейме

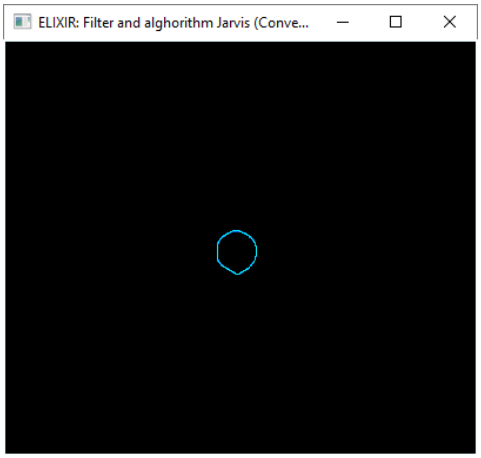


**Рис. 10.** Фильтрация контуров по морфометрическим параметрам и замыкание алгоритмом Джарвиса

В данном кадре не имеется бликов, которые попадают на зрачок, разрывая найденный контур, поэтому приведем еще один пример, в котром будет представлено построение выпуклой оболочки методом Джарвиса (рис. 11).


**Рис. 11.** Построение выпуклой оболочки контура, в случае попадания блика IR-подствеки в область зрачка

После нахождения координат следующим этапом будет вывод необходимой информации во фрейм.

**Этап №9**

Вывод фреймов: оригинальный, градации серого, CLAHE, бинаризованный, морфология, медианный фильтр, найденные контуры, фильтрконтуры, финальный фрейм. Финальный фрейм с обнаруженным ROI глаза, кон-турами и найденным центром зрачка представлен на рис. 12. Точность детектирования достигает 99% при скорости обработки кадров в 14fps.

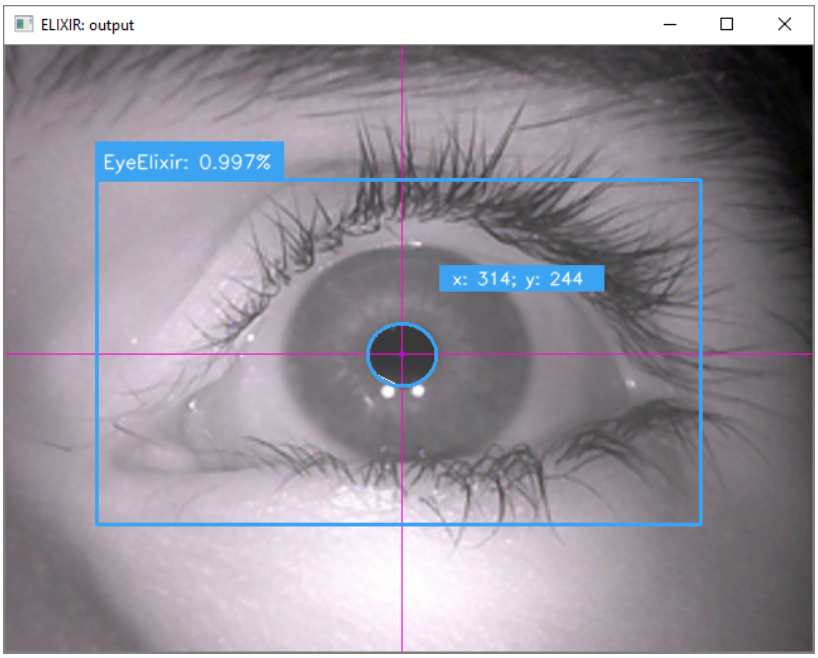


**Рис. 12.** Обнаруженный ROI глаза (YOLO v3) и зрачок (методы мат. морфологии)

## Разработка клиентского программного обеспечения для АПК

Аппаратно-программный комплекс предназначен для установки и использования в медицинских учреждениях и реабилитационных центрах для больных инсультом и лежачих больных.

Для взаимодействия аппаратного обеспечение айтрекера с ПК пациента (или его ассистента) было разработано десктопное клиентское программное обеспечение.

Для разработки дизайна клиентского приложения был использован пакет разработки интерфейсов от Adobe Systems – Adobe XD.

После проектирования UI/UX дизайна приложения, мы приступили к программной разработке, используя Qt - кроссплатформенный фреймворк для разработки программного обеспечения на языке программирования C++. Есть также «привязки» ко многим другим языкам программирова-ния: Python, Ruby, Java.

В качестве рабочего языка для backend-слоя выступил Python 3. Для его использования необходимо использовать официальную привязку к фреймворку Qt – PySide2. В качестве IDE для написания кода выступал Visual Studio Code 2019-19 версии (с привязанным интерпретатором python).

Для создания графической оболочки, frontend-слоя, использовалась технология QtQuick, входящая в состав IDE QtCreator, с использованием декларативного языка QML.

Для создания БД к сервису АПК была использована бесплатная, многофункциональная и надежная система управления реляционными базами данных MySQL. Используемый язык запросов – mysql. Используется для работы с БД размером от персональных до крупных БД масштаба организаций и предприятий.

Требования к техническому обеспечению: IBM PC-совмест. ПК, Intel Core i5 или выше, 2Гб RAM, тактовая частота процессора не ниже 1.9 ГГц, адаптер Fast Ethernet 100 или Wi-Fi, графическая карта NVIDIA GeForce GTX 750 или выше, с поддержкой технологии CUDA (не менее 128 пото-ковых ядер).

После запуска программы откроется главное окно приложения (рис. 13).

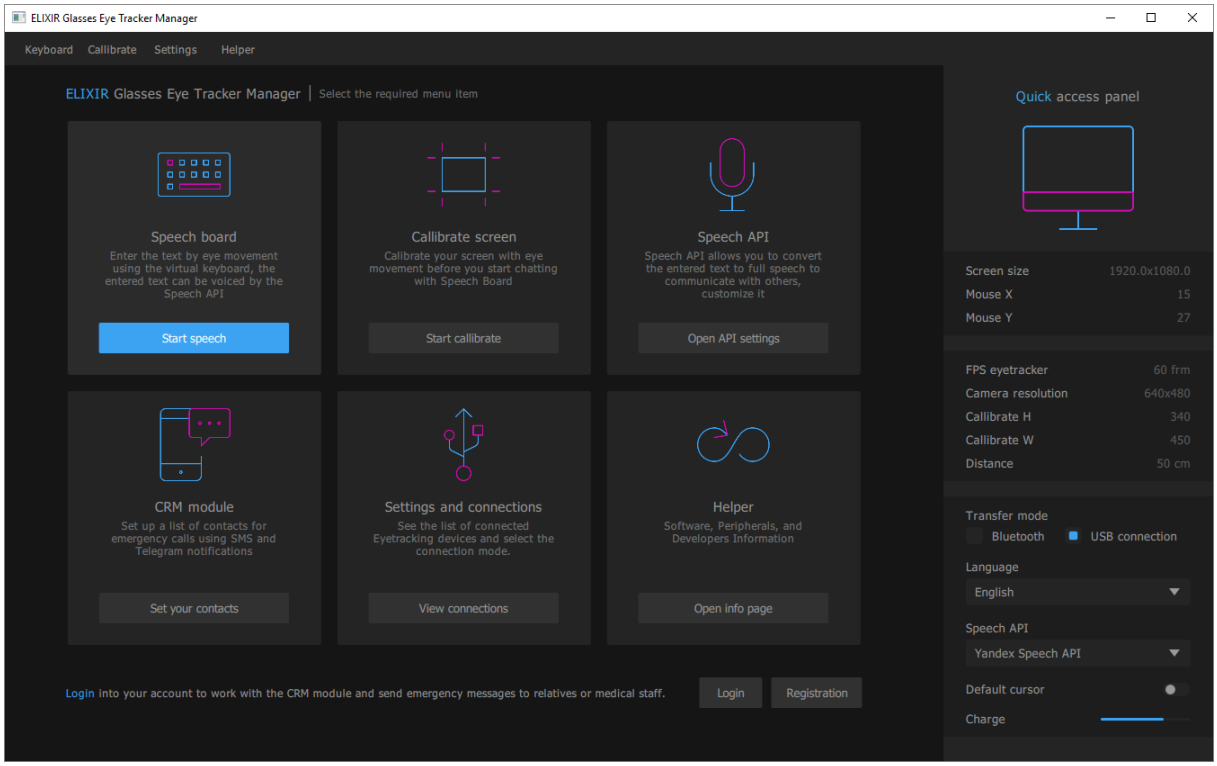


**Рис. 13.** Главное меню клиентского приложения АПК

Основное меню содержит следующие вкладки:

1) Виртуальную клавиатуру «Speech board» – позволяет пользователю вводить текст движением глаз, использовать тексты-пресеты для быстрого набора, совершать озвучку напечатанного текста, используя речевые модули, отправлять экстренные сообщения;

2) Экран калибровки «Calibrate screen» – позволяет пользователю настроить область ввода и трекинга, увеличить точность;

3) Настройки TTS-модуля «Speech API» – настройка речевых модулей (гендер, эмоциональный окрас, язык и др.), настройка текстов-пресетов;

4) Настройки CRM-модуля «CRM module» – настройка модуля отправки экстренных сообщений, редактирование списка контактов-получателей, возможность узнать текущий баланс и просмотр истории отправки сообщений;

5) Подключенные устройства «Settings and connections» – просмотр списка подключенных устройств, выбор и настройка устройства-трекера;

6) Помощник по ПО «Helper» – просмотр документации к АПК.

Для набора текста на виртуальной клавиатуре пользователю (пациенту) необходимо нажать на кнопку «Start speech» на соответствующей вкладке. В результате чего откроется окно с виртуальной клавиатурой (рис. 14).

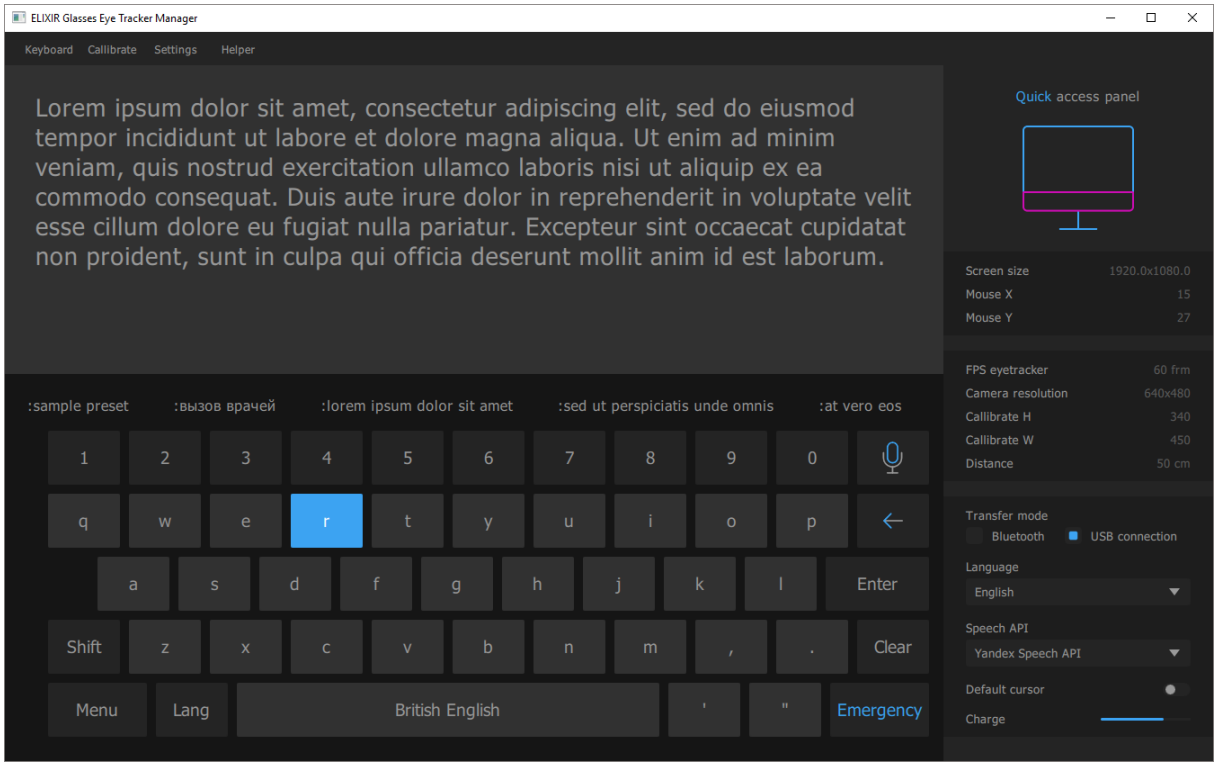


**Рис. 14.** Виртуальная клавиатура АПК

Для озвучки напечатанного текста необходимо нажать на кнопку «Speech», после чего программа ненадолго перейдет в режим ожидания (1 – 2 секунды в зависимости от объема текста). В это время ПО осуществляет аутентификацию в сервисах Google или Яндекс, отправляет набранный текст на распознавание и загружает распознанную звуковую дорожку на компьютер в буферную папку «buffer_voice_message», находящуюся в той же директории что и исполняемый файл программы, после чего программа автоматически запускает звуковую дорожку в аудиоплеере (установленным по умолчанию в ОС). Буфер программы составляет 25 звуковых сообщений – дорожек (далее идет перезапись). Если текст не изменился, и пользователь нажал на кнопку «Speech», программа не будет повторно синтезировать дорожку, а проиграет крайнюю из буферной папки (тем самым экономя средства и время пользователя).

Над клавиатурой располагается список текстов-пресетов (максимальное количество 5шт.). Пресет содержит заранее набранный текст (количество символов не ограничено) в настройках «Speech API» под определенным именем. Для использования пресета, необходимо нажать на него, текст пресета отобразится в области набора текста.

Для отправки экстренного сообщения, необходимо ввести текст используя виртуальную клавиатуру или пресеты и нажать на кнопку «Emergency», после чего

программа перейдет в режим ожидания 1 с. В это время ПО осуществляет выборку контактов-получателей из БД по идентификатору зарегистрированного пользователя, рассылает сообщение и создает записи в БД (история сообщений: текст, стоимость, контакты-получатели и др.).

Для настройки речевых модулей необходимо перейти в главное меню и нажать на кнопку «Open API settings» на соответствующей вкладке. В результате чего откроется окно с настройками речевых модулей и виртуальной клавиатуры (рис. 15).

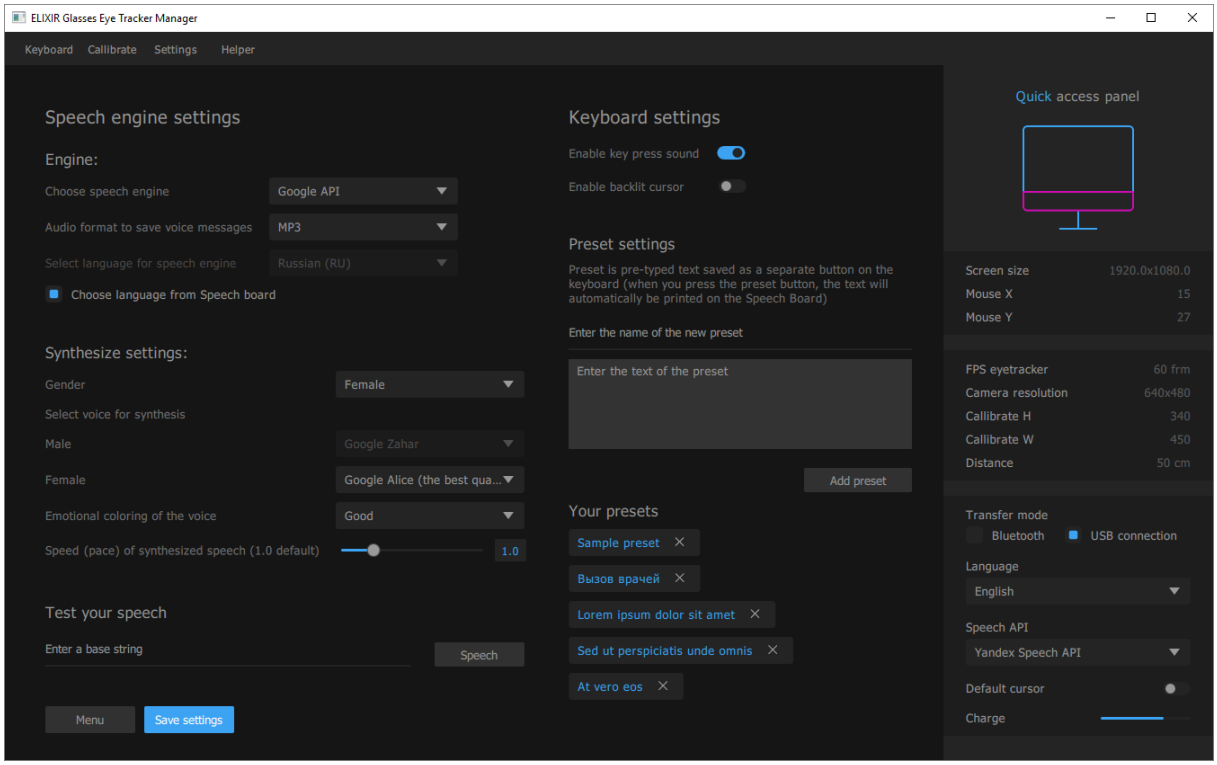


**Рис. 15.** Окно настроек TTS-модуля АПК

Окно визуально разделено на две области: «Speech engine settings», «Keyboard settings».

В области «Speech engine settings» осуществляется настройка TTS-модуля. Данный модуль обладает гибкими настройками:

1) Выбор TTS – движка. Доступны следующие варианты из выпадающего списка: Google TTS, Yandex TTS, Elixir TTS (на стадии разработки);

2) Аудиоформат сохранения в буферную папку. Доступны следующие варианты из выпадающего списка: MP3, OGG OPUS, WAVE;

3) Язык озвучки. Если выбрана опция «Choose language from Speech board» (установка языка в виртуальной клавиатуре) – включится возможность выбора языка синтеза во время печати с виртуальной клавиатуры. В противном случае пользователю необходимо принудительно установить язык синтеза из выпадающего списка: Russian (ru-RU), English (en-US), Turkish (tr-TR);

4) Гендер. Доступно три варианта из выпадающего списка: Male, Female, Neutral;

5) Голос – возможность выбора нужного тембра и качества озвучки. В зависимости от выбора значения предыдущего параметра, пользователю буду предложены мужские (3шт) или женские (3шт) голоса. Голоса с пометкой «the best quality» при синтезе создают аудиодорожки с высоким качеством (но при этом увеличивается время и стоимость синтеза);

6) Эмоциональный окрас голоса: Good, Evil, Neutral;

7) Скорость (темп) синтезированной речи. Скорость речи задается дробным числом в диапазоне от 0.1 до 3.0 перемещением ползунка. Где:

   1) 3.0 – самый быстрый темп;

   2) 1.0 (по умолчанию) – средняя скорость человеческой речи;

   3) 0.1 – самый медленный темп.

Также имеется возможность прослушать свои настройки в разделе «Test your speech».

В области «Keyboard settings» осуществляется управление основными настройками виртуальной клавиатуры. Для включения/отключения звукового сопровождения нажатия кнопки необходимо вклю-чить/отключить свитчер в пункте «Enable key press sound». Для включения/отключения эффекта курсора необходимо включить/отключить свитчер в пункте «Enable backlit cursor».

В разделе «Preset settings» пользователь может установить свои пресеты. В верхнее текстовое поле необходимо ввести наименование пресета (1 – 2 слова). В текстовую область следует ввести текст пресета. Для добавления пресета в БД нажмите на кнопку «Add preset».

## Заключение

В результате был разработан аппаратно-программный комплекс аугментативной системы коммуникации с использованием технологии айтрекинга, который позволяет людям, страдающим нарушениями двигательных функций, нервно-паралитическими синдромами и другими болезнями ЦНС полноценно осуществлять процесс коммуникации с внешним миром (медицинский персонал, родственники).

Данный АПК может применяться в медицинских учреждениях, реабилитационных центрах для людей после инсульта и лежачих больных.

Основной задачей являлось создание алгоритма трекинга движения глаз с использованием сверточных нейронных сетей, контурного анализа и методов математической морфологии, на базе библиотеки машинного зрения OpenCV а также клиентского приложения, с помощью которого можно набирать текст движением глаз и осуществлять озвучку. Разработанный алгоритм "OODI" рассчитан на применении в медицинской окулографии, например, при регистрации нистагма и тремора глаз, офтальмоплегии, анализа и детектирования саккад и фиксаций при рассмотре некоторого визуального стимула и др. Вкупе с аппаратной и программной частью, алгоритм показал точные результаты детектирования глаза,

Была доказана экономическая и практическая значимость разработанного АПК (не рассматривается в данной работе), которая определяется увеличением производительности труда врачей реабилитологов, за счет высвобожденного времени, сокращением трудовых, временных и материальных ресурсов. Практическая значимость определена возможностью качественной коммуникации пациентов с затратой минимальных физических усилий, увеличением скорости и полноценным (вербальным) общением с использованием синтезированной речи.

## Список литературы/References


[1] *Национальный регистр инсульта http://www.nabi.ru/spetsialistam/klinicheskie-rekomendatsii/registr-insulta.html.* [ *Natsional'nyy registr insul'ta http://www.nabi.ru/spetsialistam/klinicheskie-rekomendatsii/registr-insulta.html*].

[2] Зайцева Г. Л., *Жестовая речь. Дактилология.*, изд. центр ВЛАДОС, г. Москва, 2000 г. http://spedkoll.ru/obereg/2014/october_obereg/augumentation.pdf. [Zaytseva G.L., *Zhestovaya rech'. Daktilologiya.*, izd. tsentr VLADOS, g. Moskva, 2000 g. http://spedkoll.ru/obereg/2014/october_obereg/augumentation.pdf].

[3] *Окулография*, Википедия https://ru.wikipedia.org/wiki/Окулография. [ *Okulografiya*, Vikipediya https://ru.wikipedia.org/wiki/Окулография].

[4] *Как работает сверточная нейронная сеть: архитектура, примеры, особенности*, neurohive.com, 17 июля 2018 г. https://neurohive.io/ru/osnovy-data-science/glubokaya-svertochnaja-nejronnaja-set/. [ *Kak rabotaet svertochnaya neyronnaya set': arkhitektura, primery, osobennosti*, neurohive.com, 17 iyulya 2018 g. https://neurohive.io/ru/osnovy-data-science/glubokaya-svertochnaja-nejronnaja-set/].

[5] Маковейчук А. Н. и др., *Методы математическской морфологии*, Бюро информционных технологий, г. Львов, 2008 g. `http://www.hups.mil.gov.ua/periodic-app/article/6336/rus`. [Makoveychuk A. N. i dr., *Metody matematichesksoy morfologii*, Byuro informtsionnykh tekhnologiy, g. L'vov, 2008 g. http://www.hups.mil.gov.ua/periodic-app/article/6336/rus].


## Список литературы (ГОСТ)


[1] Национальный регистр инсульта. http://www.nabi.ru/spetsialistam/klinicheskie-rekomendatsii/registr-insulta.html

[2] Зайцева Г. Л. Жестовая речь. Дактилология. М.: ВЛАДОС, 2000. http://spedkoll.ru/obereg/2014/october_obereg/augumentation.pdf

[3] Окулография. Википедия. https://ru.wikipedia.org/wiki/Окулография

[4] Как работает сверточная нейронная сеть: архитектура, примеры, особенности. https://neurohive.io/ru/osnovy-data-science/glubokaya-svertochnaja-nejronnaja-set/(Дата обращения: 17 июля 2018 г.).

[5] Маковейчук А. Н. и др. Методы математическской морфологии. Бюро информционных технологий. Львов: 2008. http://www.hups.mil.gov.ua/periodic-app/article/6336/rus


**Для цитирования:** Грушко Ю. В. Аппаратно-программный комплекс для коммуникации людей, страдающих нарушениями двигательных функций, нервнопаралитическими синдромами и другими болезнями ЦНС, посредством технологии EyeTracking (окулографии) // *Вестник КРАУНЦ. Физ.-мат. науки*. 2019. Т. 27. № 2. С. 55-73. DOI: 10.26117/2079-6641-2019-27-2-55-73

**For citation:** Grushko Yu. V. Hardware and software complex of augmentation communication system on the basis of Eyetracking Technologys, *Vestnik KRAUNC. Fiz.-mat. nauki*. 2019, **27**: 2, 55-73. DOI: 10.26117/2079-6641-2019-27-2-55-73

Vestnik KRAUNC. Fiz.-Mat. Nauki. 2019. vol. 27. no.2. pp. 55–73.

DOI: 10.26117/2079-6641-2019-27-2-55-73

INFORMATION AND COMPUTING TECHNOLOGIES

MSC 91B44

# HARDWARE AND SOFTWARE COMPLEX OF AUGMENTATION COMMUNICATION SYSTEM ON THE BASIS OF EYETRACKING TECHNOLOGYS

**Yu. V. Grushko**

Vitus Bering Kamchatka State University, 683031, Petropavlovsk-Kamchatsky, Pogranichnaya st., 4, Russia

E-mail: thekidsshow96@gmail.com

The paper presents a description of the developed APC and implemented the algorithm of EyeTracking (based on convolutional neural networks and morphological analysis), which allows people suffering from motor dysfunction and neuroparalytic syndromes to carry out the process of communication with the outside world through computer control and text printing by eye movement

*Key words: Eyetracking, oculography, algorithm, HSC, convolutional neural networks, morphological analysis, contour analysis, cascade classifiers, Haar cascade, Viola-Jones method, machine vision, OpenCV, dnn, deep neural network, microcontrollers, erosion, dilation, convolutional core, binarization, algorithm for constructing convex hulls, Jarvis algorithm, YOLO.*



Поступила в редакцию / Original article submitted: 15.05.2019