

Содержание

Введение	3
1 Методы детектирование кисти человека в системах человеко-машинного взаимодействия	4
1.1 Непосредственное отделение фона изображения	4
1.2 Метод распознавания кожи в HSV и YCbCr цветовых моделях	6
1.2.1 Цветовая модель HSV	6
1.2.2 Цветовая модель YCbCr	8
1.2.3 Использование цветовых моделей RGB и YCrCb для распознавания кожи.	9
1.3 Метод Оцу	13
1.4 Рекурсивные методы построения модели фона	17
1.4.1 Visual Background Extractor.	17
1.4.2 Mixture of Gaussians.	19
Заключение	22
Список литературы	23

Введение

Развитие технологий, продолжающееся уже более полувека, приводит к тому, что нижняя граница размеров процессоров уменьшается, в то время как их производительность продолжает увеличиваться. Для рядового пользователя результатом этого является разнообразие созданных интеллектуальных систем, используемых в повседневной жизни: от смартфонов до планшетов, от бытовой техники до домашних роботов. Камнем преткновения становится обмен информацией между компьютером и пользователем, растёт потребность в исследовании новых способов, более естественных, чем ввод данных с клавиатуры, эмулирующих бытовое общение между людьми. Активно развивающимся направлением решения проблемы является управление компьютером с помощью голосовых команд. Тем не менее, несмотря на значительные успехи в области, этот способ применим далеко не во всех ситуациях. Другим исследуемым способом ввода данных является использование визуальных систем: передача информации посредством мимики и жестов. Для реализации последнего метода можно воспользоваться многими способами.

В данной работе идёт речь о поиске так называемых особых точек на кисти руки для дальнейшей их обработки. Поиск таких точек позволяет извлекать конфигурацию кисти и давать её описание, что позволяет не только задавать исполнение определённых команд по заранее определённым жестам, но и выполнять совершенно не характерные для обычной реализации данного метода действия.

В ходе исследования разрабатывается алгоритм детектирования кисти руки человека и её описания с помощью "особых" точек. Важным требованием для реализации является минимизация времени задержки на обработку каждого кадра для комфорtnого использования в прикладных задачах.

Поставленной **целью** является создание программы для детектирования и извлечения конфигурации кисти руки человека с помощью языка высокого уровня Python.

Для достижения цели необходимо решить ряд **задач**:

1. Сделать обзор теоретического материала по детектированию кисти руки человека.
2. Найти и описать наиболее популярные методы обнаружения кисти человека и выделить из них самые эффективные.
3. Провести сравнение нескольких алгоритмов решения данной задачи.

1 Методы детектирование кисти человека в системах человеко-машинного взаимодействия

Человеко-машинное взаимодействие (Human-computer interaction - HCI) - это междисциплинарное научное направление, изучающее взаимодействие между людьми и машинами. Предметом HCI является изучения, планирование и разработка методов взаимодействия человека с машиной, где в роли машины может выступать персональный компьютер, компьютерная система больших масштабов, система управления процессами и т.д. [1]. Под взаимодействием понимается любая коммуникация между человеком и машиной. Одним из методов HCI, получившим широкое распространение в последние годы, является взаимодействие, основанное на жестах человека [2, 3].

Задачу распознавания жестов руки можно разделить на подзадачи:

1. Отделение кисти руки от остальной части изображения.
2. Построение контура кисти.
3. Нахождение ключевых точек на кисти.
4. Классификация жеста исходя из статического или динамического расположения точек.

Первая подзадача, а именно детектирование кисти человека в кадре является ключевой, поскольку от качества её решения зависит качество выполнения остальных подзадач. Рассмотрим первую подзадачу, а именно детектирование кисти человека в кадре.

Существует множество решений этой подзадачи. Наиболее популярными из них являются непосредственное отделение фона изображения, распознавание цвета кожи в кадре и метод Оцу. Подробно рассмотрим каждое из них.

1.1 Непосредственное отделение фона изображения

В данном решении принимается, что в кадре движется только рука, а остальные части тела, включая фон, остаются неподвижными. Таким образом, если вначале инициализировать фон как $bgr(x, y)$, а новое изображение с жестом рассматривать как $fgr(x, y)$, то изолированный жест можно принять как разность между этими изображениями:

$$gst_i(x, y) = fgr_i(x, y) - bgr(x, y). \quad (1)$$

Полученный разностный жест переднего плана преобразуется в бинарное изображение, устанавливая соответствующий порог. Поскольку фон не является полностью статичным, например, если камера удерживается в руках оператора, то добавляется шумовая часть. Чтобы получить изображение руки без шумов, этот метод сочетается с распознаванием кожи человека. Чтобы удалить этот шум, применяется анализ связанных компонентов, чтобы заполнить пустоты, если применяется заливка какой-либо

области, а также для получения чётких краёв применяется морфологическая обработка.

Недостаток данного решения состоит в том, что довольно сложно отделить фон, даже если он задан, поскольку не ясно какие из пикселей изменились, а какие остались прежними из-за тени, смещения фокуса, изменения экспозиции и т.д. Даже если зафиксировать все параметры камеры, то тень так или иначе испортит качество решения.

Существует другой способ, он заключается в следующем. Пусть даны два пикселя

$$p_1 = (r_1, g_1, b_1) \quad p_2 = (r_2, g_2, b_2), \quad (2)$$

тогда различие между ними будем определять как

$$D = \sqrt{(r_2 - r_1)^2 + (g_2 - g_1)^2 + (b_2 - b_1)^2}. \quad (3)$$

Затем создаём битовую маску M , в которой значения определяются как

$$M_{(i,j)} = \begin{cases} 1, & D > Threshold \\ 0, & D \leq Threshold \end{cases} \quad (4)$$

Оба метода не совсем точно отделяют нужный объект от фона, поэтому рассмотрим метод распознавания кожи в HSV и YCbCr цветовых моделях.

1.2 Метод распознавания кожи в HSV и YCbCr цветовых моделях

Для того, чтобы исследовать соответствующие методы следует познакомиться с цветовыми моделями, которые они используют, поподробнее.

1.2.1 Цветовая модель HSV

HSV (*Hue, Saturation, Value*) или **HSB** (*Hue, Saturation, Brightness*) – цветовая модель, в которой координатами цвета являются:

1. **Hue** – цветовой тон, (например, красный, зелёный или сине-голубой). Варьируется в пределах 0-360°, однако иногда приводится к диапазону 0-100 или 0-1.
2. **Saturation** – насыщенность. Варьируется в пределах 0-100 или 0-1. Чем больше этот параметр, тем "чище" цвет, поэтому иногда называют *чистотой цвета*. А чем ближе этот параметр к нулю, тем ближе цвет к нейтральному серому.
3. **Value** или **Brightness** – яркость. Также задаётся в пределах 0-100 или 0-1.

Простейший способ отобразить HSV в трёхмерное пространство – воспользоваться цилиндрической системой координат (рис. 1). Здесь координата H определяется полярным углом, S – радиус-вектором, а V – Z-координатой. То есть, оттенок изменяется при движении вдоль окружности цилиндра, насыщенность – вдоль радиуса, а яркость – вдоль высоты. Несмотря на "математическую" точность, у такой модели есть существенный недостаток: на практике количество различимых глазом уровней насыщенности и оттенков уменьшается при приближении яркости (V) к нулю. Также на малых S и V появляются существенные ошибки округления при переводе RGB в HSV и наоборот. Поэтому чаще применяется коническая модель (рис. 2).

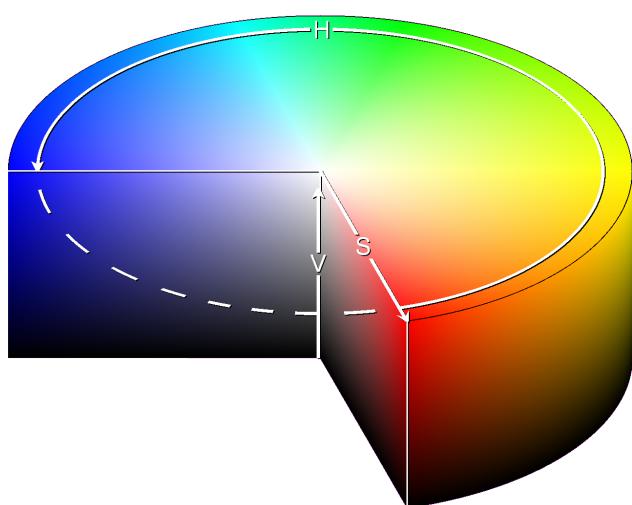


Рис. 1. HSV в цилиндрической системе координат.

Как и в цилиндре, в конической модели оттенок изменяется по окружности конуса. Насыщенность цвета возрастает с удалением от оси конуса, а яркость – с приближе-

нием к его основанию. Иногда вместо конуса используется шестиугольная правильная пирамида.

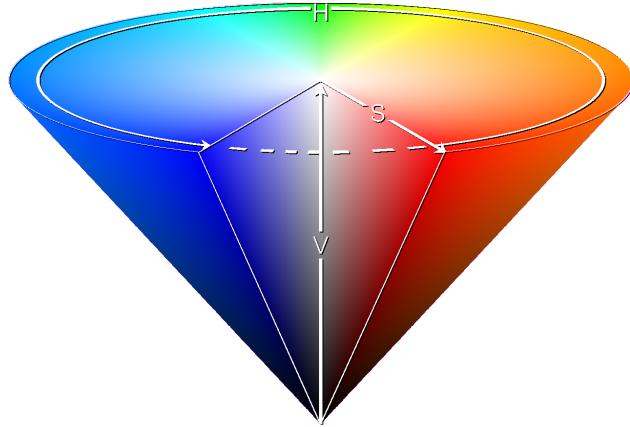


Рис. 2. HSV в конической модели.

Для перевода RGB \Rightarrow HSV будем считать, что

$$H \in [0, 360],$$

$$S, V, R, G, B \in [0, 1].$$

Пусть $MAX = \max(R, G, B)$, а $MIN = \min(R, G, B)$. Тогда

$$H = \begin{cases} 60 \cdot \frac{G - B}{MAX - MIN} + 0, & MAX = R \quad G \geq B \\ 60 \cdot \frac{G - B}{MAX - MIN} + 360, & MAX = R \quad G < B \\ 60 \cdot \frac{B - R}{MAX - MIN} + 120, & MAX = G \\ 60 \cdot \frac{R - G}{MAX - MIN} + 240, & MAX = B \end{cases} \quad (5)$$

$$S = \begin{cases} 0, & MAX = 0, \\ 1 - \frac{MIN}{MAX}, & \end{cases} \quad (6)$$

$$V = MAX \quad (7)$$

Можно заметить, что уравнение (5) не определено, когда $MAX = MIN$, поэтому существуют другие уравнения:

$$\begin{aligned} H &= \arccos \frac{0.5 \cdot ((R - G) + R - B)}{\sqrt{(R - G)^2 + (R - B) + (G - B)}} \\ S &= 1 - 3 \frac{\min(R, G, B)}{R + G + B} \\ V &= \frac{1}{3}(R + G + B) \end{aligned} \quad (8)$$

Для перевода HSV \Rightarrow RGB будем считать, что $H \in [0, 360]$, $S \in [0, 100]$ и $V \in [0, 100]$.

Если

$$\begin{aligned} H_i &= \left\lfloor \frac{H}{60} \right\rfloor \bmod 6, \\ V_{min} &= \frac{(100 - S)V}{100}, \\ a &= (V - V_{min}) \frac{H \bmod 60}{60}, \\ V_{inc} &= V_{min} + a, \\ V_{dec} &= V - a, \end{aligned} \tag{9}$$

тогда по таблице 1 выбираем нужные значения. Полученные значения красного, зелёного и синего каналов RGB исчисляются в процентах. Чтобы привести их в соответствие распространённому представлению COLORREF необходимо умножить каждое из них на $\frac{255}{100}$. При целочисленном кодировании для каждого цвета в HSV есть соответствующий цвет в RGB. Однако обратное утверждение не является верным: некоторые цвета в RGB нельзя выразить в HSV так, чтобы значение каждого компонента было целым. Фактически, при таком кодировании доступна только $\frac{1}{256}$ часть цветового пространства RGB.

H_i	R	G	B
0	V	V_{inc}	V_{min}
1	V_{dec}	V	V_{min}
2	V_{min}	V	V_{inc}
3	V_{min}	V_{dec}	V
4	V_{inc}	V_{min}	V
5	V	V_{min}	V_{dec}

Таблица 1. Таблица значений RGB для перевода из HSV.

1.2.2 Цветовая модель YCbCr

Известно, что органы зрения человека менее чувствительны к цвету предметов, чем к их яркости. В цветовом пространстве RGB все три цвета считаются одинаково важными, и они обычно сохраняются с одинаковым разрешением. Однако можно отобразить цветное изображение более эффективно, отделив светимость от цветовой информации и представив её с большим разрешением, чем цвет. Цветовое пространство $YC_B C_R$ и его вариации является популярным методом эффективного представления цветных изображений.

$YC_B C_R$ – семейство цветовых пространств, которые используются для передачи цветных изображений в компонентном видео и цифровой фотографии. Является ча-

стью рекомендации МСЭ-R BT.601 при разработке стандарта видео Всемирной цифровой организации и фактически является масштабированной и смещённой копией YUV.

Y – компонента яркости, C_B – синий компонент цветности, C_R – красный компонент цветности.

Преобразование RGB $\Rightarrow YC_B C_R$ можно описать следующей формулой:

$$\begin{pmatrix} Y \\ C_B \\ C_R \end{pmatrix} = \begin{pmatrix} 16 \\ 128 \\ 128 \end{pmatrix} + \begin{pmatrix} 65.481 & 128.553 & 24.996 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix} \quad (10)$$

1.2.3 Использование цветовых моделей RGB и YCrCb для распознавания кожи.

Для того чтобы отделить кожу от остальной части изображения используют определённые диапазоны составляющих цветовых моделей, которые находятся эмпирически или итеративно. В первом случае путём проб и ошибок подбираются значения составляющих. Можно заметить, что при данном подходе кожа будет иметь разные цветовые диапазоны при разном освещении. Покажем это. Зададим диапазон для изображения в HSV цветовой модели как

$$\begin{aligned} 0 \leq H \leq 200, \\ 15 \leq S \leq 255, \\ 80 \leq V \leq 255, \end{aligned}$$

и, получаем результат, представленный на рис. 3.

```

1 import cv2
2 import numpy as np
3
4 LOWER, UPPER = np.array([0, 15, 80], dtype='uint8'), \
5                         np.array([200, 255, 255], dtype='uint8')
6
7 INPUT_TO_OUTPUT = {'image3.jpg': 'hsv_del1.png',
8                     'image7.jpg': 'hsv_del2.png'}
9
10 def HSV_skin_detection_mask_first(image):
11     converted = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
12     skinMask = cv2.inRange(converted, LOWER, UPPER)
13     kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (11, 11))
14     skinMask = cv2.erode(skinMask, kernel, iterations=2)

```

```
15     skinMask = cv2.dilate(skinMask, kernel, iterations=2)
16     skinMask = cv2.GaussianBlur(skinMask, (3, 3), 0)
17     return skinMask
18
19
20 for filename_input, filename_output in INPUT_TO_OUTPUT.items():
21     img = cv2.imread(filename_input)
22     hsv_mask = HSV_skin_detection_mask_first(img)
23     cv2.imwrite(filename_output, cv2.bitwise_and(img, img, mask=hsv_mask))
```

Можно видеть, что на первом изображении фон отделился практически идеально, но на втором видны фрагменты фона.

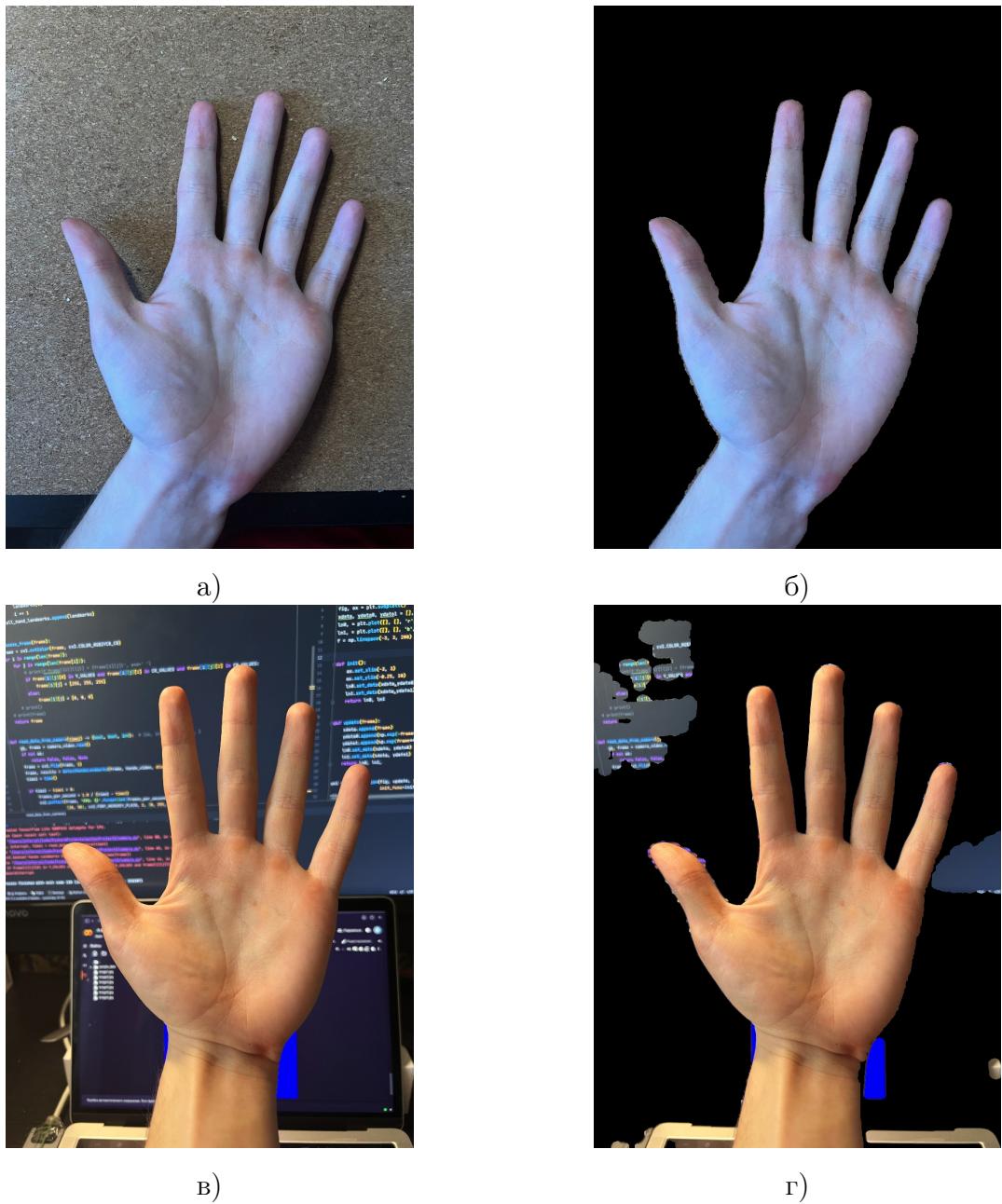


Рис. 3. Изображения с удалённым фоном в HSV.

Для цветовой модели YCbCr в статье [4] предложили два варианта диапазона компонент (11) и (12):

$$\begin{aligned} 80 < Y \leq 255, \\ 85 < C_b < 135, \\ 135 < C_r < 180 \end{aligned} \quad (11)$$

$$\begin{aligned} Y \in \forall, \\ 77 \leq C_b \leq 127, \\ 133 \leq C_r \leq 173 \end{aligned} \quad (12)$$

Выполнив код ниже, получаем сравнение двух методов на рисунке 4.

```
1 import cv2
2
3 LOWER_YCr_Cb1, UPPER_YCr_Cb1 = (79, 89, 134), \
4                                 (255, 134, 179)
5 LOWER_YCr_Cb2, UPPER_YCr_Cb2 = (0, 77, 133), \
6                                 (255, 127, 173)
7 INPUT_TO_OUTPUT = {'image3.jpg': ['ycbcr_del1_1.png', 'ycbcr_del1_2.png'],
8                    'image7.jpg': ['ycbcr_del2_1.png', 'ycbcr_del2_2.png']}
9
10 def YCrCb_skin_detection_mask(image, ex: int):
11     img_YCrCb = cv2.cvtColor(image, cv2.COLOR_BGR2YCR_CB)
12     img_YCrCb = cv2.GaussianBlur(img_YCrCb, (5, 5), 5)
13     if ex == 1:
14         skinMask = cv2.inRange(img_YCrCb, LOWER_YCr_Cb1, UPPER_YCr_Cb1)
15     else:
16         skinMask = cv2.inRange(img_YCrCb, LOWER_YCr_Cb2, UPPER_YCr_Cb2)
17     kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (11, 11))
18     skinMask = cv2.erode(skinMask, kernel, iterations=3)
19     skinMask = cv2.dilate(skinMask, kernel, iterations=3)
20     skinMask = cv2.GaussianBlur(skinMask, (3, 3), 0)
21     return skinMask
22
23 for filename, outputs in INPUT_TO_OUTPUT.items():
24     img = cv2.imread(filename)
25     for i, output in enumerate(outputs):
26         ycrcb_mask = YCrCb_skin_detection_mask(img, i)
27         image_skin = cv2.bitwise_and(img, img, mask=ycrcb_mask)
28         cv2.imwrite(output, image_skin)
```

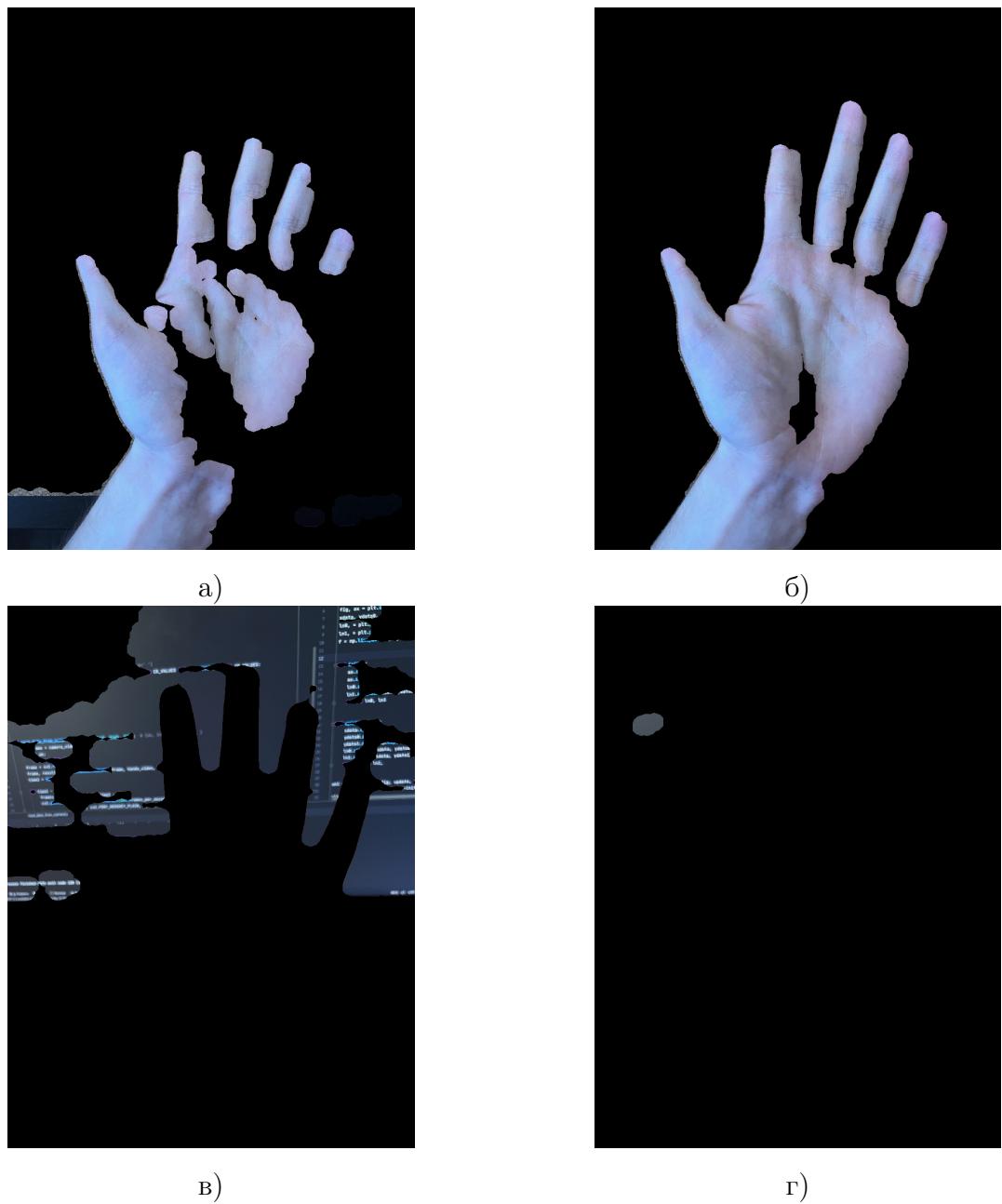


Рис. 4. Изображения с удалённым фоном в YCbCr.

Легко видеть, что оба диапазона работают не идеально, но первый показал себя намного лучше.

Таким образом, можно сделать вывод, что для детектирования цвета кожи лучше брать изображение в цветовой модели HSV.

Рассмотрим следующий метод отделения фона изображения, а именно метод Оцу.

1.3 Метод Оцу

В 1979 году Нобуюки Оцу опубликовал статью [5] метода порогового разделения, основываясь на гистограмме серых цветов изображения.

Метод Оцу – это алгоритм вычисления порога бинаризации для полутонаового изоб-

ражения, используемый в области компьютерного распознавания образов и обработки изображений для получения чёрно-белых изображений. Алгоритм позволяет разделить пиксели двух классов ("полезные" и "фоновые"), рассчитывая такой порог, чтобы внутриклассовая дисперсия была минимальной.

Пусть пиксели изображения представлены в L уровнях серого $[1, 2, \dots, L]$. Количество пикселей уровня i обозначим как n_i , а количество всех пикселей как

$$N = n_1 + n_2 + \dots + n_L.$$

Для того чтобы упростить алгоритм гистограмма монохромного изображения нормализована и рассматривается как распределение вероятностей:

$$p_i = n_i/N, \quad p_i \geq 0, \quad \sum_{i=1}^L p_i = 1. \quad (13)$$

Теперь предположим, что мы разделили пиксели на два класса C_0 и C_1 (фон и объект) по пороговому значению k . C_0 обозначает пиксели с уровнями $[1, \dots, k]$, а C_1 – пиксели с уровнями $[k+1, \dots, L]$. Следовательно, вероятности класса и средний уровень, соответственно, задаются как

$$\omega_0 = \sum_{i=1}^k p_i = \omega(k) \quad (14)$$

$$\omega_1 = \sum_{i=k+1}^L p_i = 1 - \omega(k) \quad (15)$$

и

$$\mu_0 = \sum_{i=1}^k \frac{ip_i}{\omega_0} = \frac{\mu(k)}{\omega(k)} \quad (16)$$

$$\mu_1 = \sum_{i=k+1}^L \frac{ip_i}{\omega_1} = \frac{\mu_T - \mu(k)}{1 - \omega(k)}, \quad (17)$$

где

$$\omega(k) = \sum_{i=1}^k p_i \quad (18)$$

и

$$\mu(k) = \sum_{i=1}^k ip_i. \quad (19)$$

$\omega(k)$ и $\mu(k)$ являются кумулятивными моментами нулевого и первого порядка соответственно, а

$$\mu_T = \mu(L) = \sum_{i=1}^L ip_i \quad (20)$$

является общим средним уровнем изначального изображения. Легко можно доказать, что следующие уравнения справедливы для любого k :

$$\omega_0\mu_0 + \omega_1\mu_1 = \mu_T, \quad \omega_0 + \omega_1 = 1. \quad (21)$$

Дисперсии классов определяются как

$$\sigma_0^2 = \sum_{i=1}^k \frac{(i - \mu_0)^2 p_i}{\omega_0}, \quad (22)$$

$$\sigma_1^2 = \sum_{i=k+1}^L \frac{(i - \mu_1)^2 p_i}{\omega_1}. \quad (23)$$

Оцу показал, что минимизация дисперсии *внутри* класса равносильна максимизации дисперсии *между* классами:

$$\sigma_W^2 = \omega_0 \sigma_0^2 + \omega_1 \sigma_1^2, \quad (24)$$

$$\sigma_B^2 = \omega_0 (\mu_0 - \mu_T)^2 + \omega_1 (\mu_1 - \mu_T)^2 = \omega_0 \omega_1 (\mu_1 - \mu_0)^2, \quad (25)$$

и

$$\sigma_T^2 = \sum_{i=1}^L (i - \mu_T)^2 p_i \quad (26)$$

являющееся внутриклассовой дисперсией, дисперсия между классами и общая дисперсия всех уровней соответственно. Таким образом, алгоритм можно записать как:

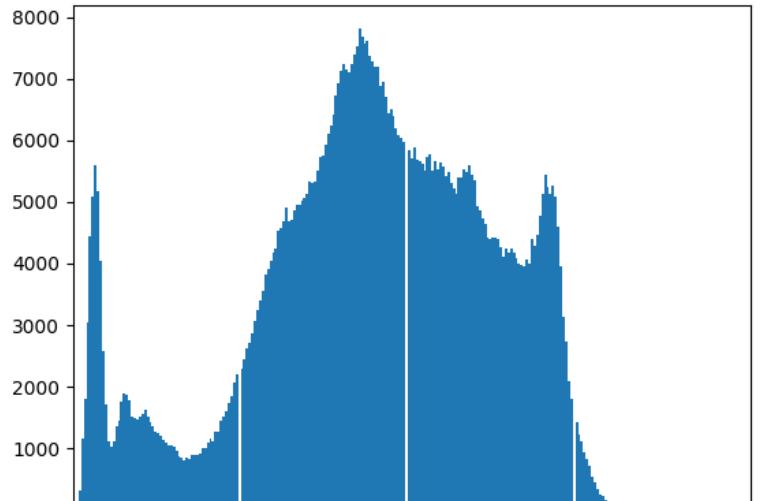
Пусть дано монохромное изображение $G(i, j)$, $i = \overline{1, Height}$, $j = \overline{1, Width}$. Счётчик повторений $k = 0$.

1. Вычислить гистограмму $p(l)$ изображения и частоту $N(l)$ для каждого уровня интенсивности изображения G .
2. Вычислить начальные значения для $\omega_0(0), \omega_1(0)$ и $\mu_1(), \mu_2(0)$.
3. Для каждого значения $t = \overline{1, max(G)}$ – полутона – горизонтальная ось гистограммы:
 - (a) Обновляем ω_0, ω_1 и μ_0, μ_1 .
 - (b) Вычисляем $\sigma_B^2(t) = \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2$.
 - (c) Если $\sigma_B^2(t)$ больше, чем имеющееся, то запоминаем σ_B^2 и значение порога t .
4. Искомый порог соответствует максимуму $\sigma_B^2(t)$.

Диаграммы для изображений изображены на рис. 5. Как видно из рисунков, на данных изображениях довольно проблематично отделить фон от изображения какой-то единственной пороговой величиной, поэтому и алгоритм Оцу на таких изображениях сработает не очень хорошо, как это показано на рис. 6.



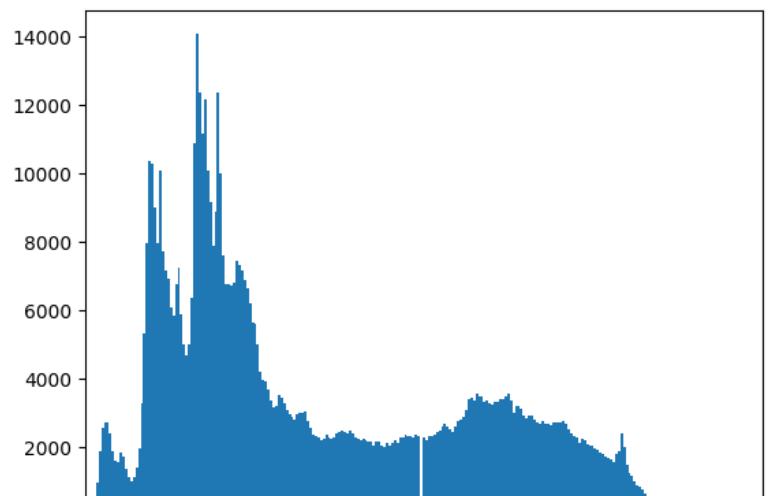
a)



б)

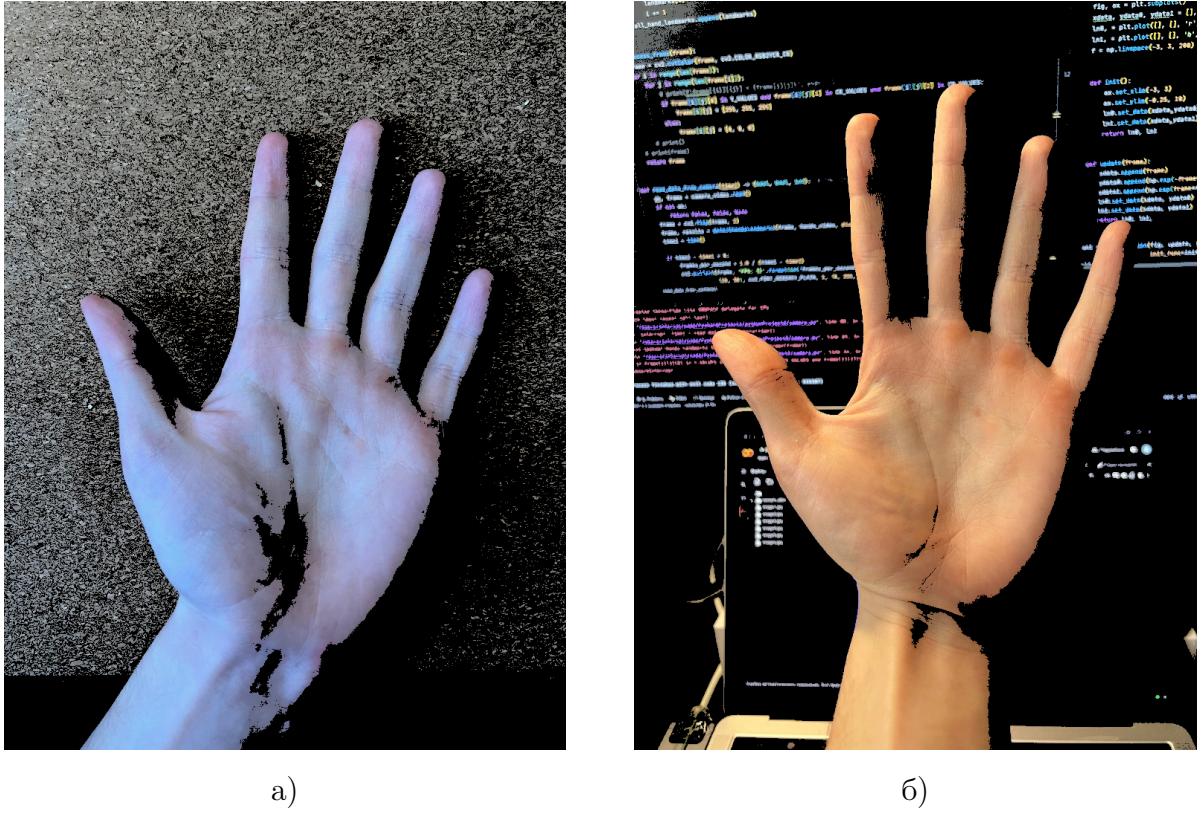


в)



г)

Рис. 5. Изображение 1 и 2 и их гистограммы.



a)

б)

Рис. 6. Результат работы программы с алгоритмом Оцу

1.4 Рекурсивные методы построения модели фона

1.4.1 Visual Background Extractor.

Универсальным и наиболее совершенным на данный момент адаптивным методом, показывающим очень хорошие результаты практически для любых ситуаций и освещений, а также значительно выигрывающим в скорости работы по сравнению с другими алгоритмами, показывающими приблизительно такое же качество поиска движущихся объектов, считается *Visual Background Extractor* [6].

Классическим принципом работы большинства активно использующихся современных адаптивных методов является построение для каждого пикселя кадра функции плотности вероятности. ViBe основывается на принципиально другой идеи: к вопросу об интенсивности (в общем случае о цвете) пикселя $p = (x, y)$ подходят как к вопросу классификации. Для каждого пикселя сохраняется некоторое количество N его предыдущих значений $\nu(p)$, не обязательно последовательных (чаще всего полагают $N = 20$). Тогда можно сказать, что для каждого пикселя текущего кадра будет иметься своя модель C , которая представляется следующим множеством:

$$C(p) = \{\nu_1(p), \dots, \nu_N(p)\} \quad (27)$$

После этого осуществляется классификация пикселя с целью либо выявить движение, либо отнести данный пиксель к фону. В общем случае для этого в цветовом про-

пространстве для пикселя p , значение которого на текущем кадре обозначим через $\nu_n(p)$, где n – номер текущего кадра, строится сфера $S_R(\nu_n(p))$ заранее определённого радиуса R (рис. 7) и определяется количество значений K_n значений из $C_n(p)$, попадающих в эту сферу:

$$K_n(p) = |S_R(\nu_n(p)) \cap C_n(p)|. \quad (28)$$

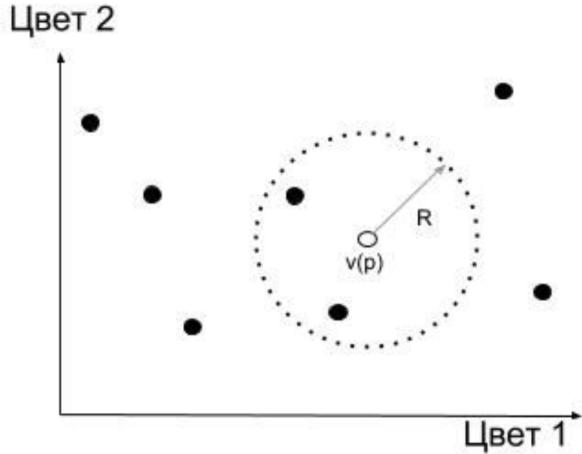


Рис. 7. Сфера в цветовом пространстве

В случае, когда изображения предварительно переводятся в градации серого, значения $\nu(p)$ представляют характеризующий интенсивность серого цвета скаляр, а не вектор, и для вычисления количества K_n "близких" значению $\nu_n(p)$ элементов из $C_n(p)$ необходимо просто осуществить следующую проверку для каждого из данных элементов:

$$|\nu(p) - \nu_n(p)| < R. \quad (29)$$

Решающее правило для текущего кадра строится следующим образом: если $K_n > K_{min}$, где K_{min} обычно принимается много меньше, чем $N/2$, пиксель p на текущем кадре принимается за фоновый, иначе предполагают, что он принадлежит движущемуся объекту. Стоит отметить, что в формулах (??), (29) можно осуществлять проверку не для всех элементов $C_n(p)$, а только до тех пор, пока не найдём K_{min} попадающих в сферу (в общем случае).

После этого для каждого пикселя необходимо обновить его модель. Этот процесс представляет из себя два последовательных шага:

1. Если p был отнесён к фону, из $C_n(p)$ случайным образом выбирается элемент, который будет заменён на $\nu_n(p)$.
2. Из окрестности $O(p)$ размера 3×3 (9 пикселей с учётом p) случайным образом выбирается пиксель, случайный элемент модели которого также будет заменён на $\nu_n(p)$.

Кроме самого процесса обновления модели фона отличительной особенностью ViBe также является процесс её инициализации. Для многих алгоритмов она производится

случайным образом, однако ViBe старается построить наиболее точную модель как можно раньше. С этой целью инициализация модели каждого пикселя производится так:

$$C_0(p) = \{\nu(z), z \in O(p)\}. \quad (30)$$

Здесь пиксель z выбирается N раз случайным образом. Такой процесс позволяет получить достаточно хорошую модель фона уже для второго кадра видео.

Несмотря на все достоинства и простоту реализации, достаточно часто приходится искать альтернативу данному алгоритму, так как он защищен множеством патентов.

1.4.2 Mixture of Gaussians.

Стандартным подходом к построению модели фона, использующимся для многих прикладных задач, является смесь гауссовых распределений (Mixture of Gaussians, MOG) [7, 8]. Как упоминалось выше, чаще всего для каждого пикселя текущего кадра с номером n строится функция плотности вероятности $P_n = P(\nu_n(p))$, и MOG используется именно этот подход. Предполагается, что для каждого пикселя текущего изображения она может быть представлена смесью нормальных распределений, где G – их число в смеси. Обозначим за w_i^n вес распределения Гаусса с номером i , E_i^n – его математическое ожидание и \sum_i^n – среднеквадратичное отклонение. Тогда для P_n получим:

$$P_n = \sum_{i=1}^G w_i^n \cdot N(\nu_n(p)|E_i^n, \sum_i^n). \quad (31)$$

Здесь в общем случае $N(\nu_n(p)|E_i^n, \sum_i^n)$ – многомерное нормальное распределение, имеющее вид:

$$N(\nu_n(p)|E_i^n, \sum_i^n) = \frac{1}{(2\pi)^{H/2} |\sum_i^n|^{1/2}} \exp\left[-\frac{1}{2}(\nu_n(p) - E_i^n)^T \cdot (\sum_i^n)^{-1} \cdot (\nu_n(p) - E_i^n)\right], \quad (32)$$

где

H – число компонентов цвета,

\sum – матрица ковариации.

Для ускорения вычислений, чтобы не вычислять обратную матрицу в формуле (32), в случае цветного изображения предполагается, что для компонентов цвета соблюдаются их независимость, а также что они имеют одинаковое стандартное отклонение. Тогда матрица ковариации примет вид:

$$\sum_i^n = (\sigma_i^n) \cdot E, \quad (33)$$

где E – единичная матрица размерности H .

После того, как для всех пикселей получена смесь, для каждой из них производится сортировка распределений по убыванию величины $\frac{w_i^n}{\sigma_i^n}$. Это делается для того, чтобы

фоновые пиксели отвечали распределению с маленькой дисперсией и большим весом. Тогда число J первых гауссиан, соответствующих распределению цвета фона для пикселя p , будет определяться из условия:

$$J = \arg \min_a \left(\sum_{i=0}^a w_i^n > A \right), \quad (34)$$

где A – некоторый порог.

Решающее правило для текущего кадра выглядит следующим образом: для каждого пикселя определяют, какому из распределений смеси принадлежит его значение $\nu_{n+1}(p)$, используя расстояние Махаланобиса:

$$\sqrt{(\nu_{n+1}(p) - E_i^n)^T \cdot \left(\sum_i^n \right)^{-1} \cdot (\nu_{n+1}(p) - E_i^n)} < 2.5 \cdot \sigma_i^n. \quad (35)$$

После этого возможно два случая:

1. Распределение нашлось. Пиксель будет отнесён к фону, если эта гауссиана является одной из J первых. В противном случае он будет отнесён к движущемуся объекту.
2. Если они одной гауссианы не нашлось, то пиксель также относят к движущемуся объекту.

Теперь рассмотрим, как для пикселя p со значением $\nu_{n+1}(p)$ происходит обновление параметров его модели. Как и при принятии решения, относится ли пиксель к фону или нет, здесь также можно быть два случая. Не существует единого понимания того, каким образом следует вычислять новые параметры, поэтому рассмотрим здесь один из наиболее популярных:

1. Распределение нашлось. Происходит обновление его параметров и веса. Если нашлось более одного распределения, данный процесс выполняется для каждого из них:

$$\begin{aligned} w_i^{n+1} &= (1 - \alpha) \cdot w_i^n + \alpha, \\ E_i^{n+1} &= (1 - g) \cdot E_i^n + g \cdot \nu_{n+1}(p), \\ (\sigma_i^{n+1})^2 &= (1 - g) \cdot (\sigma_i^n)^2 + g \cdot (\nu_{n+1}(p) - E_i^{n+1}) \cdot (\nu_{n+1}(p) - E_i^{n+1})^T, \\ g &= \alpha \cdot N(\nu_n(p) | E_i^n, \sum_i^n). \end{aligned}$$

В этих формулах α – заданная константа.

Для всех остальных распределений в смеси пересчитываются только веса:

$$w_i^{n+1} = (1 - \alpha) \cdot w_i^n.$$

2. Распределение в смеси не нашлось. Тогда самую последнюю в уже отсортированной смеси гауссиану заменяют новой: $E_G^{n+1} = \nu_{n+1}(p)$, дисперсия – максимально возможная, а вес – минимально допустимый.

Для инициализации гауссиан для каждого пикселя чаще всего применяют либо EM-алгоритм (Expectation-maximization algorithm), либо k-means, что достаточно затратно в вычислительном плане. Число входящих в смесь распределений G обычно принимают равным от 3 до 5. Также существует подход, позволяющий автоматически подбирать необходимое количество гауссиан [9].

Результат работы данного метода представлен на рисунке 8.



Рис. 8. Результат отделения фона от изображения с помощью метода MoG.

Рассмотрев наиболее популярные методы отделения кисти человека от фона изображения, можно сделать вывод, что некоторые из методов работают наилучшим образом при специфических условиях, поэтому необходимо их дополнительное исследование.

Заключение

Детектирование рук и распознавание жестов являются одними из наиболее популярных прикладных задач компьютерного зрения, но до сих пор в их решении имеются проблемы с получением точного результата из-за сложности снижения шума.

Во-первых, для решения задачи детектирования кисти были рассмотрены различные подходы, а именно детектирование на основе цветов в разных цветовых моделях (YCbCr и HSV), разделения пикселей изображения на две группы – "фон" и "передняя (полезная) часть" и рекурсивные методы построения модели фона. В результате исследования было получено, что наиболее эффективным способом в статическом изображении является детектирование в цветовой модели HSV, а при наблюдении в динамике, то есть на видеофрагменте, то наилучшие результаты показал метод MoG.

Во-вторых, для построения контура кисти на изображении были исследованы два метода: выделение с помощью оператора Кэнни и топологический структурный анализ цифрового бинарного изображения с помощью отслеживания границ. Последний позволяет находить точные координаты контура, поэтому является наилучшим среди выбранных.

В дальнейшем есть возможность получения конфигурации кисти, например, описывая её "ключевыми" точками, и её применение в большом множестве задач, включая распознавание жестов или управление какими-либо автоматическими системами.

Список литературы

- [1] Dix A., Finlay J., Abowd G.D., Beale R. Human-Computer Interaction. - Third Edition, Pearson Education Limited: 2004. - p. 857
- [2] Jiangqin W., Wen G. A FAst Sign Word Recognition Method for Chinese Sign Language // In Proceedings of the Third International Conference on Advances in Multimodal Interfaces (ICMI '00). - Springer-Verlag, London, 2000. - p.599-606
- [3] Sanna A., Lamberti F., Paravati G., Henao R., Eduardo A., Manuri F. A Kinect-Based Natural Interface for Quadrotor Control // Intelligent Technologies for Interactive Entertainment, Volume 78. Springer Berlin Heidelberg, 2012. - p. 48-56
- [4] Basilio, Jorge & Torres, Gualberto & Sanchez-Perez, Gabriel & Toscano, Karina & Perez-Meana, Hector. Explicit image detection using YCbCr space color model as skin detection, 2011. - p. 123-128.
- [5] N. Otsu. A threshold selection method from gray-level histograms (англ.) // IEEE Trans. Sys., Man., Cyber. : journal. — 1979. — Vol. 9. — p. 62—66.
- [6] M. Van Droogenbroeck, O. Barnich. Vibe: A disruptive method for background subtraction. // In T. Bouwmans, F. Porikli, B. Hoferlin, A. Vacavant, editors, Background Modeling and Foreground Detection for Video Surveillance, chapter 7. Chapman and Hall/CRC, pages 7.1-7.23, July 2014.
- [7] P. Kaewtrakulpong, R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection // Video-Based Surveillance Systems, pp. 135-144. Springer, 2002.
- [8] Z. Zivkovic, F. Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction // Pattern recognition letters, Vol. 27(7), pp. 773-780, 2006.
- [9] Z. Zivkovic, F. Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction // Pattern recognition letters, Vol. 27(7), pp. 773-780, 2006.
- [10] Satoshi S., Keiich A. 1985. Topological Structural Analysis of Digitized Binary Images by Border Following. Computer vision, graphics, and image processing, 30.
- [11] C-H Teh and Roland T. Chin. On the detection of dominant points on digital curves. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 11(8):859–872, 1989.
- [12] Graham R. L. An efficient algorithm for determining the convex hull of a finite planar set //Info. Pro. Lett. – 1972. – Т. 1. – С. 132-133.

- [13] R.A. Jarvis. On the identification of the convex hull of a finite set of points in the plane // Information Processing Letters, Volume 2, Issue 1, 1973, p. 18-21.
- [14] Kirkpatrick, David G.; Seidel, Raimund (1986). "The ultimate planar convex hull algorithm?". SIAM Journal on Computing. 15 (1): 287–299.
- [15] А.В. Носов. Локализация ключевых точек кисти руки на изображении на основе непрерывного скелета. Математические методы моделирования, управления и анализа данных, с. 77-79, 2015.
- [16] Местецкий Л. М. Непрерывная морфология бинарных изображений: фигуры, скелеты, циркуляры. М. : Физматлит, 2009.
- [17] Местецкий Л. М., Рейер И. Непрерывное скелетное представление изображения с контролируемой точностью // International Conference Graphicon. M., 2003. С. 51–54.
- [18] Носов А. В. Алгоритм распознавания жестов рук на основе скелетной модели кисти руки // Вестник СибГАУ. 2014. Вып. 2(54). С. 62–67.