В данном проекте реализована программа, которая получает на вход изображение руки и с помощью библиотеки OpenCV вычисляет количество пальцев на изображении. Программа включает в себя следующие шаги:

1. Преобразование цветового пространства

Исходное цветное изображение нужно перевести в оттенки серого, используя функцию cvtColor. При этом цветовые каналы каждого пикселя преобразуются в один канал со значением яркости (0 представляет чёрный цвет, а 255 - белый). Яркость вычисляется следующим образом:

$$Y = 0{,}299 * R + 0{,}587 * G + 0{,}114 * B$$

2. Гауссовское размытие

Для удаления лишних шумов на изображении применяют функцию GaussianBlur. Уравнение Гаусса для двух измерений:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

где x - расстояние от начала координат по горизонтальной оси, y - расстояние по вертикальной оси, а σ - стандартное отклонение гауссовского распределения. При применении в двух измерениях эта формула дает поверхность, контуры которой являются концентрическими окружностями с распределением Гаусса от центральной точки. Значения этого распределения используются для построения матрицы свертки, которая применяется к исходному изображению. Для каждого нового значения пикселя определяется среднее арифметическое взвешенное в окрестности пикселя. Значение исходного пикселя получает самый тяжелый вес (с самым высоким гауссовым значением), а соседние пиксели получают меньший вес по мере увеличения их расстояния до исходного пикселя.

3. Бинаризация

Функция threshold делит изображение на два класса по порогу яркости. Пиксели с яркостью ниже пороговой окрашиваются в цвет фона (чёрный), выше пороговой – в цвет объекта (белый). Подфункция THRESH_OTSU использует метод Оцу для вычисления порога бинаризации. Суть метода Оцу заключается в том, чтобы выставить порог между классами таким образом, чтобы каждый их них был как можно более «плотным». Если выражаться математическим языком, то это сводится к минимизации внутриклассовой

дисперсии, которая определяется как взвешенная сумма дисперсий двух классов:

$$\sigma_w^2 = w_1\sigma_1^2 + w_2\sigma_2^2$$

где веса w_1 и w_2 – вероятности первого и второго классов соответственно.

4. Нахождение контуров

Контур – это кривая, соединяющая пограничные точки одинакового цвета. Функция findContours определяет все контуры на изображении, поэтому необходимо учесть возможность попадания в область индикации других объектов. С помощью функции contourArea, которая через теорему Грина находит площадь области, ограниченной контуром, выделяем наибольший контур.

5. Построение выпуклой оболочки

Выпуклая оболочка похожа на упрощённое приближение контура, однако не является им. Выпуклое множество - такое, в котором все точки отрезка, образуемого любыми двумя точками данного множества, также принадлежат данному множеству. Области, в которых данное условие не выполняется, называются дефектами выпуклости. Функция convexHull в сущности проверяет контур на дефекты и исправляет их. Для наглядности обрисуем ладонь выпуклой оболочкой, используя функцию drawContours.

6. Вычисление дефектов

С помощью функции convexityDefects дефекты выпуклости могут быть представлены в виде 4-мерного вектора: [начальная точка, конечная точка, крайняя точка дефекта, расстояние между крайней точкой и оболочкой]. Отбирая дефекты по 4 параметру, чтобы он был сопоставим с длиной пальца, находим количество впадин между пальцами на изображении. Добавив к полученному числу единицу, определяем количество пальцев на изображении.