МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В. ЛОМОНОСОВА ФИЛИАЛ В ГОРОДЕ ТАШКЕНТЕ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И
ИНФОРМАТИКИ КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ
И ИНФОРМАТИКИ

Ванесян Роман Грачикович

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

«Оптическое распознавание схем из функциональных элементов»

Научный руководитель,
к.ф.-м.н. _____ Шуткин Ю.С.

«_____» _____ 2020 г.

ТАШКЕНТ - 2020

Содержание

1 Введение

Пусть задано некоторое множество булевых функций

$$B = f_1(x_1, \ldots, x_{n_1}), \ldots, f_s(x_1, \ldots, x_{n_s}),$$

где $n_1, \ldots, n_s \geq 0$.
Будем называть данное множество B *базисом.*

Определение 1. *Схемой из функциональных элементов (СФЭ) над стандартным базисом* $B = \{x_1 \wedge x_2, x_1 \vee x_2, \neg x_1\}$ *будем называть ориентированный граф без циклов* $G = (V, E)$*, для которого выполняются следующие условия:*

- *Каждая вершина* $v \in V$ *имеет полустепень захода* $d(v)$*, не превосходящую двух, то есть* $d(v) \leq 2$*;*
- *Каждая вершина* $v \in V$ *с полустепенью захода, равной 0, называется входной (или входом схемы) и ей приписывается некоторая булева переменная* x_i*;*
- *Существует ровно одна вершина с полустепень захода, равной 1 и приписанной меткой* F*, называемая выходом СФЭ;*
- *Все другие вершины называются внутренними вершинами схемы;*
- *Каждой внутренней вершине* $v \in V$ *с полустепенью захода, равной 1 приписывается (функциональный) элемент отрицания* $(\neg)$*;*
- *Каждой внутренней вершине* $v \in V$ *с полустепенью захода, равной 2 приписывается либо (функциональный) элемент конъюнкции* $(\vee)$*, либо (функциональный) элемент дизъюнкции* $(\wedge)$*.*

Стоит отметить, что существуют СФЭ с базисами отличными от приведенного. Однако, для упрощения изложения в данной работе мы будем рассматривать СФЭ только со стандартным базисом.

При изображении схемы из функциональных элементов входы будем обозначать окружностями, внутри которых записаны

входные переменные x_i. Вершины являющиеся операциями, — треугольниками, внутри которых записаны обозначения соответствующих функций. А выход СФЭ будем помечать прямоугольником, внутри которого записана метка F. Выходы функций будем отмечать выходными ребрами — жордановыми дугами.

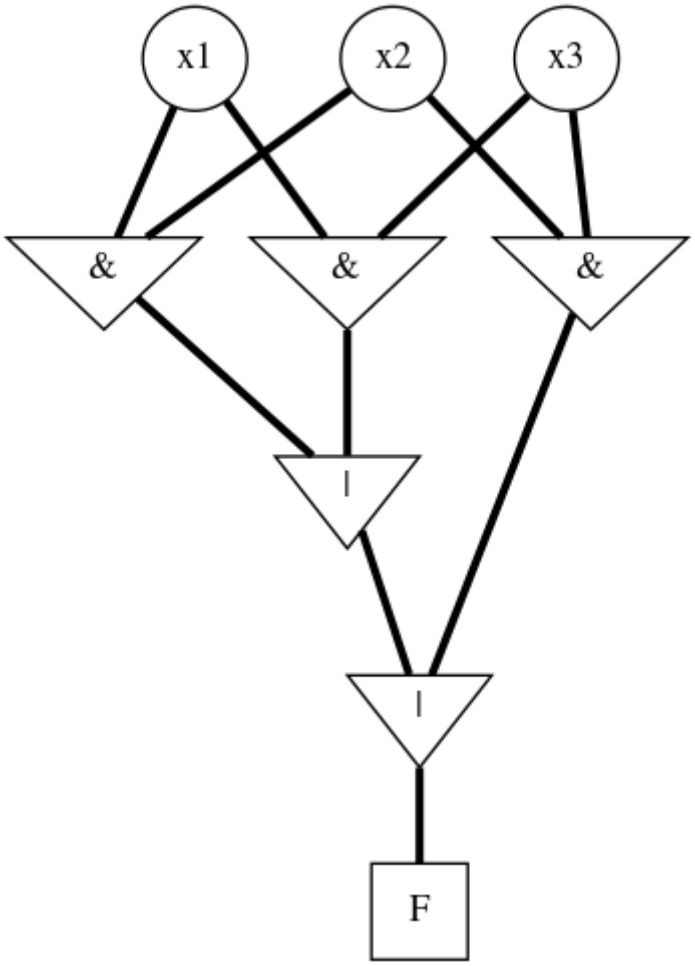

Рис. 1: Пример изображения схемы из функциональных элементов.

2 Постановка задачи

Определение 2. *Цифровым изображением будем называть конечное множество* $I = \{p_i \mid p_i \in C\}$ *на плоскости, где* C *— либо конечное множество кортежей арности* 3*, где каждый элемент имеет значение от* 0 *до* 255*, то есть:*

$$C = \{p_i \mid p_i = (x_{i_1}, x_{i_2}, x_{i_3}), 0 \leq x_{i_j} \leq 255, j = \overline{1,3}\},$$

в таком случае будем называть цифровое изображение трехканальным цифровым изображение; либо конечное множество элементов, значение которых варьируется от 0 *до* 255*, то есть:*

$$C = \{p_i \mid p_i \in \mathbb{Z}, 0 \leq p_i \leq 255\},$$

в таком случае будем называть цифровое изображение одноканальным цифровым изображение (полутоновое изображение). Элементы данного множества p_i *будем называть пикселями.*

Определение 3. *Одноканальное цифровое изображение, для которого множество* C *определено как* $\{0, 1\}$ *будем называть бинарным цифровым изображением (бинарным изображение).*

На вход программе подается цифровое изображение с рисунок схемы из функциональных элементов (СФЭ) на ней. Выходом программы является либо текстовая запись формулы представленной СФЭ, либо ее оцифрованное представление.

3 Оптическое распознавание схем из функциональных элементов

Оптическое распознавание СФЭ будем проводить в 4 этапа: предварительная обработка цифрового изображения, сегментация, распознавание, конечная обработка данных.

3.1 Предварительная обработка

Как и в любой задаче распознавания визуальных образов этап первичной обработки направлен на разделение пикселей на два класса: фоновые пиксели и пиксели образующие исследуемый объект.

Пусть на вход программе было подано трехканальное цифровое изображение. Применяя, к примеру, следующую функцию [1]:

$$f(x_1, x_2, x_3) = \lfloor 0.299 * x_1 + 0.587 * x_2 + 0.114 * x_3 \rfloor$$

к каждому пикселю $p_i = (x_{i_1}, x_{i_2}, x_{i_3})$ преобразуем исходное трехканальное цифровое изображение в одноканальное цифровое изображение. Далее, к полученному полутоновому изображению, применяя алгоритм бинаризации с использованием метода Оцу [2] для нахождения оптимального порога бинаризации за счет минимизации внутриклассовой дисперсии (стоит отметить, что у нас существует всего 2 класса: «фоновые» пиксели и «полезные» пиксели, — пиксели представляющие собой исследуемый объект), получаем бинарное изображение.

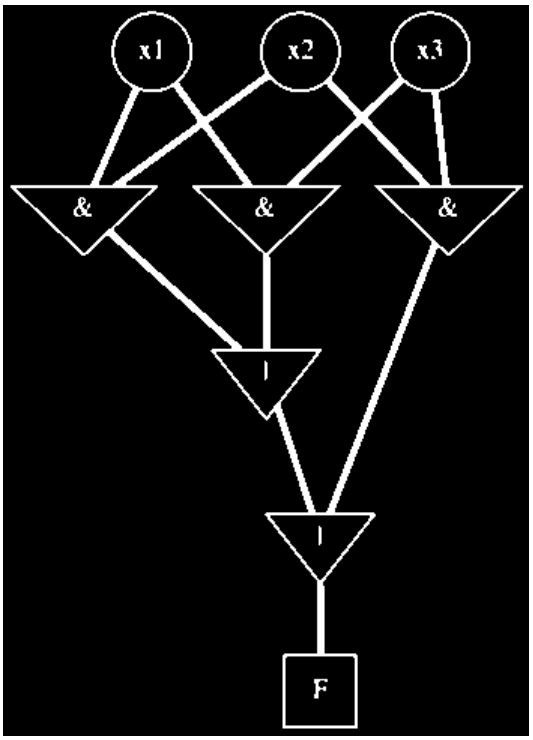

Рис. 2: Пример применения алгоритма бинаризации.

3.2 Сегментация

Входными данными для процесса сегментации является бинаризованное изображение.

Положим D — множество всех пикселей бинаризованного изображения.

Определение 4. *Множество пикселей* $V = \{p_i \mid p_i \in D\}$ *будем считать вершиной, если:*

1. *Пусть* $F \subset V$. *Множество* $H = \{p_i \mid p_i \in F, p_i = 1\}$ *образует одну из следующих фигур: треугольник, окружность, либо прямоугольник.*

2. *Пусть* $G = F \setminus H$. *Существует такое подмножество* $M = \{p_i \mid p_i \in G, p_i = 1\}$ *— метка вершины.*

3. *Никакие две рядом лежащие вершины не расположены так, что пересечение минимальных описывающих прямоугольников, содержащих соответствующие вершины есть множество не пустое, то есть:*
   $\forall V_i, V_j, \quad V_i \neq V_j, i \neq j : P(V_i) \cap P(V_j) = \varnothing.$

Исходя из определения вершин, мы можем построить алгоритм для нахождения таковых в бинарном изображении. Применим алгоритм для нахождения замкнутых контуров [3] к данному бинарному изображению. Выходом алгоритма является множеству A с занумерованными элементами — последовательности наборов координат, однозначно задающими фигуры образованные различными замкнутыми контурами. Стоит отметить, что для проверки **условий 2, 3** нас интересуют минимальные прямоугольники коллинеарные осям (в английской литературе axis-aligned minimum bounding box). Любой прямоугольник может быть задан двумя точками: левой верхней и правой нижней точками (относительно начала координат). То есть, положим p_1 - левая верхняя, а p_2 - правая нижняя точки, тогда:

$$\{p_1 = (min(x), max(y)), p_2 = (max(x), min(y))\},$$

где минимум и максимум соответствующих координат берутся по всем наборам задающим описываемую фигуру. Применяя данный алгоритм к каждому элементу множества A мы получим множество минимальных описывающих прямоугольники.

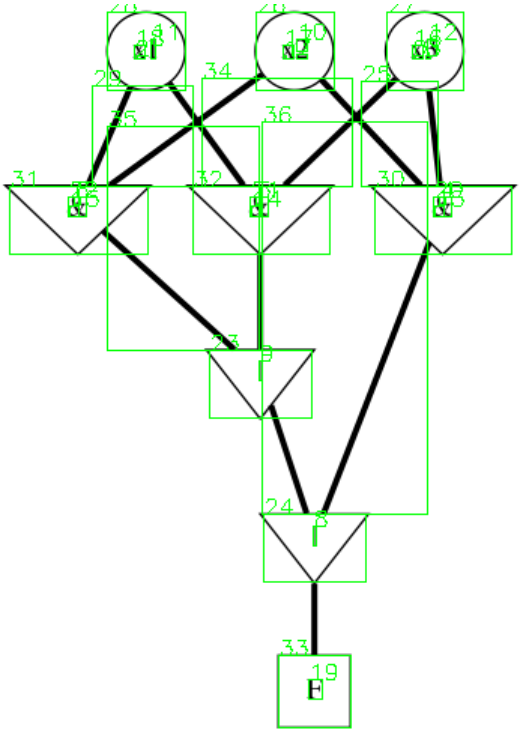

Рис. 3: Пример результата работы алгоритма построения минимальных описывающих прямоугольников.

Проверим условие 3. Положим

$$s(p_{11}, p_{12}, p_{21}, p_{22}) = \begin{cases} 1, & (p_{11} \leq p_{21} \leq p_{22} \leq p_{12}) \quad \wedge \\ & \qquad \wedge \quad (p_{11} \leq p_{21} \leq p_{22} \leq p_{12}) \\ 0, & otherwise \end{cases},$$

где (p_{11}, p_{12}) и (p_{21}, p_{22}) пары левой-верхней и правой-нижней точек, задающих соответствующие прямоугольники. Правило порядка для точек задано покоординатно.

Применим попарно правило $s(p_{11}, p_{12}, p_{21}, p_{22})$ к элементам множества описывающих минимальных прямоугольников соответствующих вершин. Если для какой-то пары прямоугольников правило $s(p_{11}, p_{12}, p_{21}, p_{22})$ дало значение 1, то исключим фигуру описанную прямоугольником с большей площадью из множества вершин.

Проверим условие 2. Положим

$$c(p_{11}, p_{12}, p_{21}, p_{22}) = \begin{cases} 1, & (p_{11} \leq p_{21} \leq p_{12}) \wedge (p_{11} \leq p_{22} \leq p_{12}) \\ 0, & otherwise \end{cases},$$

где аналогично (p_{11}, p_{12}) и (p_{21}, p_{22}) пары левой-верхней и правой-нижней точек, задающих соответствующие прямоугольники.

Применяя попарно указанное правило к множеству минимальным описывающих прямоугольником однозначно проверим условие 2. То есть, если найдется такой минимальный описывающий прямоугольник не содержащий никаких других прямоугольников, то исключим фигуру вписанную в данный прямоугольник из множества вершин.

Проверим условие 1. Для проверки условия 1, для каждого описывающего минимального прямоугольника пометим все пиксели входящие во внутренность описываемой фигуры цветом 1 (то есть присвоим им значение равное 1), все остальные пиксели прямоугольника - цветом 0. Размер каждого такого изображение есть $w \times h$, где w — ширина, а h — высота. Построим 3 бинарных изображения размера $w \times h$ с рисунками: окружности с радиусом $r = \frac{min(w,h)}{2}$ и центром в точке $(\frac{w}{2}, \frac{h}{2})$, прямоугольника с шириной w и высотой h, треугольника заданного координатами $\{(0, h), (\frac{w}{2}, 0), (w, h)\}$. Аналогично фоновые пиксели пометим цветом 0, а внутренность с контуром каждой из фигур в цвет 1.

Очевидно, что полученные изображения могут быть заданы матрицами размерности $w \times h$.

Положим

$$\delta(A, B) = \sum_{i=1}^{h} \sum_{j=1}^{w} a_{ij} \oplus_2 b_{ij}, \quad a_{ij} \in A, b_{ij} \in B$$

Пусть матрица A задает изображение вершины. Тогда будем считать, что на матрице A изображена одна из трех фигур (окружность, прямоугольник, треугольник), если сумма $\delta(A, B)$ - минимальна и не превышает некоторого заданного числа $k \in \mathbb{N}$.

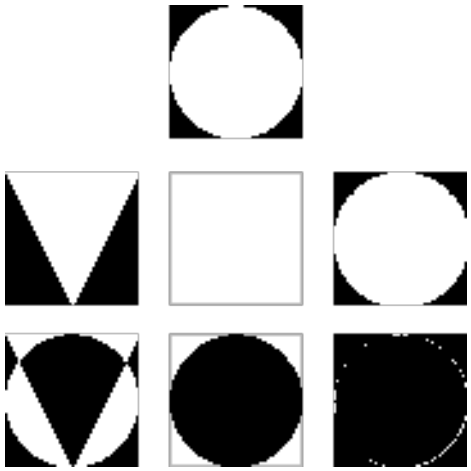

Рис. 4: Пример результата работы алгоритма проверки условия 1.

Таким образом, последовательно применяя алгоритмы проверки условия 1-3 к каждому элементу множества фигур, получим множество искомых вершин СФЭ.

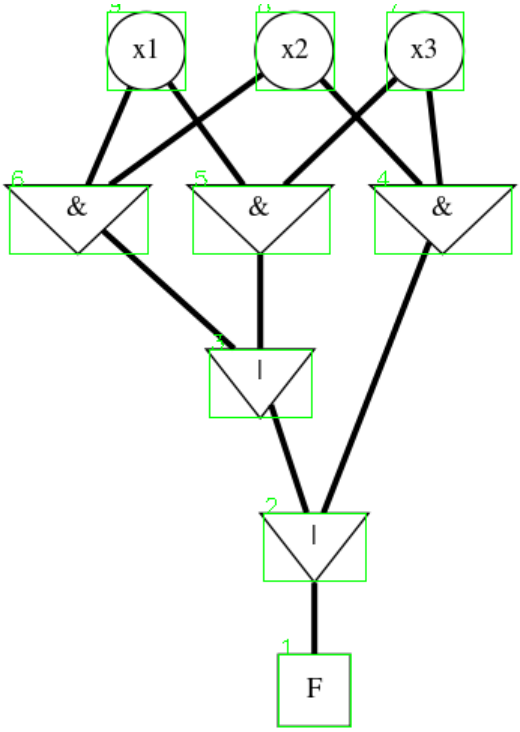

Рис. 5: Пример результата работы алгоритма выделения вершин СФЭ.

Выходом алгоритма сегментации является бинарное изображение с фигурами вершин.

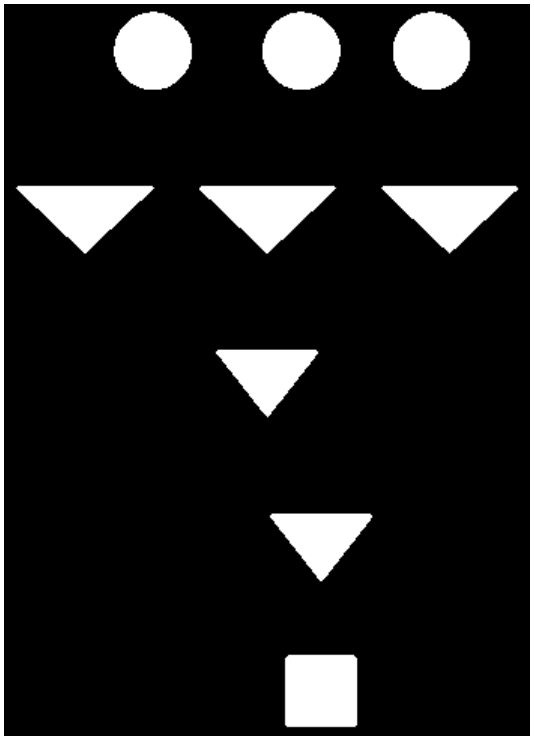

Рис. 6: Пример выхода алгоритма сегментации СФЭ.

3.3 Распознавание

Этап распознавания разобьём на два подэтапа: распознавание меток в вершинах СФЭ и топологическое распознавание СФЭ.

3.3.1 Распознавание меток

Выходными данными алгоритма примененного на этапе сегментации является бинарное изображение с фоновыми пикселями, имеющими значение 0 и внутренностями вершин с контуром, имеющими значения 1. Таким образом, на самом деле, данное изображение является еще и маской для исходного изображения. А стало быть и для его бинаризованного вида. Положим M — множество пикселей маски, а B — множество пикселей бинаризованного изображения. Положим множество L — результат применения операцию конъюнкции к каждым пикселям $p_{1i} \in D$ и $p_{2i} \in M$, то есть

$$L = \{p_i \mid p_i = p_{1i} \wedge p_{2i}, p_{1i} \in D, p_{2i} \in M\}.$$

L — бинарное изображение содержащее вершины и их метки заданной СФЭ.

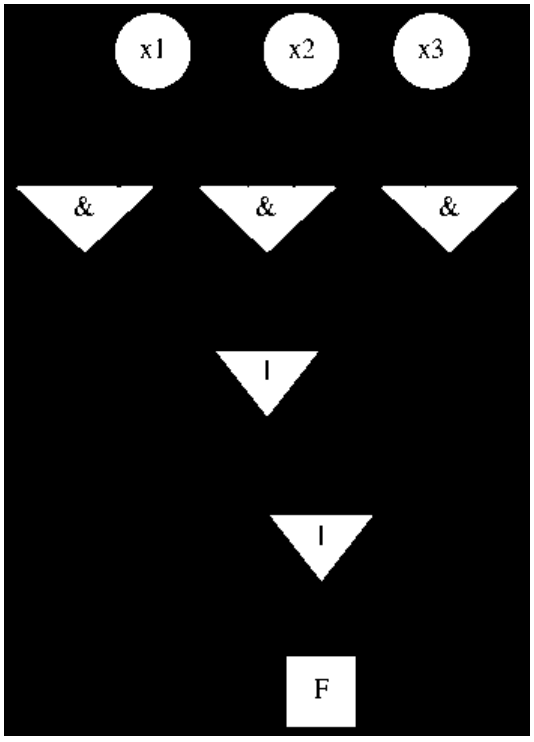

Рис. 7: Пример изображения полученного в результате наложения маски на бинаризованное изображения содержащее СФЭ.

Далее, будем рассматривать каждую вершину по отдельности. Возьмем минимальный прямоугольник содержащий данную фиксированную вершину с меткой, обозначим его через r_1. Такой прямоугольник содержит два класса пикселей: фоновые и полезные пиксели. Для удаления фоновых пикселей, инвертируем пиксели в прямоугольнике содержащем множества пикселей вершин без метки обозначим, такое множество за r_2. Применим попиксельно операцию дизъюнкции к r_1 и r_2. Получили множество пикселей, где фоновые пиксели имеют значение 1, а пиксели образующие метку — значение 0.

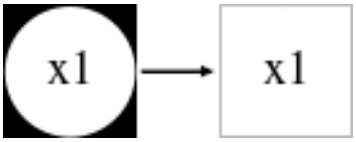

Рис. 8: Пример результата работы алгоритма выделения метки.

Далее, к полученной изображению, применяя, к примеру, алгоритм распознавания текста основанный на нейронной сети с долгой краткосрочной памятью (в английской литературе LSTM based OCR) [4, 5] получим текстовую запись метки данной вершины.

Таким образом, применяя приведенный алгоритм к каждой вершине СФЭ, получим все текстовые записи меток.

3.3.2 Топологическое распознование СФЭ

Определение 5. *Ребром будем называть жорданову дугу образованную последовательностью точек $p_i = 1$ и соединяющее вершины $v_i, v_j, i \neq j$.*

Входными данными для топологического распознавания СФЭ является изображение полученное в результате дизъюнкции бинаризованное изображения СФЭ с выходом алгоритма примененного на этапе сегментации — бинарным изображением, содержащим исключительно вершины СФЭ.

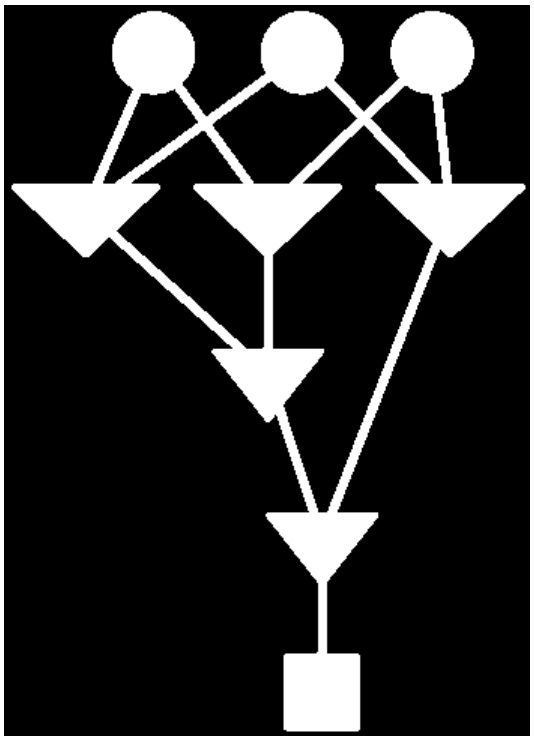

Рис. 9: Пример входного изображения для алгоритма топологического распознавания СФЭ.

Алгоритм топологического распознавания СФЭ приведенный в данной работе полностью аналогичен алгоритму топологического распознавания графа в работе [6]. Приведем краткое описание идеи данного алгоритма. Алгоритм топологического распознавания графа состоит из 3 частей: *построение скелета графа*, *классификация пикселей (точек) графа*, *обход ребер графа*.

Рассмотрим процесс построение скелета графа. Идея построения скелета объекта заключается в удалении пикселей из множества пикселей изучаемого объекта таким образом, чтобы не нарушать топологию самого объекта. Результатом данной операции будет являться *скелет* объекта. Важным свойством процесса построения скелета объекта является сохранение свойства связности пикселей объекта. Это свойство как раз и позволяет нам использовать данный процесс при топологическом распознавании графа.

Построение скелета графа основано на применении алгоритма предложенного в статье [7]. Идея данного алгоритма заключается в изменении значений пикселей с 1 на 0 контура объекта до тех пор, пока связность объекта не будет нарушена.

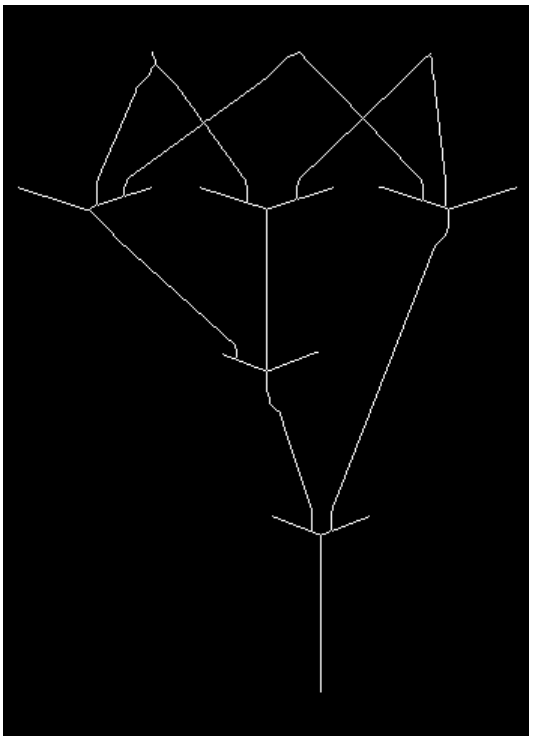

Рис. 10: Пример результата работы построения скелета графа изображенного на рис. 9.

Рассмотрим процесс классификации пикселей графа. Будем классифицировать каждый пиксель изображения как пиксель относящийся к одному из следующих классов: P_v — класс пикселей входящих в одну из вершин графа (будем называть такой класс пикселей *классом «порт» пикселей*), P_e — класс пикселей представляющих собой пиксели одного из ребер графа, P_c класс пикселей включающих в себя пиксели на пересечении двух ребер (такой класс пикселей будем называть *классом «кросс» пикселей*) и P_b — класс *класс «фоновых» пикселей*, — пиксели не входящие ни в один из приведенных ранее классов.

Определение 6. *Будем называть 4-окрестностью пикселя p_i с координатами (x, y) множество состоящее из пикселей с координатами $(x-1, y), (x+1, y), (x, y-1), (x, y+1)$.*

Напомним, что результатом построения скелета объекта изображенного на бинарном изображении является бинарным изображение.

Так как ранее, на этапе сегментации, мы однозначно выделили множество пикселей образующих вершины (с их внутренностью), и скелет объекта включает так же данные пиксели (некоторое их

подмножество), то мы можем однозначно отнести данные пиксели к классу P_v.

Пусть P_r множество пикселей 4-окрестности пикселя p_i. Положим

$$n_0(p_i) = \sum_{p_j \in P_r} p_j.$$

Будем считать, что пиксель p_i относится к классу P_b, если $n_0(p_i) \leq 1$, иначе если $n_0(p_i) = 2$, то p_i лежит в классе P_e, иначе при $n_0(p_i) \geq 2$ пиксель лежит в классе P_c.

Таким образом, применяя изложенный алгоритм классификации к каждому пикселю получим 4 класса. При том, ни один из классов не содержит пиксели другого класса.

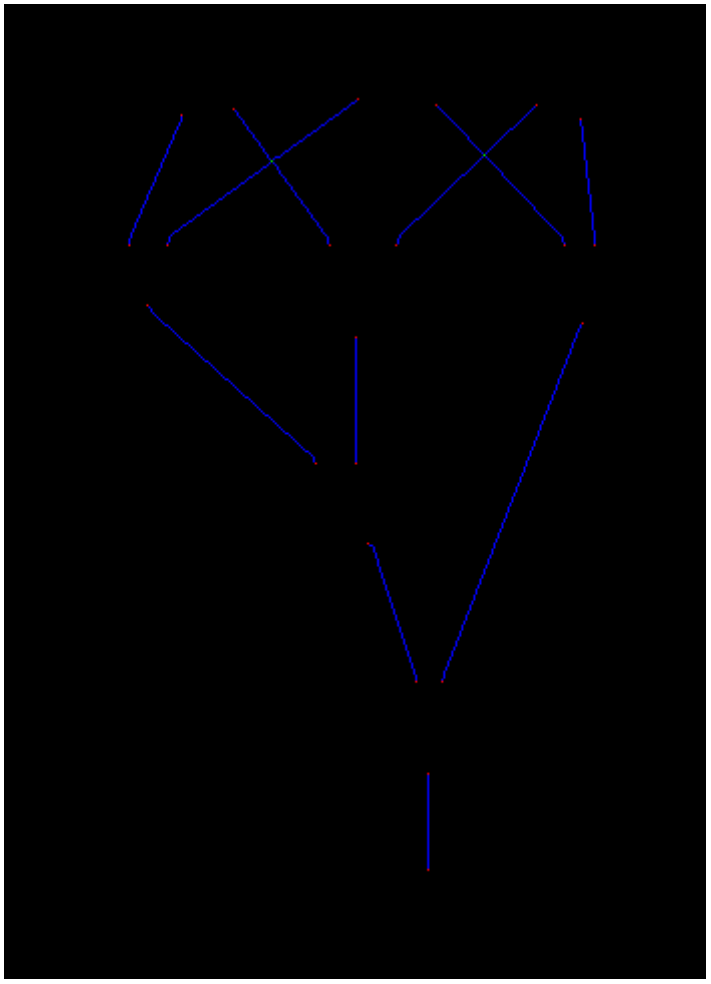

Рис. 11: Пример работы алгоритма классификации пикселей графа. Синим цветом выделены пиксели принадлежащие классу P_e, зеленым — к классу P_c, красным — к классу P_v, а черным — к классу P_b

4 Организация работы программы

Программа для реализации алгоритма распознавания СФЭ приведенного в данной работе была написана на языке Python 3. Для реализации промежуточных алгоритмов на стадиях бинаризации и сегментации была использована библиотека OpenCV 4 [8]. Для реализации алгоритма распознавания символов была использована библиотека Tesseract 4 [5].

Входом программы является трехканальное цифровое изображение.

Для интерпретации полученных данных была написана вспомогательная библиотека с реализацией структуры для построения Binary Expression Tree [9], и последующего его обхода, используя алгоритм поиска в глубину [10] с применением шаблона проектирования «посетитель» (в английской литературе Visitor pattern) [11]. Таким образом, данная библиотека позволяет интерпретировать Binary Expression Tree построенное из полученных данных как в графическом представлении (то есть интерпретирует в цифровое изображение), так и в текстовой записи.

Список литературы

[1] https://en.wikipedia.org/wiki/Luma_(video)#Rec._601_luma_versus_Rec._709_luma_coefficients

[2] N. Otsu. *A threshold selection method from gray-level histograms. IEEE Trans. Sys., Man., Cyber. : journal. — 1979. — Vol. 9. — P. 62—66.*

[3] Suzuki, S. and Abe, K., *Topological Structural Analysis of Digitized Binary Images by Border Following. CVGIP 30 1, pp 32-46 (1985)*

[4] Adnan Ul-Hasan *Generic Text Recognition using Long Short-Term Memory Networks* https://kluedo.ub.uni-kl.de/frontdoor/deliver/index/docId/4353/file/PhD_Thesis_Ul-Hasan.pdf

[5] https://tesseract-ocr.github.io/

[6] Ch. Auer, Ch. Bachmaier, F. Gleißner, and J. Reislhuber *Optical Graph Recognition, section 3.3, p. 6-9.* https://link.springer.com/content/pdf/10.1007%2F978-3-642-36763-2_47.pdf

[7] T. Y. Zhang, C.Y. Suen *A Fast parallel algorithm for thinning digital petterns.* http://agcggs680.pbworks.com/f/Zhan-Suen_algorithm.pdf

[8] https://opencv.org/opencv-4-0/

[9] https://en.wikipedia.org/wiki/Binary_expression_tree

[10] https://en.wikipedia.org/wiki/Tree_traversal#Depth-first_search

[11] https://en.wikipedia.org/wiki/Visitor_pattern