Алгоритм сравнения отпечатков пальцев на основе поиска максимального пути в графе

А. В. Поляков, И. М. Ковалев

В статье представлен новый алгоритм верификации отпечатков пальцев на основе поиска максимального пути в графе. Центральной идеей данного подхода является поиск максимального пути в специальным образом сконструированном ациклическом графе. Средняя скорость работы алгоритма верификации составляет 100 сравнений в секунду (Intel Core i5-2500 CPU @3.30 GHz 3.30GHz, 4 Гб ОЗУ, ОС Windows 7).

Ключевые слова: отпечатки пальцев, верификация, минуции, граф, максимальный путь в графе.

Введение

В статье [1] был предъявлен алгоритм верификации отпечатков пальцев на основе структуры созвездий. В настоящей статье будет предъявлен новый алгоритм верификации личности по отпечаткам пальцев, основанный на поиске максимального пути в графе. Напомним прежде основные предварительные сведения из статьи [1].

Аутентификация личности по отпечаткам пальцев является наиболее распространенной и надежной биометрической технологией. Биометрическая система аутентификации в зависимости

от контекста применения может работать в двух режимах: верификации (сравнение один к одному) и идентификации (сравнение один ко многим). Задача верификации состоит в подтверждении личности, в то время как задачей идентификации является установление личности субъекта доступа. В данной статье мы будем заниматься задачей верификации.

А именно, пусть даны два отпечатка пальцев I и T, требуется построить алгоритм сравнения этих отпечатков, который на выходе выдавал бы меру схожести отпечатков, и на основе этой информации биометрическая система выносила бы решение о принадлежности этих отпечатков одному человеку или разным людям.

В настоящее время разработано множество алгоритмов сравнения отпечатков пальцев, которые условно можно разбить на три класса [2]: корреляционное сравнение (вычисление взаимной корреляции двух изображений отпечатка [3]), сравнение минуций (точек обрыва и бифуркаций папиллярных линий [4], [5]), сравнение потоков папиллярных линий и папиллярных узоров [6]. Наиболее часто применяется подход, основанный на сравнении минуций.

Минуцией (или точкой Гальтона) называется особая точка папиллярной линии, как правило, обрыв линии или раздвоение линии (бифуркация). В этом случае задача верификации отпечатков пальцев сводится к задаче сравнения двух множеств точек, т.е. поиску такого геометрического преобразования (как правило, аффинного), переводящего максимальное количество точек из первого отпечатка в точки второго.

Несмотря на большое количество существующих биометрических алгоритмов сравнения отпечатков пальцев, задача верификации личности по опечаткам пальцев остается сложной задачей и по-прежнему актуальна из-за большой изменчивости представления отпечатков пальцев, которая обусловлена целым рядом факторов (сдвиги и повороты пальца на сканирующем устройстве, состояние кожи, давление пальца на сканирующее устройство, разрешение сканирующего устройства, шум, ошибки, возникающие при экстракции признаков).

Ежегодно в мире публикуются тысячи статей (по данным Google Scholar, с 2014 года по настоящее время было опубликовано 8100 статей [7]) по верификации отпечатков пальцев, с 2000 по 2006 год проводились соревнования Fingerprint Verification Competition (организаторами которого являлись университет Болонии (Италия), Мичиганский государственный университет (США), университет Сан-Хосе (США), Мадридский университет (Испания)) [8], [9].

Настоящая статья организована следующим образом: введена математическая модель отпечатка пальца, поставлена задача верификации отпечатков пальцев по их шаблонам, предъявлен алгоритм верификации отпечатков пальцев, оценена его вычислительную сложности по времени, а также указаны результаты точности работы алгоритма на двух тестовых базах: тестовой базе FVC2002 DB1 (100 изображений отпечатков пальцев) и тестовой базе FTdb (1055 изображений отпечатков пальцев) (в терминах EER (equal error rate) – значения, при котором вероятность ошибки первого рода равна вероятности ошибки второго рода) в сравнении с предложенным в статье [1] Astr-алгоритмом.

Математическая модель отпечатка пальца

Введем на изображении отпечатка пальца декартову систему координат, начало координат которой находится в левом нижнем углу изображения, ось абсцисс направлена горизонтально слева направо, ось ординат направлена вертикально снизу вверх. Представим отпечаток пальца в виде нумерованного списка минуций (точек обрыва и бифуркаций папиллярных линий), каждая из которых имеет следующие характеристики: координаты $x, y \in \mathbb{N}$, угол направления θ в минуции типа «обрыв» между локальным направлением папиллярной линии в минуциальной точке и осью абсцисс, в случае минуции типа «бифуркация» определение угла направления минуции сводится к предыдущему случаю инвертированием цветов исходного изображения.

Далее, на множестве минуций введем две метрики: евклидово расстояние между двумя минуциями $m_i = (x_i, y_i, \theta_i)$и $m_j = (x_j, y_j, \theta_j)$ $\rho_{eu}(m_i, m_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ и гребневый счет $\rho_{RC}(m_i, m_j)$, определяемый как количество папиллярных линий, пересекающих отрезок с концами в m_i и m_j без учета конечных точек отрезка (m_i, m_j).

Пусть $EU_I = (eu_{ij}) - n \times n$-матрица, в которой на $(i, j)-$м месте хранится значение $\rho_{eu}(m_i, m_j)$, и пусть $RC_I = (rc_{ij}) - n \times n$-матрица, в которой $rc_{ij} = \rho_{RC}(m_i, m_j)$,тогда модель отпечатка пальца имеет вид

$$I = \{(x_i, y_i, \theta_i), EU_I, RC_I\}, i = \overline{1, n}$$

.

Постановка задачи

Пусть даны два шаблона отпечатка пальцев $I = \{(x_i, y_i, \theta_i), EU_I, RC_I\}, i = \overline{1, n}$, и $T = \{(x'_j, y'_j, \theta'_j), EU_T, RC_T\}, j = \overline{1, m}$, требуется построить алгоритм сравнения этих отпечатков, который на выходе выдавал бы меру схожести отпечатков $s \in [0, 1]$, и на основе этой информации биометрическая система выносила бы решение о принадлежности этих отпечатков одному человеку или разным людям.

Алгоритм верификации отпечатков пальцев на основе поиска максимального пути в графе (FingerPath-алгоритм)

Вход алгоритма:

$I = \{(x_i, y_i, \theta_i), EU_I, RC_I\}, i = \overline{1, n}$ - реализация математической модели образца отпечатка пальца, полученного со сканирующего устройства, $T = \{(x'_j, y'_j, \theta'_j), EU_T, RC_T\}, j = \overline{1, m}$ - реализация математической модели шаблона отпечатка пальца, взятого из базы данных.

Выход алгоритма: 30-компонентный вектор признаков:

12_12_Min1 – количество минуций в образце

12_12_Min2 – количество минуций в шаблоне

12_12_L – максимальный путь в графе (определение графа дано ниже)

12_12_S1 – площадь преобразованного образца

12_12_S2 – площадь шаблона

12_12_I – площадь пересечения при наложении образца на шаблон

12_12_corr – корреляция областей пересечения при наложении образца на шаблон

12_12_IMin1 – количество минуций образца в пересечении с шаблоном (при наложении образца на шаблон)

12_12_IMin2 – количество минуций шаблона в пересечении с образцом при наложении образца на шаблон

12_21_S1 –площадь образца

12_21_S2 – площадь преобразованного шаблона

12_21_I – площадь пересечения при наложении шаблона на образце

12_21_corr – корреляция пересечения при наложении шаблона на образец

12_21_IMin1 – количество минуций образца в пересечении с шаблоном (при наложении на образец)

12_21_IMin2 - количество минуций образца в пересечении с шаблоном (при наложении шаблона на образец)

Физический смысл следующих параметров аналогичен соответствующим предыдущим и получается при перестановке местами образца и шаблона:

21_12_Min1

21_12_Min2

21_12_L

21_12_S1

21_12_S2

21_12_I

21_12_corr

21_12_IMin1

21_12_IMin2
21_21_S1
21_21_S2
21_21_I
21_21_corr
21_21_IMin1
21_21_IMin2

Шаг 1. Для шаблонов I и T строим таблицы ориентированных в конечных точках отрезков. $Table(I) = I \times I = (x_{k1}, y_{k1}, x_{k2}, y_{k2}, \theta_{k1}, \theta_{k2})$, $Table(T) = T \times T = (x'_{k1}, y'_{k1}, x'_{k2}, y'_{k2}, \theta'_{k1}, \theta'_{k2})$, Обозначим через $a_i = Table(I)(i)$ – i-й отрезок (i-ю строку) в таблице Table(I), через $b_j = Table(T)(j)$ – j-й отрезок (j-ю строку) в таблице Table(T).

Шаг 2. Зададим метрические и угловые пороги: $eu_{tresh} = 15$ пикселей, и $\varepsilon_{tresh} = \frac{\pi}{9}$, $RC_{tresh} = 1$, $\varepsilon_{chain} = \frac{\pi}{9}$(численные значения порогов были выбраны экспериментально).

Шаг 3. Строим ориентированный граф G, вершины которого - это пары отрезков вида $(Table(I)(i), Table(T)(j))$ такие, что выполнены следующие условия:

$$|\sqrt{(x_{i1} - x_{i2})^2 + (y_{i1} - y_{i2})^2} - \sqrt{(x'_{j1} - x'_{j2})^2 + (y'_{j1} - y'_{j2})^2}| < eu_{tresh}$$

$$\min\left(|\theta_{i1} - \theta'_{j1}|, 2\pi - |\theta_{i1} - \theta'_{j1}|\right) < \varepsilon_{thresh}$$

$$\min\left(|\theta_{i2} - \theta'_{j2}|, 2\pi - |\theta_{i2} - \theta'_{j2}|\right) < \varepsilon_{thresh}$$

Две вершины (a_i, b_j) и (a_l, b_m) графа G соединяются ребром (по направлению из (a_i, b_j) в (a_l, b_m)) тогда и только тогда, когда отрезки a_i и a_l имеют один общий конец, одновременно отрезки b_j и b_m имеют один общий конец, а угол поворота от a_i к a_l должен с точностью до ε_{chain} совпадать с углом поворота от b_j к b_m.

Для того, чтобы граф G был ациклическим, накладываем дополнительное условие: x-координаты пары отрезков с общей вершиной образуют строго возрастающую последовательность $x_1 < x_2 < x_3$,

Утверждение 1. Граф G - ациклический

Доказательство:

от противного. Пусть G- циклический граф, и в нем содержится цикл длины k. Занумеруем вершины цикла: $v_{i_1} = (a_{i_1}, b_{i_1}), \ldots, v_{i_k} = (a_{i_k}, b_{i_k})$. Тогда x- координата второй вершины отрезка a_{i_k} совпадает с x-координатой первой вершины отрезка a_{i_1}, что противоречит условию строгой монотонности последовательности x-координат пар отрезков.

Утверждение доказано.

Шаг 4. Будем искать в графе G путь Chain максимальной длины. Для этого:

1) Проводится топологическая сортировка графа G. Дан ориентированный граф с N вершинами и M рёбрами. Требуется перенумеровать его вершины таким образом, чтобы каждое рёбро вело из вершины с меньшим номером в вершину с большим. Для решения задачи используется алгоритм обхода графа в глубину [10]. Поскольку граф ацикличен, то решение существует.

2) Для каждой вершины графа ищем максимальный путь с концом в этой вершине. Длина этого пути записывается в массив dist. В массив back записывается предпоследняя вершина этого пути.

3) Выбираем из всех вершин вершину с максимальным путем (поиск максимума в массиве dist) и восстанавливаем путь, в котором хранятся номера предыдущих вершин, при помощи массива back.

```
typedef std::vector<int> vec;
  typedef typename boost::graph_traits<Graph>::
      in_edge_iterator EdgeIterator;
  using boost::topological_sort;
  using boost::num_vertices;
  using boost::source;
  using boost::in_edges;

  vec reversed;
  topological_sort(g, std::back_inserter(reversed));
  const int vn = num_vertices(g);
  vec dist(vn,0);
  vec back(vn,-1);

  for (vec::reverse_iterator it = reversed.rbegin(); it
      != reversed.rend(); ++it) {
    EdgeIterator ei, ei_end;
    int t = *it;
    for (boost::tie(ei,ei_end) = in_edges(t,g); ei !=
        ei_end; ++ei) {
      int s = source(*ei,g);
      if (dist[s]+1 > dist[t]) {
        dist[t] = dist[s]+1;
        back[t] = s;
      }
    }
  }
  vec::iterator p = std::max_element(dist.begin(), dist.
      end());
  if (p == dist.end())
    return;


  for (int v = p-dist.begin(); v != -1; v = back[v])
    path_.push_back(v);
  std::reverse(path_.begin(),path_.end());
  res_ = *p;
```

Шаг 5. Методом наименьших квадратов строим аффинное преобразование, переводящее множество точек максимального пути в первом шаблоне в множество точек максимального пути во втором шаблоне.

Шаг 6. Применяем аффинное преобразование координат к множеству минуций из первого шаблона С помощью преобразования, полученного на пятом шаге, получаем образ первого

отпечатка в системе координат, связанной со вторым отпечатком, и пересекаем его с изображением второго отпечатка.

Шаг 7. При помощи масок первого и второго изображения отпечатков пальцев строим область пересечения отпечатков, находим ее площадь и площади отпечатков:

```
cv::Mat1b CompareResults::intersectionMask(
    const cv::Mat1b &mask1,
    const cv::Mat1b &mask2,
    const cv::Mat1d &t21)
{
  cv::Mat1b intersection_mask = cv::Mat::zeros(mask1.rows
      , mask1.cols, CV_64F);
  for (int i = 0; i < mask2.rows; ++i) {
    for (int j = 0; j < mask2.cols; ++j) {
      point z = transformPoint(t21, j, i);
      polygon s(5, z); // (z,z,z,z,z)
```

Берем окрестность Шеннона точки (крест)

```
      ++s[1].first;
      --s[2].first;
      ++s[3].second;
      --s[4].second;\\
```

далее для каждой точки из s выполняется следующее: если черный пиксель на второй маске при преобразовании переходит в черный пиксель первой маски, то он добавляется в маску пересечения

```
      std::for_each(s.begin(), s.end(),
                    [&intersection_mask, &mask1, &mask2,
                        i, j](point &q)
      {
        if (q.first >= 0 && q.first < mask1.cols &&
            q.second >= 0 && q.second < mask1.rows &&
            mask2(i,j) == 255 && mask1(q.second,q.first)
                == 255)
          intersection_mask(q.second,q.first) = 255;
      });
    }
  }
  return intersection_mask;
}
```

Шаг 8. Вычисление корреляции областей пересечения и количество минуций в пересечении для образца и для шаблона.

Шаг 9. Вычисление скоринга:

$$s_1 = \frac{2Chain}{12\_12\_Min1 + 12\_12\_Min2}$$

Замечание. На практике оказалось, что значения приведенных ниже параметров не оказывают существенное влияние на точность верификации, при этом их отсутствие существенно снижает время вычислений: 12_12_S1 – площадь преобразованного образца

12_12_S2 – площадь шаблона

12_12_I – площадь пересечения при наложении образца на шаблон

12_12_corr – корреляция областей пересечения при наложении образца на шаблон

12_12_IMin1 – количество минуций образца в пересечении с шаблоном (при наложении образца на шаблон)

12_12_IMin2 – количество минуций шаблона в пересечении с образцом при наложении образца на шаблон

12_21_S1 –площадь образца

12_21_S2 – площадь преобразованного шаблона

12_21_I – площадь пересечения при наложении шаблона на образце

12_21_corr – корреляция пересечения при наложении шаблона на образец

12_21_IMin1 – количество минуций образца в пересечении с шаблоном (при наложении на образец)

12_21_IMin2 - количество минуций образца в пересечении с шаблоном (при наложении шаблона на образец

12_21_IMin1 – количество минуций образца в пересечении с шаблоном (при наложении на образец)

12_21_IMin2 - количество минуций образца в пересечении с шаблоном (при наложении шаблона на образец

21_12_S1

21_12_S2

21_12_I

21_12_corr
21_12_IMin1
21_12_IMin2
21_21_S1
21_21_S2
21_21_I
21_21_corr

В связи с этим, работу алгоритма можно оптимизировать, перейдя от 5-го шага сразу к 9-му.

Утверждение 2. Пусть $m < n$. тогда вычислительная сложность оптимизированного FingerPath-алгоритма в среднем составляет $\mathcal{O}(n^4)$операций

Доказательство: Доказательство ведется прямым вычислением сложности алгоритма на каждом шаге:

Шаг 1. Вычисление декартова произведения требует $\mathcal{O}(n^2)$ операций.

Шаг 2. Инициация пороговых значений параметров требует константного времени.

Шаг 3 Задание графа G в худшем случае (полный граф) требует $\mathcal{O}(n^4)$ на задание вершин и $\mathcal{O}(n^8)$ на ребра. Отметим, что худший случай (случай полного графа) на практике не реализуется.

Шаг 4 Топологическая сортировка графа проводится за $\mathcal{O}(n+m)$ операций.

Шаг 5: Метод наименьших квадратов: $\mathcal{O}(n^3)$ операций.

Шаг 6. Вычисление скоринга занимает константное время.

Таким образом, общая вычислительная сложность FingerPath-алгоритма в среднем составляет $\mathcal{O}(n^4)$ операций.

Утверждение доказано.

Экспериментальные результаты

Для программной реализации алгоритма была выбрана объектно-ориентированная парадигма разработки программного

обеспечения. Программная реализация алгоритма была написана Иваном Ковалевым на языке C++. Выбор языка программирования обусловлен следующими факторами:

1) Выделение шаблонов отпечатков пальцев производилось с помощью фреймворка NIST Biometric Image Software (NBIS), написанного на C [11];

2) Использования библиотеки OpenCV, включающей поддержку численных методов;

3) Использование библиотеки Boost для распараллеливания вычислений при обсчете базы данных отпечатков пальцев;

4) Было написано приложение для верификации с графическим пользовательским интерфейсом в Qt (фреймворк C++).

Характеристики используемой ЭВМ: Intel Core i5-2500 CPU @3.30 GHz 3.30GHz, 4 Гб ОЗУ, ОС Windows 7.

Был проведен эксперимент на двух базах базе отпечатков: FVC2002 DB1b (всего 80 отпечатков, 10 пальцев по 8 изображений каждого пальца), а также собственной базы данных FTDB (всего 1055 отпечатков, 211 отпечатков по 5 изображений каждого).

Было проведено 3160 сравнений на базе FVC2002 DB1b (из них 280 сравнений для своих отпечатков пальцев и 2880 – для чужих), и 555985 сравнений на базе FTDB (из них 2110 сравнений для своих отпечатков пальцев, и 553875 для чужих отпечатков пальцев).

В процессе сравнения были использованы следующие скоринги:

Максимальная длина цепи в графе, $Chain = 12\_12\_L$;

Корреляция, $corr = 12\_12\_corr$;

Стандартный скоринг, $s_1 = \frac{2Chain}{12\_12\_Min1+12\_12\_Min2}$;

Модифицированный скоринг, $s_2 = \frac{2Chain}{12\_12\_IMin1+12\_12\_IMin2}$;

Удельная площадь пересечения, $Inter = \frac{12\_12\_I}{\max(12\_12\_S1,12\_12\_S2)}$.

	Chain	Cor	s_1	s_2	Inter
EER	0.08	0.29	0.04	0.06	0.27

Таблица 1. Значения EER на различных скорингах

Алгоритм	**База данных**	**EER, %**
Предложенный	FVC2002 DB1_b	1.7
Astr-алгоритм	FVC2002 DB1_b	3
NIST BOZORTH3	FVC2002 DB1_b	3.6
Предложенный	FTdb	4
Astr-алгоритм	FTdb	7.4
NIST BOZORTH3	FTdb	5.3

Таблица 2. Сравнение точности FingerPath–алгоритма с Astr-алгоритмом

По данным, представленным в таблице 1 было принято решение использовать в алгоритме стандартный скоринг s_1. Скорость сравнения отпечатков пальцев методом поиска максимальной цепи в графе составила 100 отпечатков в секунду.

Результаты точности работы алгоритма и сравнение с Astr-алгоритмом представлены в таблице 2. EER (equal error rate) – значение, при котором вероятность ошибки первого рода равна вероятности ошибки второго рода.

Заключение

Представленный в статье алгоритм может быть включен в качестве компоненты в состав программно-аппаратного комплекса, предназначенного для биометрической верификации личности. Отметим, что данный подход обладает большей точностью, чем Astr-алгоритм, но при этом и асимптотическая вычислительная сложность FingerPath-алгоритма выше.

Список литературы

[1] Поляков А.В. Алгоритм сравнения отпечатков пальцев на основе структуры созвездий //Программная инженерия - 2015-№ 8 с.26-31

[2] D. Maltoni, D. Maio, A. K. Jain. Handbook of fingerprint recognition, Springer, 2003

[3] Bazen et al. (2000). Bazen A.M., Verwaaijen G.T.B., Gerez S.H., Veelenturf L.P.J. and van der Zwaag B.J. A Correlation-Based Fingerprint Verification System // Proc. Workshop on Circuits Systems and Signal Processing (ProRISC 2000), 2000.

[4] J. Starink, E. Backer. Finding point correspondences using simulated annealing"// Pattern Recognition, vol. 28, no. 2, pp. 231-240, 1995

[5] George Bebis, Taisa Deaconu, Michael Georgiopoulos. Fingerprint Identification Using Delaunay Triangulation, 1999

[6] Ito et al. (2006). Ito K., Morita A., Aoki T., Nakajima H., Kobayashi K. and Higuchi T. A Fingerprint Recognition Algorithm Combining Phase-Based Image Matching and Feature-Based Matching,// Proc. Int. Conf. on Biometrics, LNCS 3832, pp. 316–325, 2006

[7] Список статей, посвященных верификации по отпечаткам пальцев, в системе «Google Scholar». URL : $https$: $//scholar.google.ru/scholar?as_ylo = 2014\&q = fingerprint + verification\&hl = ru\&as_sdt = 0,5$ (дата обращения 8.04.2015 г.)

[8] Система тестирования алгоритмов верификации «FVC- onGoing» лаборатории биометрических систем Болонского университета (Италия), URL: $https$: $//biolab.csr.unibo.it/FVCOnGoing/UI/Form/Home.aspx$ (дата обращения 8.04.2015 г.)

[9] B. Dorizzi, R. Cappelli, M. Ferrara. Fingerprint and On-Line Signature Verification Competitions at ICB 2009//proc. International Conference on Biometrics (ICB), Alghero, Italy, pp.725-732, June 2009

[10] T. Cormen, C. Leiserson, R. Rivest. Introduction to algorithms (second edition), Williams, pp. 622-632, 2005

[11] NIST Biometric Image Software:$http$: $//www.nist.gov/itl/iad/ig/nigos.cfm$