Сегментация кровеносных сосудов на коронарной ангиограмме

Шмелева Екатерина
МЭСИ, гр. ДКО-121б
Москва 2014

Была поставлена задача изучить и реализовать алгоритм сегментации кровеносных сосудов на коронарной ангиограмме и создание обучающей выборки. Для выполнения задачи использовалась открытая библиотека компьютерного зрения OpenCV.

1. Выделение сосудов на ангиограмме

1.1. Построение согласованного фильтра

Предлагаемый алгоритм выделения сосудов заключается в свертке исходного изображения с целью повышения контрастности между пикселями, принадлежащими и не принадлежащими кровеносным сосудам.

В обработке изображений свертка изображения маской предполагает движение маски по всем пикселям изображения. Пусть I – изображения размера $M \times N$, а g – свертывающая маска размера $p \times p$, где p – нечетное целое число. Тогда для каждого пикселя (i,j) изображения I отклик фильтра вычисляется как

$$G(x,y) = \sum_{k=-a}^{a} \sum_{l=-a}^{a} g(k,l) I(i+k, j+l),$$

где $a = \frac{p-1}{2}$ и G – результирующее изображение.

Кровеносные сосуды обычно характеризуются малой кривизной, поэтому могут быть аппроксимированы некоторыми линейными сегментами.

Отражательная способность сосудов ниже, чем у других объектов, поэтому на ангиограмме они темнее фона. При этом границы этих сосудов практически никогда не бывают идеальными. Хотя профиль яркости меняется от сосуда к сосуду, изменение незначительно, и любой такой профиль может быть аппроксимирован кривой Гаусса

$$g(x,y) = -\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{d^2}{2\sigma^2}},$$

где d – расстояние между точкой (x,y) и прямой, проходящей через центр кровеносного сосуда, σ – половина предполагаемой ширины сосудов. Функция Гаусса берется с отрицательным знаком, так как сосуды темнее фона.

Предполагаем, что интенсивность фона имеет постоянное значение. В случае, когда пиксель, к которому применяется свертывающая маска, принадлежит фону, ожидаемое значение отклика для этого пикселя равно нулю. Таким образом, из каждого элемента маски должно быть вычтено среднее значение всех элементов маски m.

Ширина сосудов не является постоянной величиной, поэтому фильтр применяется для нескольких значений σ. Для каждого пикселя сохраняется только максимальный отклик.

При конструировании фильтра для двумерных изображений необходимо также учитывать, что сосуды могут проходить под любым углом $\theta \in [0, \pi)$, поэтому свертывающую маску необходимо повернуть на некоторое конечное число углов. После того, как все углы рассмотрены, для каждого пикселя также сохраняется только максимальный отклик.

Получаем, что для всех пикселей (i,j) исходного изображения I сохраняется только максимальный отклик фильтра среди всех углов θ и параметров σ:

$$G(x,y) = \max_{\theta,\sigma} \left| \sum_{k=-a}^{a} \sum_{l=-a}^{a} g(k,l) I(i+k, j+l) \right|.$$

1.2. Реализация алгоритма

Пусть $\bar{p} = [x, y]$ – некоторая дискретная точка свертывающей маски, а L – длина линейного сегмента, на котором сосуд, предположительно, имеет постоянную ориентацию. Определим прямоугольник $\mathrm{N} = \{(\mathrm{x}, \mathrm{y}) | |x| \leq 3\sigma, |y| \leq \mathrm{L}/2\}$. Веса в свертывающей маске K_i вычисляются как

$$K_i(x, y) = -\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \quad \forall \overline{p_i} \in N.$$

```
for (int y = 0; y < height; y++)
       for (int x = 0; x < width; x++)
              kernel[(y + verticalOffset) * diameter + (x + horizontalOffset)] =
-(1.0 / ((float)sigma * sqrt(2*PI))) * exp(-pow((x + horizontalOffset) - zero, 2) /
 (2 * pow(sigma, 2)));
```

Пусть A определяет количество точек $\bar{p}$ в прямоугольнике N. Тогда среднее значение свертывающей маски определяется как:

$$m_i = \sum_{\overline{p_i} \in N} \frac{K_i(x, y)}{A}.$$

```
float m = sum / (width * height);
```

Отсюда, $K'_i(x, y) = K_i(x, y) - m_i \quad \forall \overline{p_i} \in N.$

```
for (int y = 0; y < height; y++)
       for (int x = 0; x < width; x++)
              kernel[(y + verticalOffset) * diameter + (x + horizontalOffset)] -= m;

_matKernel = Mat(diameter, diameter, CV_32F, kernel);
```

В двумерном пространстве поворот можно описать одним углом θ_i со следующей матрицей линейного преобразования:

$$\overline{\overline{r_i}} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix}.$$

Поворот свертывающей маски $K'_i(x, y)$ на углы $\theta_i \in [0, \pi)$ осуществляется с помощью функций библиотеки OpenCV.

```
Point2f center(_zero, _zero);
// Вычисляет матрицу поворота.
Mat rotationMatrix = cv::getRotationMatrix2D(center, angle, 1.0);
// Выполняет аффинное преобразование.
cv::warpAffine(kernel, rotated, rotationMatrix, _matKernel.size());
```

Для свертки исходного изображения используется метод `filter2D`.

```
cv::filter2D(source, destination, -1, _matKernel);
```

Результирующее изображение определяется как максимум откликов от всех фильтров.

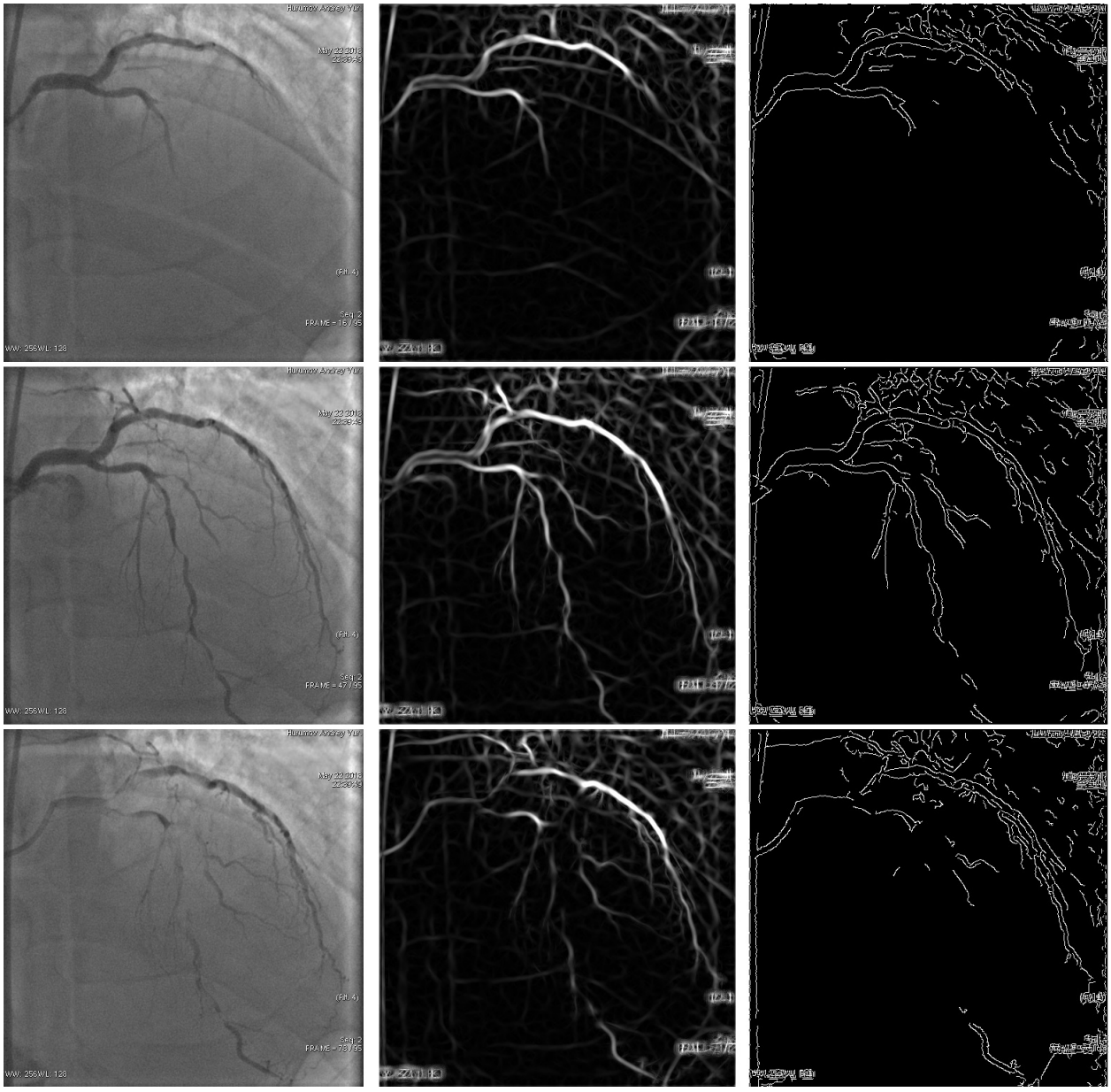
```
for (int i = 0; i < _numberOfSigmas * _numberOfAngles; i++)
       result = cv::max(result, _intermediateResults[i]);
```

Перед обработкой изображения производится шумопонижение.

```
cv::fastNlMeansDenoising(source, source, 4.0, 7, 9);
```

1.3. Результаты

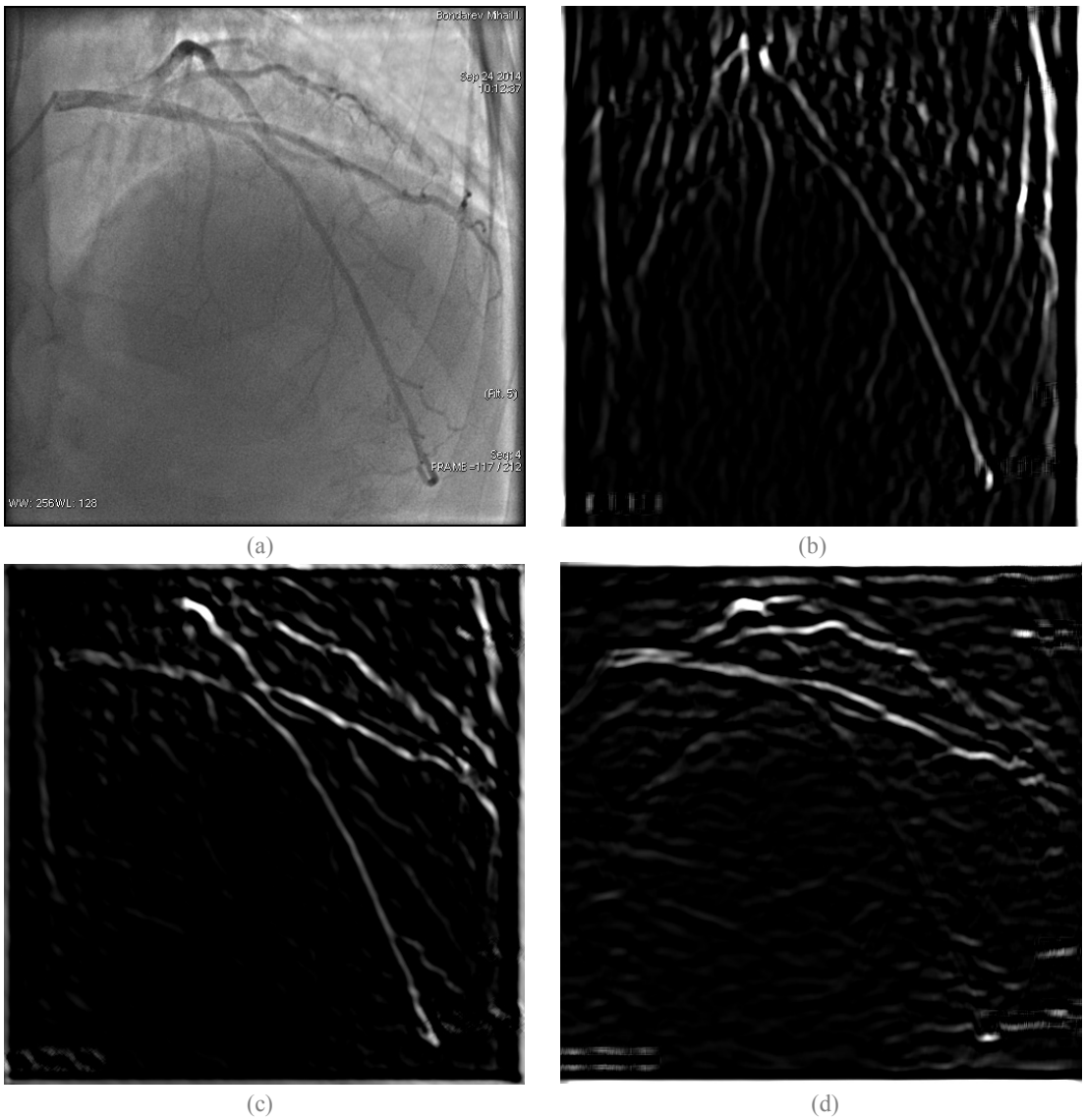
Ангиограмма, результат применения построенного фильтра и детектора границ Кэнни даны на *рисунке 1*. Исходное изображение было предварительно подвергнуто процессу шумоподавления с помощью метода нелокального усреднения. Были выбраны параметры $\sigma \in [1,3]$ и $\mathrm{L} = 15$.



(a) (b) (c)

Рис. 1: Результат применения фильтра. (a) Последовательность кадров ангиограммы размерности 512×512 пикселей. (b) Отфильтрованная последовательность кадров. (c) Детектор границ Кэнни.

Свертывающая маска применялась под 12 различными углами $\theta \in [0, \pi)$. Сосуды, лежащие в пределах $\pm 7{,}5°$ от текущего угла поворота свертывающей маски также дают хороший отклик. На *рисунке 2* отображены отклики фильтра при различных углах свертывающей маски и $\sigma = 3$.

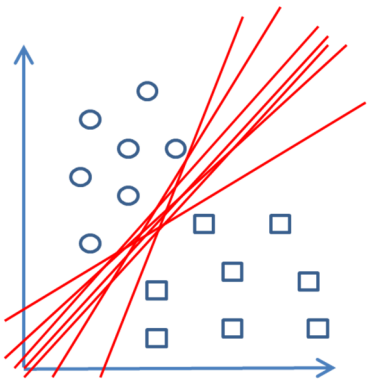


(a) (b)

(c) (d)

Рис. 2: (a) Исходное изображение. (b) Свертка исходной маской.
(c) Свертка маской с углом поворота в 45°. (d) Свертка маской с углом поворота в 90°.

2. Метод опорных векторов



Метод опорных векторов (*Support Vector Machine, SVM*) представляет набор схожих алгоритмов обучения с учителем, использующихся для задач классификации данных. При помощи данного метода обычно решаются задачи *бинарной классификации*.

Каждый объект данных представляется как точка в p-мерном пространстве. Каждая из этих точек принадлежит только одному из двух классов. Основная идея метода опорных векторов состоит в переводе исходных векторов в пространство более высокой размерности, определении возможности разделения точек *гиперплоскостью* размерностью $(p-1)$ и поиске разделяющей гиперплоскости с максимальным зазором в этом пространстве.

В случае, если два множества точек могут быть полностью отделены гиперплоскостью, они называются *линейно разделимыми*.

Две параллельных гиперплоскости строятся по обеим сторонам гиперплоскости, разделяющей наши классы. *Оптимальной разделяющей гиперплоскостью* будет гиперплоскость, максимизирующая расстояние до двух параллельных гиперплоскостей. Алгоритм работает в предположении, что чем больше разница или расстояние между этими параллельными гиперплоскостями, тем меньше будет средняя ошибка классификатора.

Метод выделяет *опорные вектора*, то есть объекты, лежащие на границах множеств.

Решение задачи бинарной классификации при помощи метода опорных векторов заключается в поиске некоторой линейной функции, которая правильно разделяет набор данных на два класса.

Пусть имеется обучающая выборка $(x_1, y_1), \dots, (x_m, y_m), x_i \in \mathbb{R}^n, y_i \in \{-1,1\}$. Метод опорных векторов строит классифицирующую функцию $F(x) = sign(\langle w, x\rangle + b)$, где w — нормальный вектор к разделяющей гиперплоскости, b — вспомогательный параметр. Объекты, для которых $F(x) = 1$, попадают в один класс, а объекты с $F(x) = -1$ — в другой.

Нормальный вектор w и параметр b выбираются таким образом, чтобы максимизировать расстояние $1/\|w\|$ до каждого класса. Проблема нахождения максимума расстояния эквивалентна нахождению минимума $\|w\|^2$:

$$\begin{cases} \arg\min\limits_{w,b} \|w\|^2, \\ y_i(\langle w, x\rangle + b) \geq 1, \mathrm{i} = 1, \dots, \mathrm{m}. \end{cases}$$

Данная задача является стандартной задачей квадратичного программирования.

Точность классификатора зависит, в частности, от выбора симметричной неотрицательно определенной функции, называемой *ядром*, а также правильного подбора его параметров. Наиболее распространёнными являются следующие ядра:

- линейное ядро – $k(x, x') = \langle x, x'\rangle$;
- полиномиальное ядро – $k(x, x') = (\langle x, x'\rangle + const)^d$;
- радиальная базисная функция – $k(x, x') = \exp(-\gamma\|x - x'\|^2)$ для $\gamma > 0$;
- радиальная базисная функция Гаусса – $k(x, x') = \exp(-\|x - x'\|^2/2\sigma^2)$;
- сигмоид – $k(x, x') = \tanh(k\langle x, x'\rangle + c)$ для $k > 0$ и $c < 0$.

Machine Learning Library (MLL) библиотеки OpenCV содержит классы и функции для работы с SVM. Поддерживается несколько типов ядер, включая упомянутые выше линейное (`SVM::LINEAR`), полиномиальное (`SVM::POLY`), радиальную базисную функцию или радиальную базисную функцию Гаусса (`SVM::RBF`), сигмоид (`SVM::SIGMOID`) и другие.

```
Ptr<SVM> svm = SVM::create();
svm->setType(SVM::C_SVC);
svm->setKernel(SVM::POLY);
svm->setGamma(3);
svm->setDegree(2);
```

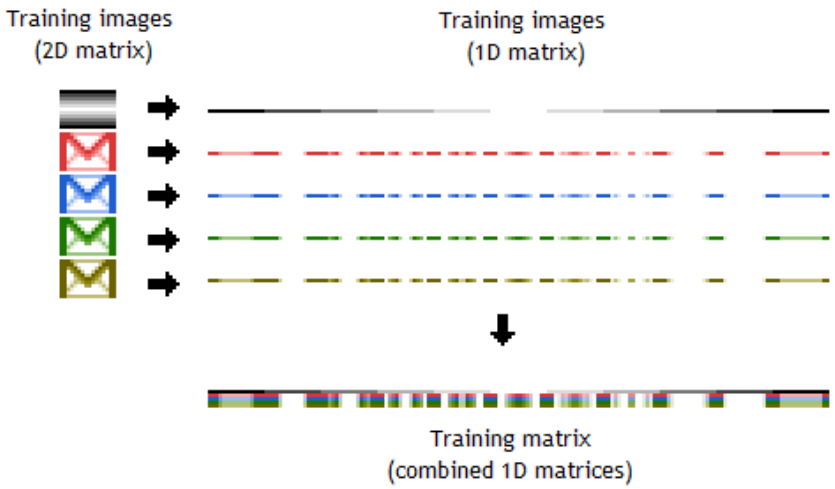
Обучение модели производится с помощью метода `train`:

```
TrainingMatrix* training = new TrainingMatrix("norm.txt", "pathalogy.txt", filter);
Ptr<TrainData> trainingData =
       TrainData::create(training->GetTrainingMatrix(),SampleTypes::ROW_SAMPLE,
       training->GetLabels());

svm->train(trainingData);
```

Каждая строка матрицы входных признаков `trainingMatrix` соответствует одному изображению, каждому элементу строки соответствует цвет пикселя в определенной точке. Так как изображение представляет собой двумерную матрицу, его необходимо преобразовать к одномерному виду. Длина каждой строки обучающей матрицы равняется площади изображения, все изображения должны иметь равный размер.

В библиотеке OpenCV возможен как описанный выше вариант с построчным хранением векторов (`SampleTypes::ROW_SAMPLE`), так и с их хранением по столбцам (`SampleTypes::COL_SAMPLE`).



Для каждой ангиограммы тренировочное изображение представляет собой последовательность из n равноудаленных отфильтрованных кадров (*см. рис. 1(b)*).

Каждому обучающему изображению ставится в соответствие его класс. Для этого объявляется одномерная матрица значений целевого признака `labels` типа `CV_32S` или `CV_32F`, каждый элемент которой сопоставляется с соответствующей строкой обучающей матрицы.

```
trainingMatrix = Mat(numberOfTrainingImages, trainingImageLength,
CV_32FC1, trainingMatrixVector.data());

labels = Mat(numberOfTrainingImages, 1, CV_32SC1, labelsVector.data());
```

Предсказание SVM-модели производится с помощью функции `predict`, получающей в качестве параметров матрицу-строку, содержащую признаки классифицируемого изображения и матрицу, в которую сохраняется отклик.

```
FrameSequence* temp = new FrameSequence(TEST, filter);
vector<byte> data = temp->ToByteSequence();
Mat input = Mat(1, data.size(), CV_32F, data.data());

Mat response;
svm->predict(floatInput, response);
cout << response;
```

Возможно сохранение и загрузка алгоритма из файла.

```
svm->save("database.xml");
svm->load("database.xml");
```

Заключение

В процессе выполнения поставленной задачи мной были изучены принципы построения свертывающих фильтров и реализована программа для создания обучающей матрицы входных признаков для классификации пациентов с патологиями и без на основе коронарных ангиограмм. В рамках программы реализован алгоритм выделения сосудов на изображении. Фильтрация выполнена на основе метода, предложенного в [1] и [2], суть которого состоит в:

- выборе параметра σ в зависимости от ширины сосудов на ангиограмме;
- построении свертывающих масок на основе функции Гаусса для каждого значения σ;
- выборе углов поворота свертывающей маски и получения дополнительных масок поворотом исходной на заданный угол θ;
- свертке исходного изображения каждой из масок;
- максимизации полученных результатов.

Список литературы и Интернет-ресурсов

1. S. Chaudhuri, S. Chatterjee, N. Katz, M. Nelson, M. Goldbaum, ”Detection of blood vessels in retinal images using two-dimensional matched filters” // IEEE Trans on Med Imaging, Vol 8, No. 3, – IEEE, 1989 – С. 263-269.
2. A. Cavinato, “Spline-based refinement of contours in binarymaps of retinal vessels” – University of Padova, 2012 – 30 с.
3. Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, “A Practical Guide to Support Vector Classification” – Department of Computer Science and Information Engineering, National Taiwan University, 2003 – 16 с.
4. OpenCV API Reference [Электронный ресурс] // OpenCV Documentation URL http://docs.opencv.org (дата обращения: 29.06.2015).