УДК 004.932.72'1

ОБРАБОТКА ИЗОБРАЖЕНИЙ С ПОМОЩЬЮ БИБЛИОТЕКИ OPENIMAJ

А.В. ВОЙТОВ
(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)

Рассматриваются возможности библиотеки OpenIMAJ.

Иногда возникают задачи, при которых необходимо анализировать изображения в своих программах. Иногда необходимо найти лицо, иногда некий предмет. Для решения подобных задач может помочь библиотека OpenIMAJ.

OpenIMAJ - это удостоенный наград набор библиотек и инструментов для анализа мультимедийного контента и создания контента. OpenIMAJ очень широк и содержит все: от современного компьютерного видения (например, дескрипторы SIFT, обнаружение выявляемого региона, распознавание лиц и т. п) И расширенную кластеризацию данных до программного обеспечения, которое выполняет анализ содержимого, компоновки и структуры веб-страниц.

OpenIMAJ доступна для языка java и представляет собой множество Jar модулей, кажды из который решает разные задачи, такие как обработка изображений, обработка видео обнаружение лиц и т. п.

Для обнаружения лиц OpenIMAJ предоставляет интерфейс FaceDetector с методом detectFaces, который принимает изображения и возвращает список с областями где он обнаружил лица. Основными реализациями данного интерфейса являются HaarCascadeDetector и FKEFaceDetector.

HaarCascadeDetector использует алгоритм признаков Хаара для нахождения лиц. Пример использования данного класса представлен в листинге 1. Конструктор данного класса принимает размер минимального квадрата в пикселях, которым может быть лицо. Алгоритм признаков хаара работает с одним каналом изображения и поэтому используется метод calculateIntensity чтобы привести картинку к градациям серого.

```
     FaceDetector<DetectedFace,FImage> fd = new HaarCascadeDetector(40);
     List<DetectedFace> faces = fd.detectFaces( Trans-
forms.calculateIntensity(frame));
     for( DetectedFace face : faces ) {
         frame.drawShape(face.getBounds(), RGBColour.RED);
     }
```

Листинг 1. – пример использования каскадов HaarCascadeDetector

Реализация FKEFaceDetector может быть более полезна, так как она находит такие лицевые точки как уголки глаз, рта, носа. Поэтому объект который он возвращает это KEDetectedFace, который содержит массив лицевых точек(FacialKeypoint). На листинге 2 представлен пример выделения глаз используя реализацию FKEFaceDetector.

```
     FaceDetector<KEDetectedFace,FImage> fd = new FKEFaceDetector();
     List<KEDetectedFace> faces = fd.detectFaces( Transforms.calculateIntensity(
frame ) );
     for( KEDetectedFace face : KEDetectedFace ) {
         for( FacialKeypoint fk : face.getKeypoints() ) {
             if( fk.type == EYE_LEFT_CENTER )
                  frame.drawShape(new Circle(fk.position, 3), RGBColour.GREEN);
             if( fk.type == EYE_RIGHT_CENTER )
                  frame.drawShape(new Circle(fk.position, 3), RGBColour.GREEN);
         }
     }
```

Листинг 2. – пример использования каскадов FKEFaceDetector

Еще одной из интересных задач является нахождение предмета на изображении. Для этого используется алгоритм SIFT. Идея этого алгоритма заключается в следующем, необходимо найти «интересные точки» на изображении, описать их, затем найти такие точки на втором изображении и сравнить их между собой. Под «интересным точками» обычно подразумевают места на изображении,

где резко меняется цвет, яркость и т. п. Их особенность заключается в том, что они не зависят от того, где находятся, их размера или поворота.

Для поиска таких точек в OpenIMAJ используется реализация DoGSIFTEngine. Его использование показано в листинге 3.

```
      MBFImage query = ImageUtilities.readMBF(new
URL("http://static.openimaj.org/media/tutorial/query.jpg"));
      MBFImage target = ImageUtilities.readMBF(new
URL("http://static.openimaj.org/media/tutorial/target.jpg"));
      DoGSIFTEngine engine = new DoGSIFTEngine();
      LocalFeatureList<Keypoint> queryKeypoints = en-
gine.findFeatures(query.flatten());
      LocalFeatureList<Keypoint> targetKeypoints = en-
gine.findFeatures(target.flatten());
```

Листинг 3. – Поиск «интересных точек»

Когда у нас есть точки на исходном изображении и точки на целевом изображении их необходимо сравнить. Для этого есть несколько реализации интерфейса LocalFeatureMatcher.

BasicMatcher находит все похожие точки на изображении, и совпадений получается очень много. В таком случае обычно используют класс ConsistentLocalFeatureMatcher2d. Основной его особенностью является то, что он сравнивает не просто точки, а еще и сами плоскости моделей, тем самым отсекая лишние точки. Пример его использования приведен в листинге 4, а его работа проиллюстрирована на рисунке 1.

```
      RobustAffineTransformEstimator modelFitter = new RobustAffineTransformEstima-
tor(5.0, 1500,
        new RANSAC.PercentageInliersStoppingCondition(0.5));
      matcher = new ConsistentLocalFeatureMatcher2d<Keypoint>(
        new FastBasicKeypointMatcher<Keypoint>(8), modelFitter);
      matcher.setModelFeatures(queryKeypoints);
      matcher.findMatches(targetKeypoints);
      MBFImage consistentMatches = MatchingUtilities.drawMatches(query, target,
matcher.getMatches(),
        RGBColour.RED);
```

Листинг 4. – Пример использования ConsistentLocalFeatureMatcher2d

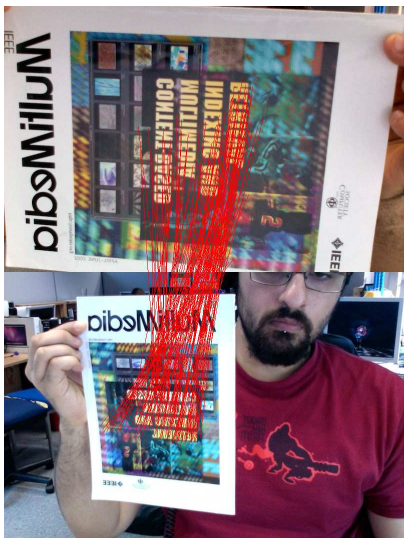

Рисунок 1. – Пример работы ConsistentLocalFeatureMatcher2d

Класс RobustAffineTransformEstimator имеет метод getModel(), который возвращает матрицу внутреннего аффинного преобразования. Данная матрица поможет преобразовать форму исходного

изображения в новую найденную. Пример этого приведен в листинг 5, а пример работы проиллюстрирован на рисунке 2.

```
target.drawShape(query.getBounds().transform(modelFitter.getModel()
  .getTransform().inverse()), 3,RGBColour.BLUE);
DisplayUtilities.display(target);
```

Листинг 5. – Пример выделения контуров

Рисунок 2. – Пример выделения контуров

В данной статье были рассмотрены такие возможности библиотеки OpenIMAJ, как нахождение лица на изображении и нахождение предметов при помощи алгоритма SIFT.

ЛИТЕРАТУРА

1. Material Design [Электронный ресурс] / Википедия. – Режим доступа: https://ru.wikipedia.org/wiki/Material_Design. – Дата доступа: 08.09.18.
2. Material Design Lite [Электронный ресурс] / Официальный сайт Material Design Lite. – Режим доступа: https://getmdl.io/. – Дата доступа: 08.09.18.