Diabeł tkwi w szczegółach, czyli rzecz o przetwarzaniu obrazów

„Wie Pan, jest jeszcze jeden mały detal, który nie daje mi spokoju" – te słowa z pewnością są dobrze znane miłośnikom serialu o przygodach porucznika Columbo. Amerykański policjant dzięki umiejętności kojarzenia faktów oraz na podstawie drobnych śladów potrafił odkryć prawdę i wskazać sprawcę zbrodni. W rzeczywistym świecie, który jak wiemy jest w większości biegunowo odmienny od tego, co jest prezentowane w serialach czy filmach, detale stanowią istotny element naszego życia. Nierzadko bowiem zróżnicowane procesy (jak chociażby identyfikacja człowieka na podstawie jego cech biometrycznych czy detekcja zróżnicowanych obiektów na obrazach) mogłyby zwrócić błędny wynik, gdyby autor implementacji czy algorytmu nie zwrócił uwagi na pewne szczegóły.

PRZETWARZANIE OBRAZÓW I JEGO ZASTOSOWANIA

Czym jest przetwarzanie obrazów? Najprościej (bądź najbardziej przystępnie) możemy je zdefiniować jako zbiór metod wykonujących pewne operacje na obrazie z wykorzystaniem narzędzi matematycznych takich jak mnożenie macierzy czy wyznaczanie wartości średnich. Metody te są często stosowane w zróżnicowanych procesach.

Jednym z bardziej popularnych zastosowań, w którym możemy je łatwo zaobserwować, są procedury identyfikacji czy weryfikacji biometrycznej. Na łamach miesięcznika „Programista" ukazywały się już artykuły, które odnosiły się do tej gałęzi nauki [1] [2]. W przypadku biometrii wstępne przetwarzanie obrazów ma bardzo duży wpływ na wynik końcowy. Bez dokładnych algorytmów wykonujących preprocessing wyniki klasyfikacji mogłyby nie oddawać rzeczywistej tożsamości użytkownika.

Inną gałęzią nauki, w ramach której możemy zaobserwować zastosowanie procedur przetwarzania obrazów, jest chociażby medycyna. Zauważmy bowiem, że nierzadko lekarze w pierwszej kolejności korzystają z zaimplementowanych rozwiązań, które wykorzystując analizę i przetwarzanie obrazów, dokonują detekcji zmian patologicznych. Pozwala to im na rozpoznanie różnych zaburzeń nawet w bardzo wczesnym stadium, co zwiększa szanse pacjenta na całkowite wyleczenie.

Oczywiście moglibyśmy w ten sposób przedstawiać jeszcze wiele różnorakich zastosowań przetwarzania obrazów, jednakże głównym celem tego artykułu jest zaprezentowanie czytelnikowi podstawowych algorytmów wykonujących pewne operacje na obrazie.

W dalszej części artykułu zaprezentowane zostaną:

- metody filtracji (rozmycie obrazu, wyostrzenie obrazu, filtr Sobela czy filtr usuwający szum w postaci „sól i pieprz"),
- metody binaryzacji (z ręcznym doborem progu, procentowa selekcja koloru czarnego)
- metodyki morfologii matematycznej (dylatacja, erozja, otwarcie i zamknięcie).

Wszystkie algorytmy zostały zaimplementowane z wykorzystaniem języka Java.

WCZYTYWANIE I ZAPISYWANIE OBRAZÓW W JAVIE

Pierwszym krokiem, który musimy zawsze wykonać, jest wczytanie obrazu w formie pierwotnej. Aby tego dokonać, przygotowaliśmy przykładowy obraz o nazwie `fingerprint1`, który wczytamy do zmiennej `fingerprintImage`. Należy również pamiętać o tym, że w przypadku niepowodzenia (na przykład – błędna ścieżka do pliku czy brak odpowiednich uprawnień do jego odczytania) metoda `ImageIO.read` wyrzuca wyjątek klasy `IOException`. W związku z tym dana procedura musi zostać ujęta w ramach bloku try-catch. Kod, z użyciem którego wczytujemy wybrany obraz, zaprezentowano w Listingu 1.

Listing 1. Wczytanie obrazu z wybranego pliku

```
String filePath = "./fingerprint1.jpg";
try {
  BufferedImage fingerprintImage = ImageIO.read(new
  File(filePath));
} catch(IOException ioException) {
  logger.log(Level.SEVERE, ioException.getMessage());
}
```

Aby jednak uzyskać możliwość obserwacji jakości wykonanych przez nas operacji na obrazie, równie ważna jest możliwość zapisania bieżącej postaci obrazu, która jest przypisana do zmiennej `imageOutput`. W tym przypadku skorzystamy z metody `write` z `ImageIO`. Analogicznie jak w przypadku poprzednim, tak i tutaj, ze względu na możliwość wyrzucenia wyjątku, musimy zastosować blok try-catch. Kod, z użyciem którego będziemy mogli zapisać obraz, zaprezentowano w Listingu 2.

Listing 2. Zapisanie bieżącej wersji obrazu

```
String outputPath = "./processedImage.jpg";
try{
  ImageIO.write(imageOutput, "jpg", new File(outputPath));
} catch(IOException ioException) {
  logger.log(Level.SEVERE, ioException.getMessage());
}
```