Autorzy:
Grzegorz Malinowski
Dorota Stolpe

Rozpoznawanie Cyfr Pisanych Ręcznie

Projekt z laboratorium Rozpoznawania Obrazów

1 Cele

Zadaniem projektu było stworzenie systemu, który rozpoznawał będzie zeskanowane cyfry pisane ręcznie. Dodatkowo wiadomo, że cyfry te pisane są przez różne osoby. System tego typu może znaleźć zastosowanie w instytucjach, w których przetwarzana jest duża ilość formularzy/danych z pisanymi przez ludzi liczbami np. przy odczytywaniu kodów pocztowych. Istnieje analogiczny problem rozpoznawania cyfr, ale dane wejściowe są w postaci strumienia współrzędnych położenia pióra na tablecie. Wydaje się, że system dla danych wejściowych w postaci skanów cyfr może mieć szersze zastosowanie.

2 Dane wejściowe

Jako dane wejściowe zastosowano zbiór 600 bitmap z zeskanowanymi cyframi. Każdy bitmapa zawiera 100 odpowiednio przygotowanych cyfr (tzn. przeskalowanych i ustawionych wg środka ciężkości), które można w prosty sposób „pociąć". Uzyskujemy w ten sposób 100 bitmap o wymiarach 28x28 pikseli, każda zawierająca przygotowaną do rozpoznawania jedną cyfrę.

3 Obróbka wstępna

3.1 Progowanie

Zastosowane do rozpoznawania bitmapy są zapisane ze skalą szarości 0-255. Dla celów rozpoznawania łatwiejszy do analizy jest obraz tylko czarno/biały, bez odcieni szarości. W tym celu zastosowano progowanie. Nie ustalono jednak stałego progu dla wszystkich próbek, gdyż mają one różne nasycenie. Przy stałym progu jasne próbki tracą dużo informacji a ciemne są mocno zniekształcone. Dla każdej próbki wyliczana jest średnia wartość nasycenia. Wartość progu użytego do progowania jest wartością proporcjonalną do wartości nasycenia. Im ciemniejszy obraz tym wyższa wartość progu. Należy mieć na uwadze, że

informacja o odcieniu szarości w różnych miejscach obrazu może być także przydatna przy rozpoznawaniu.

3.2 Odszumianie

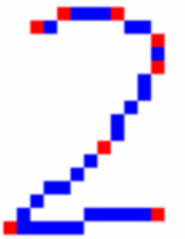
Przyjęto założenie, że powstałe po progowaniu pojedyncze lub podwójne punkty, które nie mają w swoim otoczeniu żadnych innych są szumem i usunięto je z obrazu. (Zastosowana metoda jest wrażliwa na szum.)

4 Zamodelowane cyfr i ocena podobieństwa próbki do modelu

Dla każdej cyfry stworzony został jej model (lub kilka modeli). Rozpoznawana bitmapa porównywana jest ze wszystkimi modelami, a decyzja o ostatecznej klasyfikacji zależy od wartości funkcji podobieństwa. Jest to funkcja typu koszt, dlatego wybierany jest model dla którego wartość funkcji jest minimalna.

4.1 Definicja modelu

Model jest to ciąg połączonych odcinkami punktów. Punkty te zostały nazwane punktami podstawowymi.



Rys1. Model cyfry 2 z zaznaczonymi
na czerwono punktami podstawowymi

Plik zawierający opis modelu zawiera listę współrzędnych punktów podstawowych. Jak wiadomo niektórych cyfr nie da się napisać „jednym pociągnięciem pióra" dlatego wprowadzono znacznik *break*, który informuje, że dany punkt oraz jego następnik nie są połączone odcinkiem.

W drugiej części pliku modelu (tzn. po linii zawierającej wyłącznie ciąg ###) można zdefiniować obszary, które dla prawidłowo napisanej cyfry nie powinny być całkowicie zaczernione (np. wnętrze cyfry 0). Szczególnym przypadkiem jest zdefiniowanie odcinka na którym musi leżeć przynajmniej jeden niezapalony punkt (np. pomiędzy początkiem a końcem cyfry 3). Obszary te definiuje się wypisując w jednej linii numery punktów podstawowych liczone od 0.

Każdy człowiek ma inny charakter pisma. Powstaje problem jak zamodelować cyfry. Płaski model dwójki może się sporo różnić od napisanej przez kogoś cienkiej, wysokiej dwójki dlatego zdefiniowane modele powinny być w miarę przeciętne. Ponadto stosowane jest

ograniczone przystosowanie/adaptacja modelu do rozpoznawanej próbki. Podnosi to jakość rozpoznawania. Wiele też zależy od zastosowanej funkcji oceny podobieństwa.

4.2 Funkcja oceny

Do oceny podobieństwa rozpoznawanej bitmapy z modelem zastosowana została funkcja postaci:

(min) A*missRatio+B*distanceRatio+C*emptyFactor

gdzie missRatio, distanceRatio i emptyFactor to odpowiednio wyliczane współczynniki zależne od próbki i modelu. Wartości A, B i C zostały dobrane metodą prób i błędów.

Współczynniki:

- *missRatio* – dla każdego czarnego punktu rozpoznawanej bitmapy odnajdywany jest najbliżej położony punkt modelu (spośród wszystkich punktów modelu nie tylko podstawowych). Następnie liczona jest liczba punktów modelu, które nie zostały uznane za najbliżej położone do żadnego punktu. Liczba ta jest dzielona przez liczbę wszystkich punktów modelu.
- *distanceRatio* – dla każdego punktu próbki obliczana jest minimalna odległość od punktów modelu. Wartość ta jest podnoszona do ustalonej potęgi. distanceRatio to suma potęg dla wszystkich czarnych punktów obrazu podzielona przez liczbę wszystkich czarnych punktów rozpoznawanego obrazu.
- *emptyFactor* – odpowiada liczbie obszarów (zdefiniowanych w drugiej części pliku modelu), które nie powinny być całkowicie zaczernione ale dla danej bitmapy nie spełniają tego warunku

Potrzebę stosowania missRatio oraz distanceRatio łatwo wytłumaczyć na przykładzie dopasowywania bitmapy 8-ki do modelu 3-ki oraz odwrotnie. Jeśli model 3-ki dopasuje się idealnie do prawej połowy bitmapy 8-ki, to wartość missRatio wyniesie 0, ale ciągle lewa połowa bitmapy 8-ki nie została dopasowana i należy obciążyć funkcję podobieństwa kosztem. Robi to distanceRatio. Jeśli natomiast do zeskanowanej 3-ki dopasowujemy model 8-ki, którego prawa połowa idealnie się dopasowała, to ciągle zostaje niedopasowana lewa połowa modelu. Obciążenie funkcji podobieństwa załatwia współczynnik missRatio.

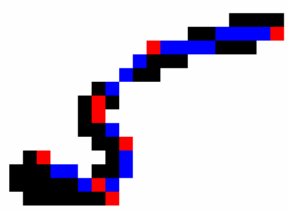
4.3 Adaptacja modelu

Bardzo rzadko się zdarza, że model idealnie pasuje do rozpoznawanej próbki. Nawet jeśli jest to próbka tej samej cyfry co model może być np. przesunięty, skręcony lub powyginany w nieokreślony

sposób. Przed obliczeniem podobieństwa model i próbka są nakładane na siebie, a następnie model jest "dopasowywany" do badanej próbki. Operacja ta polega na tym, że czarne punkty bitmapy oddziaływają na punkty podstawowe modelu w sposób analogiczny jak przyciąganie grawitacyjne. Przyciąganie to zależy wykładniczo od odległości punktów bitmapy. Obliczenia nie są wykonywane dla całego obrazu ale w ograniczonym oknie otaczającym punkt podstawowy modelu (odległe punkty praktycznie nie oddziaływają). Dodatkowo, aby zapobiec nadmiernej deformacji modelu, na punkty podstawowe działa siła analogiczna do tej, jak gdyby były przyczepione do swojej pozycji początkowej na gumce/sprężynce. Współczynniki tych sił są tak dobrane, aby punkt podstawowy mógł przemieścić się maksymalnie o pewną ustaloną liczbę pikseli. Nie może zajść sytuacja, że punkt podstawowy będzie ciągle w ruchu i nie ustabilizuje się.

Poniżej przedstawiono przykład działania adaptacji modelu do próbki na przykładzie cyfry 5. Rysunki przedstawiają rozpoznawaną bitmapę z nałożonym modelem przed i po adaptacji. Pod rysunkami znajdują się wartości współczynników.

missRatio: 0.55
distanceRatio: 0.09



missRatio: 0.04
distanceRatio: 0.06

Rys2.Ilustracja adaptacji modelu
oraz wartości współczynników

Model został także rozszerzony o definicję tzn. "ostrych" punktów. Są to miejsca, w których osoba pisząca cyfrę zaczyna pisać, kończy lub wykonuje nagły zwrot piórem np. w połowie pisania cyfry 3. W badanej próbce wyszukiwane są takie punkty. Punkty podstawowe modelu oznaczone jako "ostre" są przez nie bardziej przyciągane. Bez tej sztuczki punkty podstawowe modelu definiujące „ostre” fragmenty byłyby podczas adaptacji modelu zawsze niepotrzebnie wciągane do środka obrazu i wartość współczynnika *distanceRatio* pogorszyłaby się.

Rys3. Przykład „ostrych” punktów

5 Realizacja projektu

5.1 Środowisko wykonawcze

Projekt został zrealizowany w środowisku Java 1.5 z użyciem biblioteki Java Advanced Imaging 1.1.2. Biblioteka JAI znacznie ułatwia programiście wykonywanie operacji odczytu danych obrazowych z pliku oraz dostarcza szereg przekształceń na obrazie takich jak obroty, filtrowanie, przekształcenia afiniczne i wiele innych. Znacznie ułatwia operacje na danych obrazowych. Jej wadą jest mniejsza szybkość wykonywania operacji niż bibliotek napisanych dla języka C.

5.2 Struktura programu

Projekt składa się z kilku klas Javy umieszczonych w osobnych plikach. Są to:

- *Img* – odpowiedzialna za wyświetlanie oraz takie operacje na obrazie jak skalowanie, obliczenie jasności obrazu, progowanie (z uwzględnieniem stopnia zaciemnienia obrazu), obliczenie minimalnych i maksymalnych współrzędnych obrazu (użyte do ograniczenia obrazu i poprawienia efektywności), sprawdzenie czy w danym obszarze obrazu znajdują się miejsca niezaczernione
- *DataReader* – wykorzystywana do wczytania bitmap z plików wejściowych, wycinania poszczególnych próbek / cyfr i odczytywania z pliku *train.lab* informacji o poprawnej wartości rozpoznawanej próbki
- *DigitModel* – zawiera informacje opisujące modele poszczególnych cyfr. Umożliwia adaptację modelu do bitmapy oraz obliczenie wartości wszystkich współczynników funkcji podobieństwa.
- *Recognition* – odpowiedzialna za obliczenie wartości funkcji podobieństwa dla rozpoznawanej próbki i wszystkich zdefiniowanych modeli oraz tego z najmniejszą wartością funkcji podobieństwa
- *ModelDesigner* – narzędzie wykorzystywane do tworzenia modeli cyfr
- *Tester* – używana, jak sama nazwa wskazuje, do testowania rozpoznawania

5.3 Obsługa programu

Model Designer

Po uruchomieniu należy otworzyć plik z jedną z bitmap do rozpoznawania. Po kliknięciu na konkretną cyfrę zostanie ona wyświetlona się ona w powiększeniu. W dolnym oknie edycyjnym można zdefiniować model. Wszystkie wprowadzone zmiany wyświetlane są na bieżąco. Współrzędne punktów podstawowych modelu trzeba wpisywać ręcznie. Nie ma możliwości klikania i zaznaczania myszką na obrazie. Zdefiniowany model trzeba też ręcznie skopiować i zapisać do pliku przez zewnętrzny edytor. Na formatce umieszczona zostało także pole „Dopasuj model”. Gdy

jest zaznaczone wprowadzany model jest na bieżąco dopasowywany do próbki umieszczonej w oknie. W momencie tworzenia modelu opcja ta powinna być odznaczona, gdyż projektant modelu nie wie w jaki sposób przebiegała adaptacja modelu. Na pasku statusu wyświetlane są komunikaty o błędach oraz wartości poszczególnych współczynników. W prawym dolnym rogu umieszczone jest drzewo zawierające zdefiniowane modele. Po rozwinięciu i kliknięciu na wybrany plik modelu zostanie on wczytany do okienka edycyjnego.

Tester

Służy do uruchamiania rozpoznawania na większym zbiorze danych. Do uruchomienia wystarczy podać liczę próbek do rozpoznania. Dane te są pobierane z plików bitmap. Przyjęto że uwzględniane są pliki nazwane *mnist00000.bmp* i z kolejnymi rosnącymi co 1 numerami. Wystarczy nacisnąć *Start* i poczekać do zakończenia obliczeń.

W górnej części ekranu znajduje się macierz pomyłek. Wiersze odpowiadają za wartości oczekiwane, kolumny za rozpoznane. W centralnej części ekranu wypełniana jest lista błędnych rozpoznań. Tajemnicze liczby w każdym wierszu to odpowiednio: numer pliku danych (liczony od 0), numer cyfry w pliku (liczony od 0 poziomo), wartość oczekiwana i model który został uznany za najbardziej podobny do próbki. Początkowo na liście znajdują się wszystkie błędne rozpoznania. Po kliknięciu w konkretną komórkę macierzy pomyłek na liście umieszczane są tylko dopasowania odpowiadające wybranej komórce.

W celu ułatwienia analizy dopasowań umieszczono możliwość bezpośredniego przejścia do Model Designera. Wystarczy kliknąć dwukrotnie na liście błędnych dopasowań a otworzy się model designer z ustawioną rozpoznawaną bitmapą i dopasowanym modelem.

5.4 Ścieżki i pliki

Pliki modeli są umieszczone w podkatalogu *dm* głównego katalogu projektu. Pliki z bitmapami do rozpoznawania umieszczone są w głównym katalogu projektu i mają nazwy *mnist00000.bmp* gdzie na miejscu 00000 mogą znaleźć się liczby od 00000 do 00599. W katalogu głównym projektu powinna się także znaleźć plik *train.lab* z poprawnymi wartościami rozpoznania.

5.5 Pliki modeli

Przyjęto konwencję, że pierwszym znakiem nazwy pliku zawierającego model cyfry powinna być cyfra, której definiowany model dotyczy. Dzięki temu nie trzeba dodatkowo specyfikować jakiej cyfry dotyczy dany model. Program wczytuje wszystkie pliki z katalogu modeli. Aby dodać nowo skonstruowany model wystarczy odpowiednio go nazwać i umieścić w katalogu modeli.

6 Propozycje dalszego rozwoju projektu i uwagi

- Dopracować metodę znajdowania „ostrych” fragmentów. W chwili obecnej niektóre punkty błędnie są uznawane za „ostre”.
- Współczynniki funkcji oceny dobrane są metodą prób i błędów. Najlepszym rozwiązaniem byłoby dobranie wartości współczynników za pomocą jakiegoś systemu uczącego się.
- Dla każdej cyfry zastosowano po kilka modeli. Można zdefiniować ich więcej, ale szybkość rozpoznawania maleje wraz ze wzrostem liczby modeli.
- Można zastanowić się nad wyszukaniem ostrych krawędzi, dopasowaniem tych krawędzi do ostrych krawędzi zdefiniowanych w modelu, a następnie wykonaniem operacji analogicznej do symulowanego wyżarzania – „potrząsać” modelem początkowo w dość swobodny sposób i stopniowo ograniczać jego ruchliwość. Zapamiętywać rozwiązania lepsze od obecnych, a gorsze przyjmować tylko z pewnym prawdopodobieństwem.
- Przy założeniu, ze system miałby być wykorzystywany w praktyce trzeba by wykonywać rozpoznawanie w sposób bardziej efektywny. Szybsze może okazać się wykrywanie cech obrazu i przekazywanie tych informacji na wejście sieci neuronowej.
- Można rozważyć zastosowanie przekształceń afinicznych, ale zastosowanie kilku modeli dla cyfry oraz adaptacji modeli rozwiązuje ten problem.