Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

Zpracování Digitálního Obrazu

Návrh řešení semestrální práce ZDO

KKY/ZDO

Karel Müller, Bogdan Yeremenko
13.04.2024

1 Úvod

V současné době, kdy je zpracování obrazových dat stále důležitější v mnoha technologických a vědeckých oblastech, se význam efektivního a přesného zpracování těchto dat neustále zvyšuje. Tato semestrální práce se zaměřuje na aplikace počítačového vidění a strojového učení, konkrétně na zpracování a analýzu anotací obrazových dat. Cílem je navrhnout řešení, které umožní efektivní manipulaci s anotacemi a jejich vizualizaci, což je klíčové pro další analýzu a výzkum v mnoha aplikovaných oblastech, jako jsou medicínské diagnózy, autonomní vozidla, nebo bezpečnostní systémy.

Analýza dat

Náš dataset obsahuje 134 obrázků různých rozměrů. Tyto rozměry jsou ovlivněny skutečností, že obrázky byly získány studenty během praktických úloh. Studenti naší katedry tyto obrázky anotovali v rámci zpracování semestrální práce, což se rovněž promítá do konečné kvality datasetu. Kvůli omezené velikosti datasetu jsme se zaměřili především na detekci švů pomocí morfologického zpracování obrazu, místo úloh, které vyžadují trénovací techniky. Pro reprezentaci problému jsme využili algoritmus SVM, který je méně náchylný k přetrénování než neuronové sítě. Jako metriku pro hodnocení kvality jsme zvolili přesnost mezi počtem švů detekovaných ručně a pomocí kódu. Pro každou metodu budou uvedeny výsledky jak na celém datasetu, tak i na části, kterou jsme z původního datasetu vybrali jako správnou, protože obsahuje sporné momenty v segmentaci. Tyto sporné momenty jsou, jak jsme již uvedli, ovlivněny tím, že anotaci prováděli studenti.

Metody detekce počtu švů

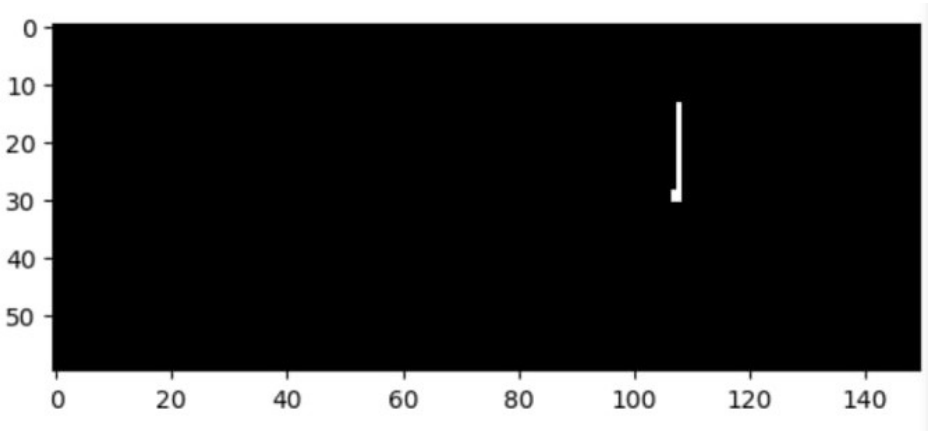
Pro detekci počtu švů na jednotlivých obrázcích jsme zvolili tři metody. Toto bylo umožněno především díky využití barevné nitě, která kontrastuje s prasečí kůží. První dvě metody využívají kombinace morfologických operací na obrázcích různé složitosti, zatímco třetí metoda využívá SVM pro srovnání s prvními dvěma metodami. Každý přístup obsahuje vlastní předzpracování obrázků, proto jsou detaily uvedeny v jednotlivých sekcích popisujících každou metodu.

První morfologická metoda

Předzpracování první morfologické metody je založeno na základním zvýraznění kontrastu v obrázcích. Pomocí CLAHE (Contrast Limited Adaptive Histogram Equalization) jsme adaptivně zvýšili kontrast jednotlivých obrázků a následně jsme je pomocí prahování převedli do binární podoby. První metoda využívá menší počet morfologických operací. Hlavní myšlenka spočívá v tom, že postupná eroze pomocí obdélníkového okénka o rozměrech (2,1) nechá po sobě zřetelné švy. Problém však byl, že švy byly viditelné po obou stranách řezu. Proto jsme ponechali pouze objekty v dolní polovině obrázků, protože v této části jsou švy výraznější. Následně jsme pomocí funkce `skimage.measure.label` spočítali počet objektů, který by měl odpovídat počtu švů na obrázku. Na celém datasetu dosáhla naše přesnost 25,37%, což nepovažujeme za dostatečné. Z tohoto důvodu jsme se rozhodli rozšířit počet předzpracování a morfologických operací, čímž jsme získali druhý způsob řešení.

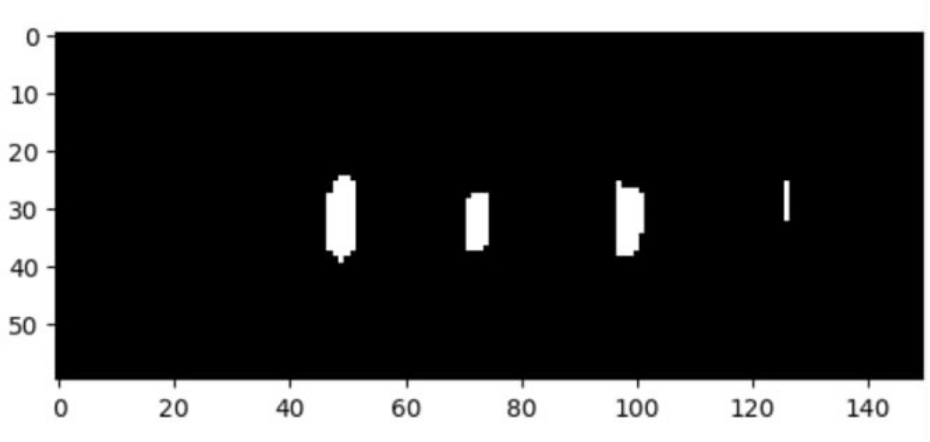
Druhá Morfologická operace

Kvůli nízké hodnotě přesnosti jsme se rozhodli provést složitější předzpracování a zjednodušit morfologické operace. Nejprve jsme upravili rozměr obrázků na 60x150 pixelů a zvolili hranový detektor, konkrétně Robertsův křížový operátor. Po testování jsme zjistili, že na mnoha obrázcích kvůli původní nízké kontrastnosti jsou hrany po detekci málo výrazné. Proto jsme zvýšili kontrast stejným algoritmem jako v první metodě, avšak tentokrát na vstupní obrázek hran. Následně jsme pomocí prahování převedli obrázky do binární podoby tak, aby byly díry bez švů oddělené. Pro lepší oddělení jsme aplikovali dilataci s obdélníkovým okénkem, aby se spojil řez v případě, že nebyl dostatečně zvýrazněn.



Obrázek 1: Výsledek detekce švů pomocí první morfologické operace

Dalším krokem bylo ponechání pouze největšího objektu na binárním obrázku pomocí `skimage.measure.regionprops`. To bylo nezbytné pro následnou dilataci, která měla za cíl vyplnit jednotlivé díry v obrázku a identifikovat delší vertikální hrany, aniž by spojila díry, které byly také zvýrazněny oproti pozadí řezu. Poté jsme pomocí obdélníkové eroze zúžili hrany, aby se odstranily náhodné dlouhé kolmé linie. Experimentálně se ukázalo, že tento přístup je lepší než použití širšího okénka pro vertikální erozi. Pro tento účel jsme zvolili obdélník o rozměrech 29x3 pixelů. Tímto způsobem jsme na celkovém datasetu dosáhli přesnosti 54.47 , a na námi vybraném a považovaném za správně anotovaném datasetu jsme dosáhli přesnosti 60.



Obrázek 2: Výsledek detekce švů pomocí druhé morfologické operace

Výhody a nevýhody morfologických operací

Hlavním problémem a nevýhodou použití těchto metod v našem případě byla nekonzistentnost dat. Kvůli různé kvalitě obrázků a odlišnému osvětlení bylo obtížné zvolit jednotné předzpracování pro všechny obrázky. Dalším problémem u hranových detektorů založených na gradientu byl stín, který způsoboval mnoho komplikací, zejména v případě bílého pozadí. Některé obrázky také obsahovaly nástroje použité během operace, což rovněž ovlivnilo výsledky. Ideálně bychom potřebovali obrázky se stejným osvětlením a rozlišením, aby bylo možné zajistit konzistentní kvalitu předzpracování.

Výhodou této metody bylo, že kvalitu zpracování jsme měli plně pod kontrolou. Kvůli nedostatečnému počtu dat jsme nemohli natrénovat klasifikátor pomocí neuronové sítě nebo jiné metody

strojového učení s učitelem, protože, jak jsme uvedli v algoritmu SVM, docházelo by k přetrénování. Proto jsme měli výhodu, že naše detekované objekty měly jednotnou podobu a byly ve většině případů velmi kontrastní, což umožnilo jejich zvýraznění a oddělení pomocí morfologických operací.

Gradientní metody nedosahující výsledků

Rovněž jsme se pokusili vypočítat počet vertikálních hran různými způsoby. Hlavní metodou byl horizontální průchod binárními obrázky po jednotlivých pixelech a počítání přechodů hodnoty z 0 na 1. Problém, na který jsme narazili, spočíval v tom, že obrázky mají různou polohu švů, a nebylo tedy možné provádět průchod v dolní části nebo v horní části obrázku konzistentně. To bylo způsobeno hlavně tím, že díry pro švy byly také zvýrazněny a nacházely se v různé vzdálenosti od řezu. Navíc řezy nebyly vždy horizontální, což ovlivňovalo volbu průchodu.

SVM

Při trénování SVM modelu jsme nejprve pracovali s původními obrázky, na kterých model dosáhl přesnosti 20%. Tento výsledek jsme zlepšili předzpracováním obrázků pomocí hranového detektoru, což zvýšilo kontrast a zřetelnost švů. Konkrétně jsme využili adaptivní histogramové vyrovnání (CLAHE) pro zvýšení kontrastu a následně jsme aplikovali prahování, čímž jsme obrázky převedli do binární podoby. Tímto způsobem jsme vytvořili předzpracované snímky, na kterých jsme poté model opětovně trénovali. Po aplikaci těchto technik se přesnost modelu zvýšila na 33%. Tento postup ukazuje, jak významné může být kvalitní předzpracování dat pro zvýšení výkonnosti modelu strojového učení, zvláště v úlohách, kde jsou obrázky hlavním zdrojem informací.

2 Praktické řešení

Pro praktické vypracování úlohy byly vytvořeny 2 skripty obsahující dohromady 3 metody zpracování. Skripty budou popsány v následujících podkapitolách.

2.1 Popis skriptu SVM

Tento dokument popisuje Python skript, který provádí klasifikaci počtu stehů na obrazech řezů pomocí modelu Support Vector Machine (SVM). Skript zahrnuje načítání dat, zpracování obrazů, trénink modelu a hodnocení jeho přesnosti.

2.1.1 Importování knihoven

```
import numpy as np
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pandas as pd
import cv2
import os
from skimage.transform import resize
from skimage import io, filters, exposure
import matplotlib.pyplot as plt
```

2.1.2 Načtení dat

Data jsou načtena z CSV souboru `output_segmentation.csv`, který obsahuje počet stehů (`n_stiches`) pro jednotlivé obrazy.

```
df2 = pd.read_csv('output_segmentation.csv')
print(df2.head())
y = df2["n_stiches"]
```

2.1.3 Příprava obrazových dat

Skript prochází všechny soubory ve složce `images/incision_couples`, načítá obrazy, provádí hraniční detekci pomocí Robertsova filtru, zlepšuje kontrast a mění velikost obrazů.

```
X = []
folder_path = "D:/ZDO/semestralni prace/images/incision_couples"
desired_width = 60
desired_height = 152

for filename in os.listdir(folder_path):
    img_path = os.path.join(folder_path, filename)
    if os.path.isfile(img_path):
        img = io.imread(img_path, as_gray=True)
        edge_roberts = filters.roberts(img)
        enhanced_image = exposure.equalize_adapthist(edge_roberts, clip_limit=0.02)
        resized_img = resize(enhanced_image, (desired_width, desired_height),
            anti_aliasing=True)
        X.append(resized_img.flatten())

X1 = np.array(X)
```

2.1.4 Rozdělení dat na trénovací a testovací sady

Data jsou rozdělena na trénovací a testovací sady v poměru 75% ku 25%.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

2.1.5 Trénink modelu SVM

SVM model je vytvořen s lineárním jádrem a trénován na trénovacích datech. Přesnost trénovací a testovací sady je zaznamenávána po dobu 100 epoch.

```
model = svm.SVC(kernel='linear')

train_accuracies = []
test_accuracies = []

for epoch in range(1, 101):
    model.fit(X_train, y_train)
    train_accuracy = model.score(X_train, y_train)
    test_accuracy = model.score(X_test, y_test)
    train_accuracies.append(train_accuracy)
    test_accuracies.append(test_accuracy)
    print(f"Epoch {epoch}, Train Accuracy: {train_accuracy}, Test Accuracy: {test_accuracy}")

plt.plot(range(1, 101), train_accuracies, label='Train Accuracy')
plt.plot(range(1, 101), test_accuracies, label='Test Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
```

```
plt.title('Training Progress')
plt.legend()
plt.show()
```

2.1.6 Predikce a hodnocení

Model provádí predikci na testovacích datech a výstup je vytištěn.

```
prediction = model.predict(X_test)
print(prediction)
```

2.1.7 Výpočet skóre

Výpočet skóre na základě shody predikcí s cílovými hodnotami.

```
goal = [3, 4, 1, 3, 1, 2, 5]
score = 0
for index in range(0, len(goal)):
    print(goal[index])
    if goal[index] == prediction[index]:
        score = score + 1
normilized_score = score / len(goal) * 100
print(normilized_score)
```

2.2 Skript Detekce_svu

Tento dokument popisuje Python skript, který se zaměřuje na detekci švů pomocí morfologických operací. Skript zahrnuje načítání dat, předzpracování obrazů, trénink modelu a hodnocení jeho přesnosti.

2.2.1 Importování knihoven

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
#  dal   potebn  knihovny...
```

2.2.2 Načtení a předzpracování dat

Popis načítání datasetu a předzpracování obrazových dat.

```
#  Naten  datasetu
data = pd.read_csv('path_to_dataset.csv')
print(data.head())

#  Pedzpracovn   obraz
images = []
for img_path in data['image_paths']:
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
    #  dal   zpracovn  obrazu
    images.append(img)

# Transformace dat do vhodnho formtu
```

```
X = np.array(images)
y = data['labels'].values
```

2.2.3 Rozdělení dat

Rozdělení datasetu na trénovací a testovací sady.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

2.2.4 Trénink modelu

Definice a trénink modelu.

```
from sklearn.svm import SVC

model = SVC(kernel='linear')

train_accuracies = []
test_accuracies = []

for epoch in range(1, 101):
    model.fit(X_train, y_train)
    train_accuracy = model.score(X_train, y_train)
    test_accuracy = model.score(X_test, y_test)
    train_accuracies.append(train_accuracy)
    test_accuracies.append(test_accuracy)
    print(f"Epoch {epoch}, Train Accuracy: {train_accuracy}, Test Accuracy: {test_accuracy}")

plt.plot(range(1, 101), train_accuracies, label='Train Accuracy')
plt.plot(range(1, 101), test_accuracies, label='Test Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training Progress')
plt.legend()
plt.show()
```

2.2.5 Vyhodnocení modelu

Predikce na testovacích datech a výpočet metrik pro hodnocení výkonnosti modelu.

```
predictions = model.predict(X_test)
print(predictions)

from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, predictions)
print(f'Accuracy: {accuracy}')
```

3 Závěr

Celkově plánované řešení zahrnuje integraci různých technologických komponent a metod, které by měly efektivně řešit stanovené úkoly. Tato integrace nejenže umožní efektivní práci s anotovanými obrazovými daty, ale také zvýší flexibilitu a škálovatelnost procesů pro budoucí rozšíření a aplikace.