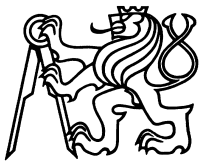
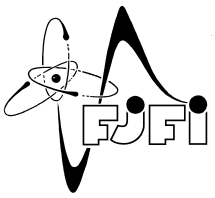


České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská



Dron ovládaný gesty

Gesture-controlled drone

Bakalářská práce

Autor: **Michal Průšek**

Vedoucí práce: **Ing. Adam Novozámský, Ph.D.**

Konzultant: **Ing. Václav Kůs, PhD.**

Akademický rok: 2022/2023

České vysoké učení technické v Praze, Fakulta jaderná a fyzikálně inženýrská

Katedra: matematiky Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	Michal Průšek
Studijní program:	Matematické inženýrství
Studijní specializace:	Matematická informatika
Název práce (česky):	Dron ovládaný gesty
Název práce (anglicky):	Gesture-controlled drone

Pokyny pro vypracování:

1) Proveďte rešerši stávajících metod strojového učení zaměřené na odhad pozice rukou a na jejím základě vyberte jednu metodu, kterou budete v práci dále používat.
2) Sestavte databázi gest pro ovládání dronu a napište modul, který bude tyto gesta rozpoznávat.
3) Nasnímejte dronem několik sekvencí takových gest a otestujte jejich správnou klasifikaci.
4) Seznamte se s programováním dronu. Především s přístupem k obrazovým datům, které dron snímá, tak s jeho řízením pomocí posílaných příkazů.
5) Napište program, který bude dron ovládat na základě gest snímaných kamerou v dronu.

Doporučená literatura:

1) R. C. Gonzalez, R. E. Woods, Digital Image Processing (4th ed.). Pearson, 2018.
2) I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.
3) C. Lugaresi et al., MediaPipe: A Framework for Building Perception Pipelines, arXiv e-prints, p. earXiv:1906.08172, Jun. 2019.

Jméno a pracoviště vedoucího bakalářské práce:

Ing. Adam Novozámský, Ph.D.
Ústav teorie informace a automatizace AV ČR, v.v.i., oddělení Zpracování obrazové informace, Pod Vodárenskou věží 4, 182 08, Praha 8

Jméno a pracoviště konzultanta:

Ing. Václav Kůs, PhD.
KM FJFI ČVUT Praha, Trojanova 13, 120 00 Praha 2

Datum zadání bakalářské práce: 31.10.2022

Datum odevzdání bakalářské práce: 2.8.2023

Doba platnosti zadání je dva roky od data zadání.

V Praze dne 31.10.2022

.....
garant oboru

.....
vedoucí katedry

.....
děkan

Poděkování:
Chtěl bych zde poděkovat především svému školiteli Ing. Adamovi Novozámskému, Ph.D. za pečlivost, ochotu, vstřícnost a odborné i lidské zázemí při vedení mé bakalářské práce. Dále děkuji svému konzultantovi Ing. Václavovi Kůsovi, PhD. za konstruktivní a věcné rady v průběhu vypracovávání této bakalářské práce.

Čestné prohlášení:
Prohlašuji, že jsem tuto práci vypracoval samostatně a uvedl jsem všechnu použitou literaturu.

V Praze dne 2. srpna 2023 Michal Průšek

Název práce:

Dron ovládaný gesty

Autor: Michal Průšek

Obor: Matematické inženýrství

Zaměření: Matematická informatika

Druh práce: Bakalářská práce

Vedoucí práce: Ing. Adam Novozámský, Ph.D., Ústav teorie informace a automatizace AV ČR, v.v.i., oddělení Zpracování obrazové informace, Pod Vodárenskou věží 4, 182 08, Praha 8

Konzultant: Ing. Václav Kůs, PhD., KM FJFI ČVUT Praha, Trojanova 13, 120 00 Praha 2

Abstrakt: Předmětem této práce je segmentace gest s využitím klasických metod zpracování obrazu. Jmenovitě pak prahování, morfologické transformace, zpětná projekce histogramu a další. Následně je popsán přístup ke klasifikaci gesta na základě jeho kontury a s využítím Fourierových deskriptorů. Nakonec je provedeno srovnání úspěšností klasifikace gesta námi navrženou metodou s úspěšností klasifikace na základě modelu neuronových sítí.

Klíčová slova: segmentace gesta, klasifikace gesta, zpětná projekce histogramu, kontura gesta, Fourierovy deskriptory, neuronové sítě

Title:

Gesture controlled drone

Author: Michal Průšek

Abstract: The subject of this work is gesture segmentation using classical image processing methods. Namely, thresholding, morphological transformations, histogram backprojection and others. Subsequently, an approach to gesture classification based on its contour and using Fourier descriptors is described. Finally, a comparison is made between the success rates of gesture classification using our proposed method and those based on a neural network model.

Key words: gesture segmentation, gesture classification, Histogram Backprojection, gesture contour, Fourier descriptors, neural network

Obsah

Úvod

Dron (z anglického drone) je bezpilotní létající zařízení, které může být řízeno na dálku, nebo je schopno samostatně létat pomocí autonomních řídících systémů nebo také pomocí předprogramovaných letových plánů. Dnes se bezpilotní drony hojně využívají k mnoha civilním úkolům například průzkumu terénu, hašení požárů nebo policejnímu sledování. Své použití našly také v armádě, kde slouží k průzkumným i útočným letům.

Cílem této bakalářské práce je navrhnout metodu, která na základě předloženého obrazu z kamery dronu detekuje a rozpozná gesto. Dalším cílem pak je na základě rozpoznaného gesta ovládat pohyb dronu. Pro účely této práce budeme uvažovat pouze statická gesta. Pro klasifikaci gest existují zpravidla dva typy přístupů.

Prvním z nich je přístup založen na klasifikaci barevného obrazu pomocí neuronových sítí, kdy na vytvořené databázi gest o velkém počtu vzorků natrénujeme model neuronové sítě. Naučený model neuronové sítě pak následně z libovolného předloženého obrazu gesta s určitou přesností rozpozná, o jaké gesto se jedná.

Druhou možností, jak klasifikovat gesta, je pak použití metod zpracování obrazu na barevný obraz z kamery. Výsledkem těchto metod je pak binární obraz gesta, kde pixely bílé barvy reprezentují část obrazu, kde byla detekována ruka provádějící gesto. Naopak, černé pixely reprezentují pozadí. Následně binární obrázek podlehá dalším metodám zpracování obrazu, které záleží na zvolené metodě klasifikace gesta.

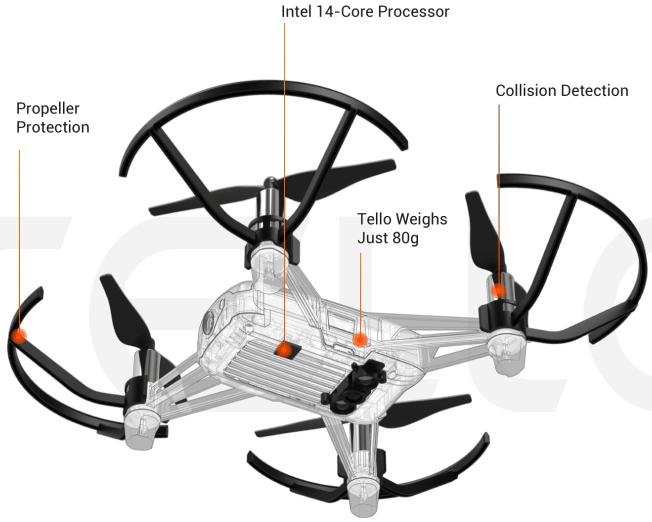
Základem pro úspěšnou klasifikaci gesta je jeho úspěšná možná segmentace. Segmentovat gesto (obecně lidskou kůži) lze velkým množstvím způsobů od segmentace kůže pomocí konvolučních neuronových sítí [14] přes prosté prahování na základě empirického pozorování odstínů kůže [18] až po segmentaci založenou na adaptivním histogramu kůže [26], [10]. Posledním zmíněným způsobem segmentace se zabývá právě tato práce.

Kapitola 1

Konstrukce a ovládání dronu

V dnešní době se setkáváme s velkým množstvím dronů různých typů a velikostí. Americké ministerstvo obrany dělí drony do pěti kategorií, kde je každé kategorii přiřazena určitá velikost, maximální vzletová hmotnost, provozní výška a hmotnost. Užší skupinou dronů jsou kvadrokoptéry. Obecně pod pojmem kvadrokoptéra rozumíme vrtulník se čtyřmy rotory. V současnosti se vývoj kvadrokoptér soustředí na menší bezpilotní letouny, které vynikají svými manévrovacími schopnostmi. Předmětem této práce je právě malá výuková kvadrokoptéra určená pro pohyb ve vnitřních prostorách.

Jedná se o model Tello [25] od čínské společnosti Ryze Robotics vytvořený ve spolupráci s firmou DJI, který disponuje až 13 minutovou dobou letu při vzletové hmotnosti 81 gramů. Dron je také vybaven kamerou schopnou snímat video v rozlišení 720p s frekvencí 30 snímků za sekundu. Přenos signálů ve formě streamování videa z kamery drona a přijímání letových instrukcí mezi dronem jinými elektronickými zařízeními je zprostředkováno pomocí WiFi. Výšku nad zemí monitoruje 3D kamerový systém ToF. ToF kamerový systém [9] (z anglického time-of-flight) slouží k měření vzdálenosti mezi kamerou a objektem založený na pozorování doby letu umělého světelného signálu mezi těmito dvěma objekty. Na následujícím obrázku je znázorněn právě model Tello.



Obrázek 1.1: model dronu DJI Tello (převzato ze stránky https://www.ryzerobotics.com/tello [25])

1.1 Analýza pohybu dronu

Každý ze čtyř rotorů dronu je přímo spojen s příslušným elektromotorem. Směry otáčení rotorů protilehlých dvojic jsou navzájem různé. Tím se kompletně eliminuje veškerý reakční otáčivý moment působící na dron. Pohyb ve vertikálním směru můžeme korigovat zvýšením nebo snížením rychlosti otáčení všech čtyř rotorů. Poklesem rychlosti otáčení jedné z dvojice protilehlých rotorů a zvýšením rychlosti otáčení druhé dvojice protilehlých rotorů o stejný díl způsobíme otáčení dronu podél jeho svislé osy. Při pohybu dronu ve vodorovném směru se opět sníží rychlost otáčení dvojice rotorů ve směru pohybu [12].

1.2 Ovládání dronu v jazyce Python

Model dronu Tello můžeme ovládat několika různými způsoby. Vedle mobilní aplikace od výrobce je volně k dispozici také knihovna pro jazyk Python s názvem DJITelloPy [5], která využívá oficiální Tello Software Development Kit (Tello SDK) přímo od Ryze Robotics. Knihovna obsahuje přibližně 80 příkazů, které slouží jak k ovládání drona, tak k přístupu dat z jeho senzorů nebo kamery. Základními příkazy jsou například:

- **connect()** – Slouží k připojení k Tello SDK. Je nutno použít před jakýmkoliv z příkazů sloužících k ovládání dronu.
- **streamon()** – Slouží k zahájení streamování videa.
- **takeoff()** – Slouží k automatickému vzletu dronu.
- **send_rc_control()** – Umožňuje pohyb dronu ve všech směrech.
- **flip()** – Dron udělá salto ve zvoleném směru.
- **land()** – Slouží k automatickému přistání.
- **emergency()** – Dron ihned vypne všechny své čtyři motory.

Dron disponuje bezpečnostními opatřeními, které zastaví motory dronu při jeho otočení do svislé polohy (kolmo k zemi) nebo výraznému zásahu lopatky jedné z vrtulí. Přesto je výhodné rozšířit je o další podmínky vypnutí motorů, kdy dron překročí určenou výšku nad zemí, nebo naopak, když je pod hranicí minimální přípustné výšky nad zemí. Při tendenci uživatele chytit dron zespodu dostává řídící jednotka dronu od ToF kamerového systému signál, že je dron nízko nad zemí a zvýší rychlost otáčení motorů – to může být při manipulaci s dronem nežádoucí.

Každý příkaz pak lze přiřadit zvolené klávese a tím kompletně ovládat dron pomocí klávesnice. Zároveň lze využít knihovnu MediaPipe [13], která může sloužit jako nástroj k detekci obličeje na snímcích z kamery dronu a tím mu umožnit pohybovat se za obličejem uživatele. Jelikož samotný model dronu nedisponuje zařízením, které by mu napomáhalo k orientaci v horizontální rovině, může pak obličej uživatele sloužit jako referenční bod, podle kterého si dron udržuje svou pozici v prostoru.

1.3 Ovládání dronu pomocí gest

Jak již bylo zmíněno, dron sám o sobě nemá senzory, které by mu umožňovaly orientaci v horizontální rovině. Proto je příhodné využít znalosti pozice obličeje v obraze, kterou nám dává MediaPipe a

pohybovat s dronem v závislosti na tom, kde se obličej v obraze nachází. Cílem je, aby byl střed ohraničujícího obdélníku obličeje (jeho souřadnice poskytuje MediaPipe) vždy co nejblíže středu samotného obrazu z kamery dronu. Tudíž minimalizovat velikost vektoru

$$\vec{v} = (v_x, v_y) = \vec{o} - \vec{s} \tag{1.1}$$

kde $\vec{o} = (o_x, o_y)$ je vektor souřadnic středu obrazu O, který nám posyktuje kamera dronu a $\vec{s} = (s_x, s_y)$ je vektor souřadnic středu obdélníku v obrazu O. x-ovou složku vektoru $\vec{v}$, v_y můžeme korigovat otáčením dronu v horizontální rovině podél jeho osy otáčení. Naopak, y-ovou složku téhož vektoru v_y můžeme ovlivnit pohybem dronu ve vertikálním směru. Zároveň pohyb dronu vpřed a vzad můžeme určit na základě vzdálenosti dronu od obličeje. Pokud je dron moc blízko obličeji, je relativní velikost ohraničujícího obdélníku obličeje vzhledem k obrazu z kamery dronu příliš velká. Naopak, pokud je dron příliš daleko od obličeje, je ohraničující obdélník příliš malý. Nastavíme-li referenční hodnotu obsahu ohraničujícího obdélníku A, můžeme přimět dron, aby se pohyboval vpřed nebo vzdad v závislosti na tom, zda je aktuální obsah ohraničujícího obdélníku obličeje větší nebo menší než A. Pokud je menší než A, potom se dron pohybuje vpřed. Naopak, pokud je větší než A, pohybuje se vzad.

V praxi se stává, že je dron přehlcen příkazy a než vykoná jeden pohyb, přijde mu několik dalších příkazů, které vykonává postupně. Toto je možné eliminovat tím, že nevyžadujeme, aby složky vektoru $\vec{v}$ v_x a v_y byly přímo rovny nule, ale stačí, když budou pod nějakou přípustnou mezí. To samé platí i o rozdílu referenčního obsahu A a obsahu ohraničujícího obdélníku, který se momentálně nachází v obrazu nasnímaném kamerou dronu.

Je zároveň užitečné ovládat dron nejen pouze popsaným způsobem, ale také vstupy z klávesnice pro případ, že ovládání výše popsaným způsobem selže. K tomu využíváme knihovnu Pygame v jazyce Python.

Zároveň v obraze nasnímaném kamerou dronu se nachází i ohraničující obdélník pro gesto vedle ohraničujícího obdélníku pro obličej. Ten je určen pro rozpoznání a klasifikaci gesta.

Kapitola 2

Metody zpracování obrazu

Pod pojmem obraz rozumíme zobrazení $f : M \times N \longrightarrow \mathbb{R}^3$, kde $M, N \subset \mathbb{R}$. Pokud jsou $M, N \subset \mathbb{N}_0$ konečné a $f(M \times N) \subset \mathbb{N}_0^3$, pak je obraz digitální. Obraz nazveme 8-bitový (jednokanálový, šedotónový), pokud $f(M \times N) \subset \{0, ..., 255\}$. Jelikož u digitálního obrazu jsou M a N konečné podmnožiny $\mathbb{N}_0$, je i jejich kartézský součin $M \times N$ konečná množina. Uspořádanou dvojici (x, y), kde $x \in M$ a $y \in N$ nazveme pixel, $f(x, y)$ pak intenzita pixelu. Množinu $M \times N$ si pak lze představit jako množinu pixelů obrazovky, kde zobrazení f přiřazuje každému pixelu jeho intenzitu (zobrazení $f : M \times N \longrightarrow \mathbb{N}_0^3$ přiřazuje pixelům barvu v určitém prostoru barev).

Oblast zpracování (digitálního) obrazu pak označuje zpracování digitálního obrazu pomocí digitálního zařízení. Pod takovou oblast spadá široké spektrum metod, od segmentačních, přes metody rekonstrukce obrazu, až po klasifikační.

Velkou skupinou metod jsou již zmíněné metody segmentace obrazu. Cílem těchto metod je automatické rozdělení vlastního obrazu na oblasti se společnými vlastnostmi mající určitý smysluplný význam. Typickým konkrétním cílem pak může být identifikace a oddělení určeného objektu v obrazu od jeho pozadí. Výsledné segmentované obrazy mají své využití například v oblasti počítačového vidění nebo analýze získaných snímků z dálkového průzkumu Země. Součástí této skupiny jsou například metody jako prahování, regionální metody, detekce hran, obrysové trasování (sledování hranice) [8] a v neposlední řadě segmentace rozvodím (Watershed) [1].

2.1 Prahování

Pro svou jednoduchou a intuitivní implementaci a velmi rychlou výpočetní rychlost patří prahování mezi nejzákladnější metody zpracování obrazu. Obecně pod pojmem prahování rozumíme rozdělení pixelů obrazu do dvou nebo více skupin podle jejich intenzit. Hraniční hodnoty intenzit oddělující jednotlivé skupiny pixelů nazýváme prahy. V případech kdy je práh konstanta nezávislá na pozici zkoumaných pixelů obrazu jej nazýváme globální, v opačném případě jej nazýváme adaptivní. Naopak, pokud práh závisí na libovolném okolí zkoumaného pixelu (například je funkcí intenzity sousedních pixelů), pak říkáme, že je lokální. V segmentačních úlohách je často postačující jeden globální práh pro daný obraz. Úlohy vyžadující určení více než dvou globálních prahů jsou často velmi těžce řešitelné a efektivněji lze dosáhnout výsledků využitím proměnných hodnot prahů pro různé části obrazu. Pokud výsledkem prahování označíme obraz g a P hodnotu prahu, pak předpis metody prahování lze definovat následovně

$$g(x, y) = \begin{cases} 1 & \text{když } f(x, y) > P \\ 0 & \text{jinak} \end{cases} \tag{2.1}$$

Tudíž prahování rozdělí pixely na základě jejich intenzity dle podmínky 2.1 na dvě třídy pixelů $c_1 := \{(x, y) \mid g(x, y) = 1\}$ a $c_2 := \{(x, y) \mid g(x, y) = 0\}$. Ačkoliv bychom přepis pro prahování mohli zobecnit i pro barevné obrazy (zavedením P jakožto uspořádané trojice a vyhodnocování podmínky v předpisu 2.1 po složkách), pro účely kapitoly 2.1 se omezíme pouze na jednokanálové 8-bitové obrazy.

V případě, že je objekt v obrazu, který chceme vysegmentovat, jistým způsobem homogenní a pozadí dostatečně odlišné, postačí nám jeden globální práh. Pro určení tohoto prahu můžeme využít následující algoritmus:

1. Zvolíme počáteční odhad prahu $P^{(0)}$.
2. Prahujeme obraz f pomocí prahu $P^{(n)}$ dle předpisu 2.1, kde horní index n označuje stupeň iterace. Tím dostaneme dvě třídy (skupiny) pixelů $c_1^{(n)}$ a $c_2^{(n)}$.
3. Určíme průměrné intenzity pixelů v obou skupinách $c_1^{(n)}$ a $c_2^{(n)}$. Označíme je jako $p_1^{(n)}$ a $p_2^{(n)}$.
4. Spočítáme novou hodnotu prahu $P^{(n+1)}$ v $n + 1$. iteraci jakožto aritmetický průměr $p_1^{(n)}$ a $p_2^{(n)}$

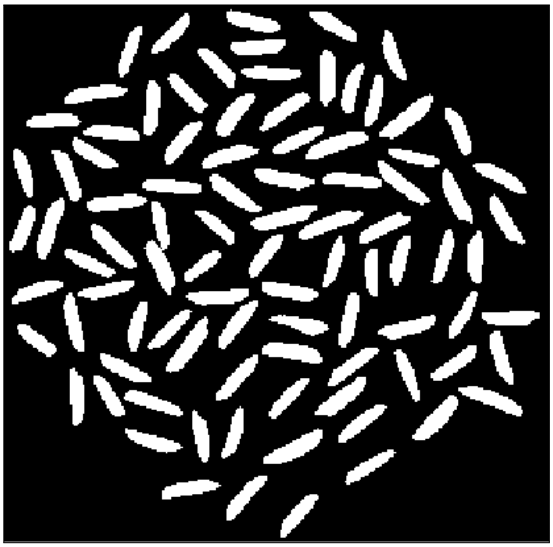
$$P^{(n+1)} = \frac{1}{2}(p_1^{(n)} + p_2^{(n)})$$

5. Opakujeme kroky 2 až 4 dokud není vzdálenost mezi hodnotami prahů $\Delta P^{(n+1)} := |P^{(n)} - P^{(n+1)}|$ ve dvou po sobě jdoucích iteracích n a $n + 1$ pod přípustnou konstantní mezí ϵ

Zmíněný algoritmus je robustní v již nastíněných případech, kdy jsou intenzity pixelů objektu a pozadí dostatečně odlišné (v histogramu obrazu lze pozorovat zřetelnou propast mezi skupinami pixelů). S tím je spojena skutečnost, že algoritmus není konvergentní pro libovolnou volbu počátečního prahu $P^{(0)}$. Hodnotu $P^{(0)}$ je nutno volit tak, aby byla větší než je minimální hodnota intenzity pixelů obrazu a zároveň menší než maximální. Konvergenci algoritmu lze pak zajistit volbou počáteční hodnoty prahu $P^{(0)}$ tak, aby odpovídala průměrné hodnotě intenzity pixelů obrazu.

Při použití zmíněného algoritmu na 8-bitový obraz (obrázek 2.1) dostaneme výsledný práh P o velikosti $P = 99$. Počáteční práh $P^{(0)}$ jsme volili tak, aby odpovídal průměrné hodnotě intenzity pixelů v 8-bitovém obrazu. Konstantu ϵ jsme položili rovno jedné. Prahování vstupního 8-bitového obrazu 2.1 prahem $P = 99$ znázorňuje obrázek 2.2.

Obrázek 2.1: Vstupní obraz



Obrázek 2.2: Vstupní obraz prahovaný algoritmem s výslednou hodnotou prahování $P = 99$

2.1.1 Otsuova metoda

Efektivnější metodou hledání optimálního globálního prahu je pak Otsuova metoda [15],[8]. Otsuova metoda je založená na hledání optimálního globálního prahu ve smyslu maximalizace mezitřídního rozptylu dvou nebo více tříd (skupin) pixelů. Na rozdíl o výše zmíněného algoritmu není Otsuova metoda iterativní a nevyžaduje tak žádnou volbu počáteční hodnoty prahu $P^{(0)}$.

Uvažujme množinu intenzit $\{0, 1, ... , I - 1\}$ a obraz s rozlišením $m \times n$ pixelů. Označíme-li n_i jako počet pixelů obrazu s intenzitou i, pak pro celkový počet pixelů obrazu platí

$$m \cdot n = n_0 + n_1 + ... + n_{I-1} \tag{2.2}$$

Normalizovaným histogramem obrazu nazýváme vektor $\vec{p} \in \mathbb{R}^I$ jehož i-tá složka je určena vztahem

$$p_i = \frac{n_i}{m \cdot n} \tag{2.3}$$

a platí

$$\sum_{i=0}^{I-1} p_i = 1 \tag{2.4}$$

kde $p_i \geq 0$. Zvolíme-li práh $P(k) = k$, kde $0 < k < I - 1$ a rozdělíme jím pixely do dvou tříd c_1 a c_2, kde c_1 obsahuje pixely s intenzitou $i \in [0, k]$ a c_2, naopak, pixely s intenzitou $i \in [k + 1, I - 1]$, potom lze určit pravděpodobnost $P_1(k)$, že libovolně vybraný pixel z obrazu bude patřit do třídy c_1, následovně

$$P_1(k) = \sum_{i=0}^{k} p_i \tag{2.5}$$

Analogicky lze zavést $P_2(k)$ jakožto pravděpodobnost, že vybraný pixel patří do třídy c_2. Z vlastnosti pravděpodobnosti komplementárních jevů pak platí vztah

$$P_2(k) = 1 - P_1(k) \tag{2.6}$$

Průměrná intenzita pixelů ve třídě c_1 se určí pomocí následujícího vztahu

$$m_1(k) = \sum_{i=0}^{k} i \cdot P(\, i \mid c_1) \tag{2.7}$$

To lze s využítím Bayesovy věty vyjádřit jako

$$m_1(k) = \sum_{i=0}^{k} i \cdot \frac{P(\, c_1 \mid i\,) \cdot P(i)}{P(c_1)} \tag{2.8}$$

Kde $P(c_1|i)$ je pravděpodobnost výskytu pixelu o intenzitě i ve třídě c_1. $P(c_1|i)$ je ovšem rovno 1, protože indexy i v sumě mohou nabývat hodnot 0 až k a tudíž všechny pixely s hodnotami intenzit $i \in [0, k]$ patří do c_1 z definice. $P(i)$ je pravděpodobnost výskytu i-té hodnoty, to odpovídá i-té složce histogramu $\vec{p}$. $P(c_1)$ je pravděpodobnost výskytu třídy c_1, respektive $P_1(k)$ ze vztahu 2.5. V Případech, kdy je $P_1(k)$ rovno 0, pokládáme $m_1(k)$ také rovno 0. Celkem tedy dostáváme

$$m_1(k) = \frac{1}{P_1(k)} \sum_{i=0}^{k} i \cdot p_i \tag{2.9}$$

Obdobně lze určit průměrná intezita pixelů ve třídě c_2 jako

$$m_2(k) = \frac{1}{P_2(k)} \sum_{i=k+1}^{I-1} i \cdot p_i \tag{2.10}$$

Opět pokud je $P_2(k)$ rovno 0, pak i $m_2(k)$ pokládáme rovno 0. Pro průměrnou intenzitu pixelů s intenzitou nepřesahující k platí

$$m(k) = \sum_{i=0}^{k} i \cdot p_i \tag{2.11}$$

Tudíž pro průměrnou intenzitu pixelů obrazu bude platit vztah

$$m_G = \sum_{i=0}^{I-1} i \cdot p_i \tag{2.12}$$

Pomocí vlastnosti sčítání sum v mezích lze ověřit platnost následujícího vztahu

$$P_1(k) \cdot m_1(k) + P_2(k) \cdot m_2(k) = m_G \tag{2.13}$$

Pro kvalitativní zhodnocení efektivity určeného prahu k zavedeme bezrozměrnou veličinu

$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_G^2} \tag{2.14}$$

kde σ_G^2 je globální rozptyl definovaný vztahem

$$\sigma_G^2 = \sum_{i=0}^{I-1} (i - m_G)^2 \cdot p_i \tag{2.15}$$

$\sigma_B^2(k)$ je mezitřídní výběrový rozptyl definovaný podle následujícího vztahu

$$\sigma_B^2(k) = P_1(k) \cdot (m_1(k) - m_G)^2 + P_2(k) \cdot (m_2(k) - m_G)^2 \tag{2.16}$$

To lze vhodnými úpravami převést do tvaru následujícího

$$\sigma_B^2(k) = P_1(k) \cdot P_2(k) \cdot (m_1(k) - m_2(k))^2 \tag{2.17}$$

Dalšími úpravami dostaneme finální tvar

$$\sigma_B^2(k) = \frac{(m_G \cdot P_1(k) - m(k))^2}{P_1(k) \cdot (1 - P_1(k))} \tag{2.18}$$

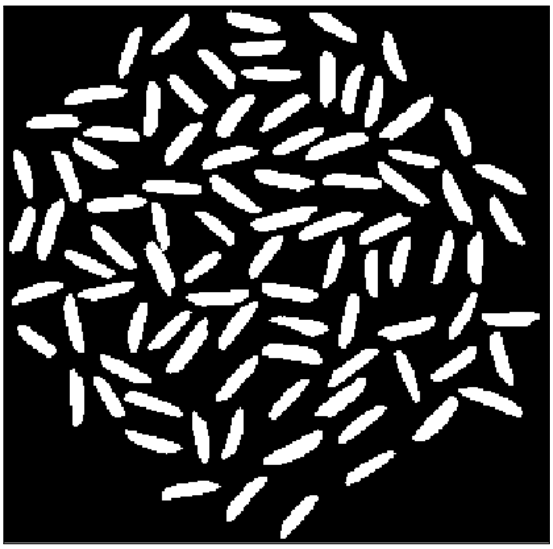
Poslední zmíněný tvar je z výpočetního hlediska velmi výhodný, protože k výpočtu $\sigma_B^2(k)$ postačí vypočítat m_G pouze jedenkrát a zbývá pak pouze výpočet parametrů $P_1(k)$ a $m(k)$ pro každé $k \in \{0, \dots, I-1\}$. Jelikož σ_G^2 je pro daný obraz konstantní, jsou $\eta(k)$ a $\sigma_B^2(k)$ na sobě přímo úměrně závislé dle vztahu 2.14 až na multiplikativní konstantu σ_G^2. Optimální hodnotu prahu k^* pak dostaneme maximalizací η, tudíž maximalizací σ_B^2 přes hodnoty k.

$$\sigma_B^2(k^*) = \max_{k \in \{0,\dots,I-1\}} \sigma_B^2(k) \tag{2.19}$$

Pokud je maxima nabýváno pro více než jednu hodnotu k, je příhodné volit k^* jakožto průměr hodnot k, ve kterých je maxima nabýváno. Vyčíslování $\sigma_B^2(k)$ z rovnice 2.18 a hledání maxima je výpočetně nenáročná operace pro 8-bitové digitální obrazy, kde $k \in \{0, \dots, 255\}$. Po získání optimální hodnoty prahu k^* lze obraz prahovat standardním způsobem pomocí k^*. Všimněme si, že veškeré parametry vstupující do rovnice 2.18 pro výpočet $\sigma_B^2(k)$, je možné získat z histogramu obrazu. Kromě optimální hodnoty prahu k^* je možné z histogramu obrazu získat i další informace. Například pravděpodobnosti $P_1(k^*)$ a $P_2(k^*)$ vyčíslené v hodnotě k^* ukazují poměrné rozdělení pixelů v obrazu do tříd c_1 a c_2. Podobně $m_1(k)$ a $m_2(k)$ jsou průměrné hodnoty intenzit pixelů ve třídách c_1 a c_2.

Použitím Otsuovy metody na vstupní 8-bitový obraz (obrázek 2.3) dostaneme výsledný práh $k^* = 98$. Následným prahováním vstupního 8-bitového obrazu pomocí prahu k^* dostaneme výsledek v podobě obrázku 2.4.

Obrázek 2.3: Vstupní obraz



Obrázek 2.4: Prahování vstupního obrazu Otsuovo metodou ($k^* = 98$)

2.1.2 Regionální modifikace Otsuovy metody

V případě segmentace obličeje od tmavého pozadí obrazu nastává často problém se stíny, které vrhá nos, prohlubně kolem očí nebo okolí scény obrazu. Při prahování klasickou Otsuovou metodou může nastat jev, kdy metoda vyhodnotí, že pixely zastíněné části obličeje jsou součástí třídy pixelů z tmavého pozadí obrazu. Ovšem Otsuovu metodu lze jednoduše modifikovat tak, abychom částečně eliminovali tyto nedostatky, které nám prahování globálním prahem přináší. Jedná se o použití Otsuovy metody na separované části obrazu (navrženo v [10]).

Množina pixelů obrazovky $O := M \times N$ lze rozložit na konečný počet disjunktních menších podmnožin $O' := M' \times N' \subset M \times N$, na jejichž obrazy $f(O')$ je pak možné jednotlivě použít Otsuovu metodu. Otsuova metoda pro každý z menších obrazů $f(O')$ určí jeho optimální hodnotu prahu k^*. Prahováním $f(O')$ podle jeho optimální hodnoty prahu k^* dostaneme obraz $f(O')^*$. Zpětným disjunktním sjednocením těchto menších prahovaných obrazů $f(O')^*$ pak dostaneme obraz $f(O)^*$, jehož rozlišení odpovídá rozlišení původního obrazu $f(O)$.

Při nedokonalém rozkladu původní množiny pixelů obrazovky O může nastat, že nějaké z dílčích množin rozkladu obsahují, co se intenzity týče, výhradně homogenní část objektu, který chceme segmentovat. Například při segmentaci obličeje od pozadí nějaká dílčí množina O' obsahuje pouze čelo, nebo tvář. V takovém případě Otsuova metoda rozdělí i pixely této homogenní části objektu na dvě třídy c_1 a c_2. Tímto pak dojde ke ztrátě informace v podobě zařazení části pixelů obličeje do třídy pozadí obrazu, které by v případě prahování celého obrazu $f(O)$ Otsuovo metodou mohly být zařazeny správně do třídy pixelů obličeje. Tento nežádoucí jev lze ovšem eliminovat následovně.

Je možné původní obraz $f(O)$ také prahovat s využitím Otsuovy metody a dostat tak obraz $\overline{f(O)}$. Dále pak obraz $\overline{f(O)}$ bitově sečíst s obrazem $f(O)^*$ získaný použitím regionální modifikace Otsuovy metody zmíněné výše. Tím částečně eliminujeme negativní vlivy, které prahování globálním prahem přináší a to bez ztráty informace.

Když podmožiny O' pixelů obrazovky O zvolíme tak, že obraz O „rozsekáme“ na menší obdélníky o počtu 5 odbdélníků na výšku a 5 na šířku, dostaneme celkem 25 podmonožin O', které následně prahujeme Otsuovo metodou. Výsledkem je pak obrázek 2.6. Obrázek 2.7 je pak bitový součet 8-bitového obrazu prahovaného Otsuovo metodou s 8-bitovým obrazem prahovaného regionálně Otsuovo metodou výše zmíněným způsobem.

Obrázek 2.5: Prahování vstupního obrazu pomocí prahu $P = 117$ získaného z Otsuovy metody

Obrázek 2.6: Prahování vstupního obrazu pomocí regionální modifikace Otsuovy metody

Obrázek 2.7: Bitový součet obrazu 2.5 s obrazem 2.6

2.1.3 Šum a jeho vliv na prahování

Hlavními zdroji šumu v digitálních snímcích jsou získávání a přenos obrazu. Při pořizování snímku je častokrát snímací senzor ovlivněn vnějším prostředím nebo kvalitou světlocitlivých prvků v senzoru. Naopak, při přenosu signálu mohou být snímky poškozeny vlivem interferenčních jevů v přenosovém médiu. Zejména při bezdrátovém přenosu mohou být signály zatíženy šumem vlivem nepříznivých podmínek v atmosféře, jako jsou například bouřky nebo husté sněžení.

Šum v obraze $f : M \times N \longrightarrow \mathbb{N}_0^3$ můžeme definovat jako obraz $h : M \times N \longrightarrow \mathbb{N}_0^3$, kde $M \times N$ je možina pixelů. Pro účely této kapitoly se ovšem omezíme pouze na šedotónové (8-bitové) obrazy. Některé z reálných šumů lze simulovat způsobem, že definujeme zobrazení h tak, aby přiřazovalo pixelům $(x, y) \in M \times N$ hodnoty intenzit, které svým rozložením četností odpovídají nějaké hustotě pravděpodobnosti.

Jednou z možností je simulace Gaussova šumu, který plyne z vlastností Gaussova (Normálního) rozdělení s parametry μ a σ^2, které značíme $N(\mu, \sigma^2)$. Hustota pravděpodobnosti f Gaussova rozdělení odpovídá vztahu

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{2.20}$$

kde $-\infty < x < +\infty$ reprezentuje intenzitu, μ je střední hodnota rozdělení, za kterou můžeme dosadit její maximálně věrohodný odhad v podobě průměrné hodnoty intenzity x (značíme $\overline{x}$) a $\sigma > 0$ je směrodatná odchylka, za kterou opět můžeme dosadit její maximálně věrohodný odhad z dat $\hat{\sigma}$. Obrázek zašuměný Gaussovým (bílým aditivním) šumem vypadá následovně.

Obrázek 2.8: Vstupní obraz

Obrázek 2.9: Vstupní obraz zašuměný Gaussovým šumem ($SNR = 0$ dB)

Dalším typem šumu je například Impulsní šum ("Salt and Pepper"). Impulsní šum přiřazuje každému pixelu (x, y) o intenzitě $f(x, y) = i$ v šedotónovém (8-bitovém) obraze intenzitu následovně

$$f(x,y) = \begin{cases} 255, & \text{s pravděpodobností } \frac{p}{2} \\ 0, & \text{s pravděpodobností } \frac{p}{2} \\ i, & \text{s pravděpodobností } 1-p \end{cases} \tag{2.21}$$

Obrázek zatížený Impulsním šumem vypadá následovně

Obrázek 2.10: Vstupní obraz

Obrázek 2.11: Vstupní obraz zašuměný Impulsním šumem ($p = 0,2$)

Dalšími typy šumu jsou například Rayleighův, Exponenciální nebo Elangův (Gamma) šum [2]. Speciálním typem šumu je pak Periodický šum [27].

Obecně na odstranění šumu v obraze funguje konvoluce. Ta ovšem rozmaže i původní obraz. Existují ale její modifikace, které se snaží tento vedlejší efekt potlačit.

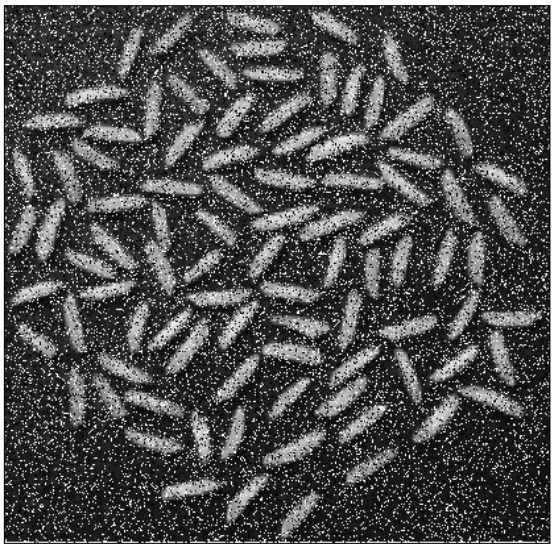
Pro každý typ šumu může být nejefektivnější jiná metoda. Pro Impulsní šum obecně funguje mediánový filtr [17]. Naopak, pro Gaussův šum se jeví jako nejefektivnější metoda Non-local means [3]. Výsledek jejího použití na obraz zatížený Gaussovým šumem vypadá následovně.

Obrázek 2.12: Obraz zatížený Gaussovým šumem

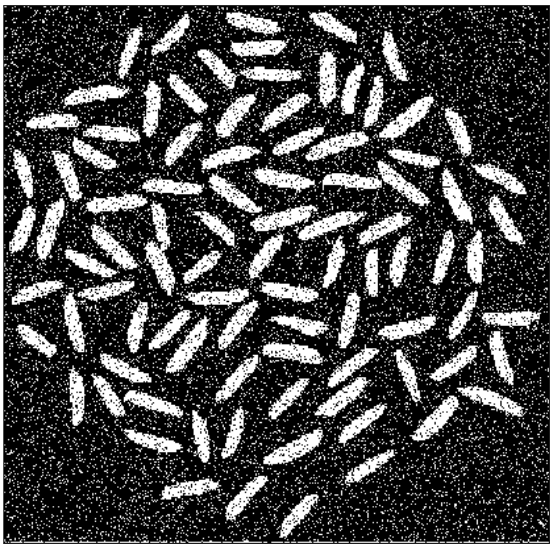
Obrázek 2.13: Metoda Non-local means použitá na obrázek 2.12

Vlivem šumu se může z jednoduché prahovací úlohy stát úloha prahováním neřešitelná. Prahování obrazů zatížených šumem bude produkovat neuspokojivý výsledek. Proto je nutné na obrazy zatížené šu-

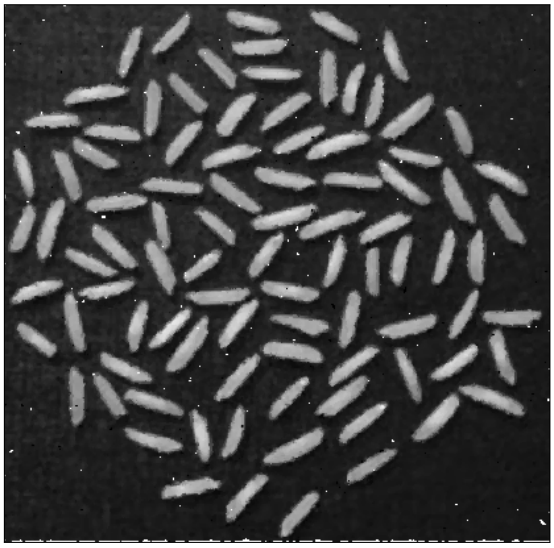
mem před prahováním použít adekvátní metodu zpracování obrazu, která obrazy šumu zbaví. Na následujících obrázcích lze vidět jaký vliv bude mít prahování na zašuměný a odšuměný obraz. Původní obraz rýže na černém pozadí ze sekce o prahování 2.1 zašumím Impuslním šumem s parametrem $p = 0,2$. Jak již bylo zmíněno, na odšumění obrazu zatíženého Impulsním šumem obecně funguje mediánový filtr. Ten použijeme pro odšumění obrazu.



Obrázek 2.14: Obraz zatížený Impulsním šumem ($p = 0,2$)



Obrázek 2.15: Prahování obrazu 2.14 prahem $P = 100$



Obrázek 2.16: Odšuměný obraz 2.14 mediánovým filtrem

Obrázek 2.17: Odšuměný obraz 2.16 prahovaný prahem $P = 100$

2.2 Morfologické transformace

Matematická morfologie je souhrný název pro techniky zpracování geometrických struktur založené na teorii množin. Primárně se morfologie zabývá binárními obrazy, ale je možné ji zobecnit i na 8-bitové. Binární obraz definujeme jako zobrazení $f : M \times N \longrightarrow \{0, 1\}$, tudíž pixely $(x, y) \in M \times N$ lze rozdělit právě na dvě třídy podle jejich hodnoty intenzity. Na třídu $c_1 := \{(x, y) \in M \times N \mid f(x, y) = 1\}$ a třídu $c_2 := \{(x, y) \in M \times N \mid f(x, y) = 0\}$, kde třída c_1 reprezentuje bílé pixely (pixely popředí (objektu) s intenzitou 1) a c_2 obsahuje pixely černé (pixely z pozadí s intenzitou 0). Pro množiny pixelů $O \subset M \times N$ a $S \subset \mathbb{N}_0^2$ pak definujeme Morfologickou (binární) transformaci T jakožto relaci mezi O a S, kde množina O reprezentuje obraz a množinu S nazýváme strukturní element. V praxi množina O odpovídá bílým pixelům z popředí obrazu (námi zavedené třídě c_1).

2.2.1 Dilatace

Jednou ze dvou základních (binárních) morfologických operací je Dilatace. Dilatací rozumíme relaci vektorového součtu množin pixelů O a S definovaný následovně

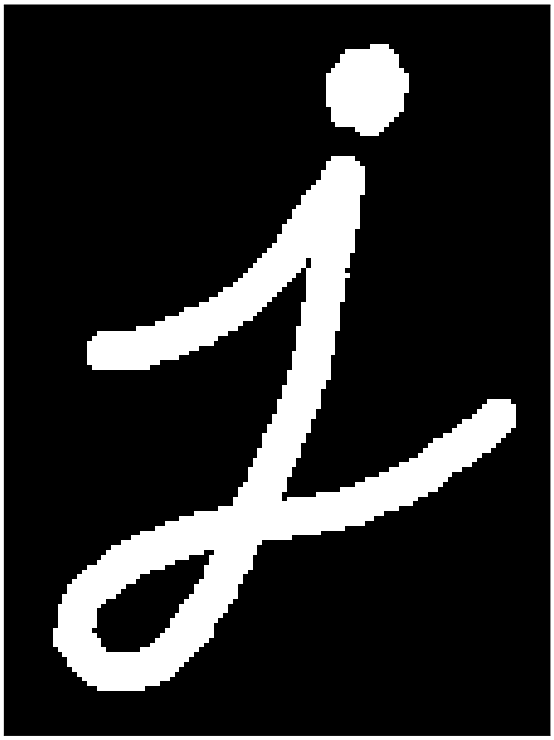
$$O \oplus S := \{(a, b) + (c, d) \mid (a, b) \in O, (c, d) \in S\} \tag{2.22}$$

kde operaci + pro pixely (a, b) a (c, d) definujeme jako sčítání vektorů po složkách

$$(a, b) + (c, d) = (a + c, b + d) \quad \forall a, b, c, d \in \mathbb{N}_0 \tag{2.23}$$

Efektem aplikace Dilatace na obraz je pak „ztloustnutí“ všech ploch obsahujících bílé pixely. Jinými slovy doplnění množiny bílých pixelů o určitý počet jejich sousedních, který odpovídá mohutnosti strukturního elementu.

Použijeme-li Dilataci na binární obraz 2.18, dostaneme obrázek 2.19.



Obrázek 2.18: Vstupní obraz

Obrázek 2.19: Morfologická Dilatace aplikovaná na obrázek 2.18

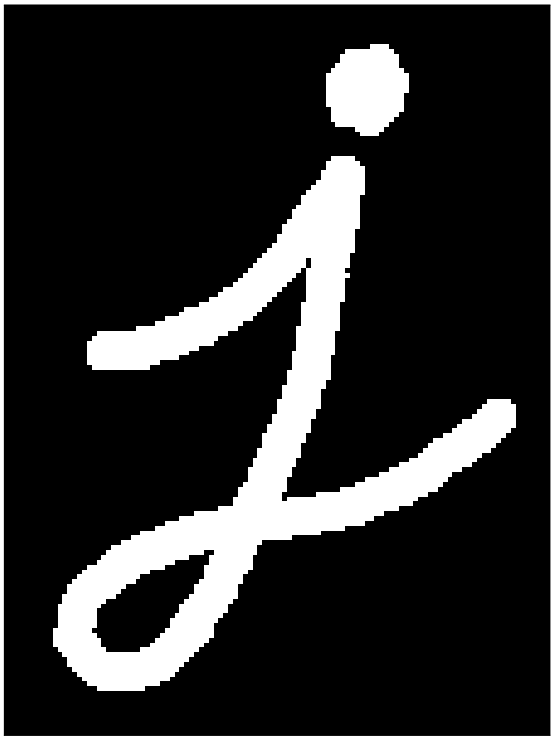
2.2.2 Eroze

Druhou ze základních (binárních) morfologických operací je Eroze. Naopak, Erozi definujeme jako relaci vektorového rozdílu množin pixelů O a S jakožto

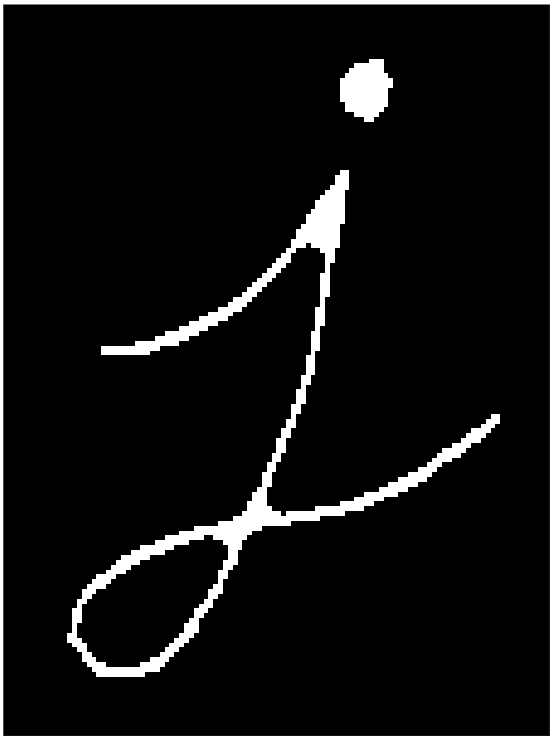
$$O \ominus S := \{(a,b) \in O \mid (a,b) + (c,d) \in O, (c,d) \in S\} \tag{2.24}$$

Oproti Dilataci aplikací Eroze na obraz zapříčiníme „ztenčení“ všech bílých ploch v obraze. Jinak řečeno vyjmutí některých z množiny bílých pixelů, jejichž počet opět odpovídá mohutnosti strukturního elementu.

Analogicky k Dilataci, aplikací Eroze na binární obrázek 2.18 dostaneme obrázek 2.21.



Obrázek 2.20: Vstupní obraz



Obrázek 2.21: Morfologická Eroze aplikovaná na obrázek 2.20

Lze ukázat, že operace Dilatace a Eroze nejsou k sobě navzájem inverzní a tudíž sekvenčním složením těchto dvou transformací lze získat další morfologické operace Otevření a Uzavření.

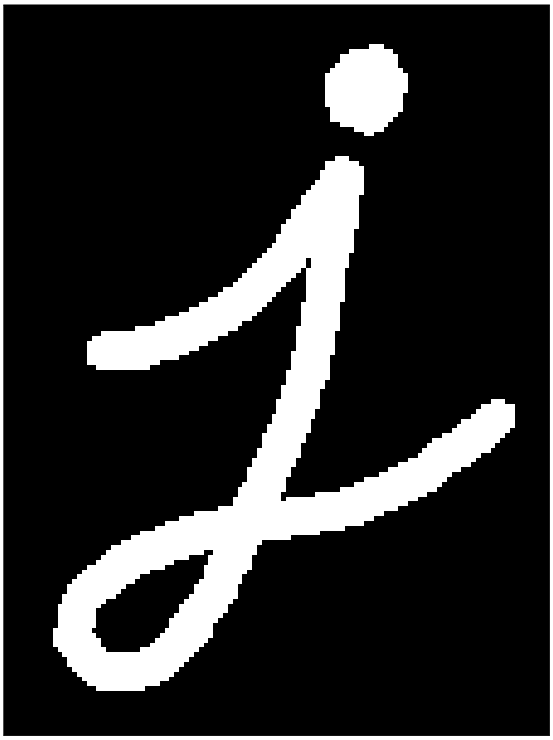
2.2.3 Otevření a Uzavření

Aplikaci Eroze na obraz odstraníme nejen nežádoucí struktury, ale také „ztenčíme“ i všechny ostatní. Použitím Dilatace na erodovaný obraz pak opět „nafoukneme“ zbylé bílé plochy a do jisté míry je tím restaurujeme do stavu, v jakém byly v původním obrazu. Toto sekvenční složení Eroze a Dilatace se nazývá Otevření definované jako

$$O \circ S := (O \ominus S) \oplus S \tag{2.25}$$

Aplikace Otevření na binární obraz 2.22 se projeví v podobě obrázku 2.23.

Obrázek 2.22: Vstupní obraz



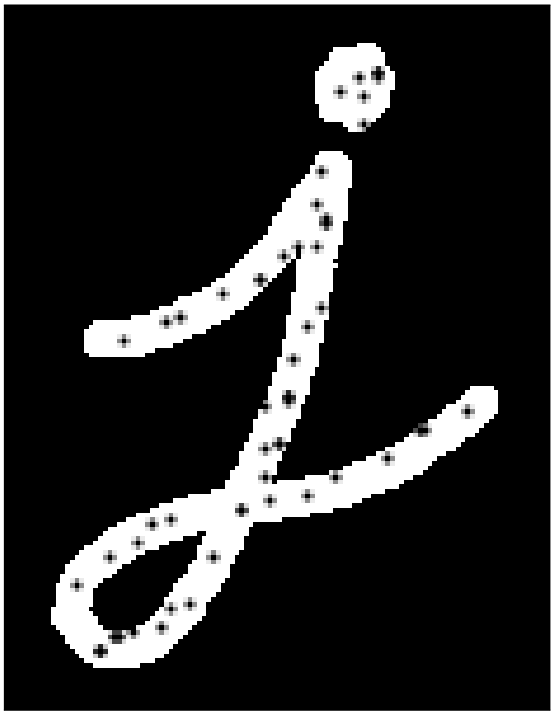
Obrázek 2.23: Otevření aplikované na obrázek 2.22

Duálním operátorem k (binárnímu) Otevření je pak operátor (binárního) Uzavření, který odpovídá sekvenčnímu složení Eroze a Dilatace v opačném pořadí

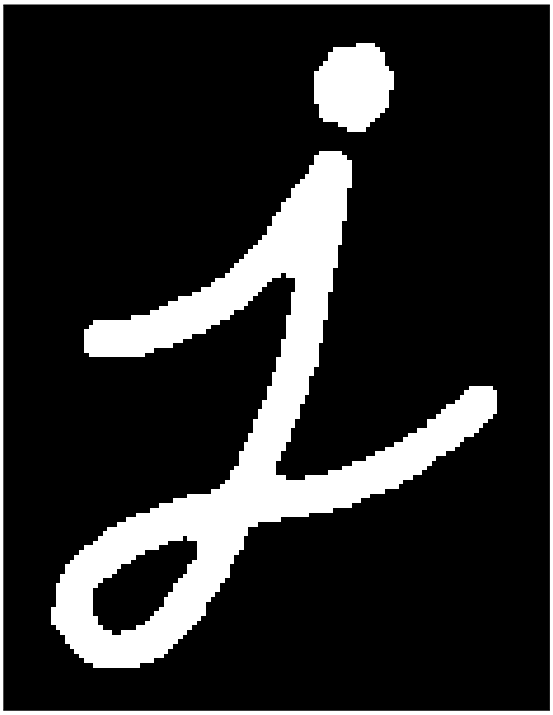
$$O \bullet S := (O \oplus S) \ominus S \tag{2.26}$$

Naopak, Uzavření v praxi spojí bílé plochy, které jsou blízko u sebe a zaplní v nich díry.

Použitím Uzavření na binární obraz 2.24 dostaneme obrázek 2.25.



Obrázek 2.24: Vstupní obraz



Obrázek 2.25: Uzavření aplikované na obrázek 2.24

Zajímavou vlastností operátorů Otevření a Uzavření je pak idempotentnost, která říká, že opakovaným sekvenčním skládáním těchto operací se výsledek předchozího nezmění.

$$O \circ S = (O \circ S) \circ S \tag{2.27}$$

$$O \bullet S = (O \bullet S) \bullet S \tag{2.28}$$

2.3 Zpětná projekce histogramu

Další metodou využívanou pro segmentaci obrazu nebo obecně mapování objektů v oblasti zájmu je zpětná projekce histogramu [23]. Tato metoda transformuje barevný obraz na 8-bitový, kde intenzity pixelů v 8-bitovém obrazu odpovídají pravděpodobnosti výskytu pixelu v daném referenčním vzorku objektu, který v obraze hledáme.

Provedení metody je následující. Uvažujme barevný obraz tj. zobrazení $f : M \times N \longrightarrow \mathbb{N}_0^3$ a barevný vzorek objektu definovaný také jako barevný obraz zobrazením $g : A \times B \longrightarrow \mathbb{N}_0^3$. Určíme normalizovaný histogram h vzorku g, kde i-tá složka normalizovaného histogramu h je definovaná jako

$$h_i = \frac{n_i}{m \cdot n} \tag{2.29}$$

kde n_i je počet pixelů s intenzitou i ve vzorku g, m je šířka vzorku a n je výška vzorku. Intenzita $i \in \{0, \ldots, I-1\}^3$ je obecně uspořádaná trojice reprezentující hodnoty barevných složek v prostoru barev. Z toho plyne, že i normalizovaný histogram h je obecně zobrazení $h : \{0, \ldots, I-1\}^3 \longrightarrow [0, 1]$. Následně pak každému pixelu $(x, y) \in M \times N$ v 8-bitovém obrazu f' přiřadíme hodnotu jeho relativního zastoupení ve vzorku následovně

$$f'(x, y) = h(f(x, y)) \cdot 255 \tag{2.30}$$

Tím získáme 8-bitový obraz f', který má stejné rozlišení jako f a intezita každého jeho pixelu odpovídá pravděpodobnosti výskytu téhož pixelu v obrazu f o intenzitě i ve vzorku g až na multiplikativní konstantu. Multiplikativní (škálovací) konstantou rozumíme číslo $255 = I - 1$, kde I je počet hodnot intenzity, kterých může být nabýváno v 8-bitovém obrazu.

Nyní demonstrujeme zpětnou projekci histogramu na příkladu z praktické části bakalářské práce. Mějme výřez gesta a obličeje v RGB barevném prostoru.

(a) Výřez obličeje

(b) Výřez gesta

Obrázek 2.26: Výřezy obličeje a gesta v RGB prostoru barev

Převedením obrazů v reprezentaci RGB prostoru barev do YCrCb prostoru barev (zdůvodněno v sekci 3.1.3) dostaneme

(a) Výřez obličeje

(b) Výřez gesta

Obrázek 2.27: Výřezy obličeje a gesta v YCrCb prostoru barev

Následně modifikovanou zpětnou projekcí histogramu (viz sekce 3.1.4) dostaneme obraz f' (viz obrázek 2.28). Následným prahováním prahem $P = 5$ dostáváme binární obraz znázorněný na obrázku 2.29. Nahrazením bílých pixelů původními barevnými v RGB reprezentaci z obrázku 2.26b pak dostáváme výsledek v podobě obrázku 2.30.

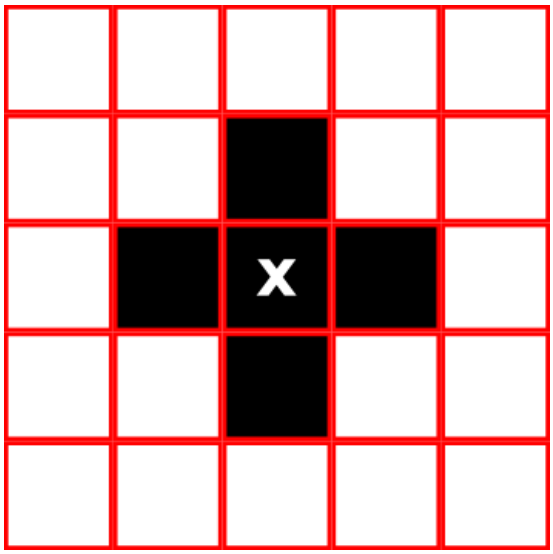
Obrázek 2.28: Výsledný obraz f' vzniklý zpětnou projekcí histogramu

Obrázek 2.29: Prahovaný obraz f' prahem $P = 5$

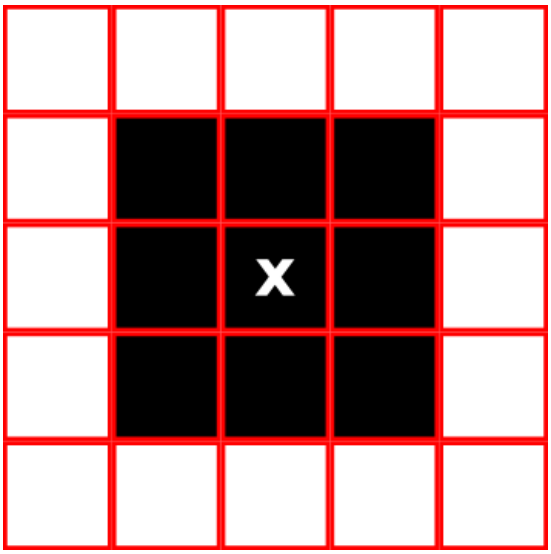
Obrázek 2.30: Nahrazení bílých pixelů v 2.29 barevnými z obrázku 2.26b

2.4 Algoritmy pro hledání kontur u binárních obrázků

Hledání kontur je jednou ze základních metod zpracování digitálních binárních obrázků. Obecně je výstupem algoritmů pro hledání kontur v binárních obrázcích posloupnost souřadnic, nebo řetězové kódy pixelů, které oddělují křivkově souvislé množiny pixelů s intenzitou 1 od křivkově souvislých množin pixelů s hodnotou 0. Pixely s intenzitou 1 budeme nazývat 1-pixely, analogicky pak pixely s intenzitou 0 budeme nazývat 0-pixely. Křivkově souvislé množiny 1-pixelů pak 1-komponenty, obdobně 0-komponenty křivkově souvislé množiny složené z 0-pixelů. Množinu 1-pixelů oddělující 1-komponentu od sousední 0-komponenty pak nazveme hranice (kontura). 4-sousedstvím pixelu X nazveme množinu pixelů, které s pixelem X sdílí stranu, 8-sousedstvím pixelu X pak nazveme množinu pixelů, které s pixelem X sdílí stranu nebo vrchol.



Obrázek 2.31: 4-sousedství pixelu X



Obrázek 2.32: 8-sousedství pixelu X

Hraniční pixel je pak každý 1-pixel, který ve svém 8-sousedství, respektive 4-sousedství obsahuje alespoň jeden 0-pixel. Nalezené kontury pak dále mohou sloužit k rozpoznávání objektů v obrazu, ana-

lýze obrazu nebo ke komprimaci dat. Nejjednoduššími algoritmy hledání kontur jsou algoritmy sledování hranice, které na základě volby počátečního hraničního pixelu najdou celou hranici, jejíž je tento hraniční pixel součástí. Naopak, složitější algoritmy, častokrát implementující algoritmy sledování hranice, jsou schopny hledat kontury v celém obraze bez počáteční znalosti hraničního pixelu a také řadí kontury do hierarchické struktury, která rozlišuje kontury mezi vnějšími a do nich vnořenými.

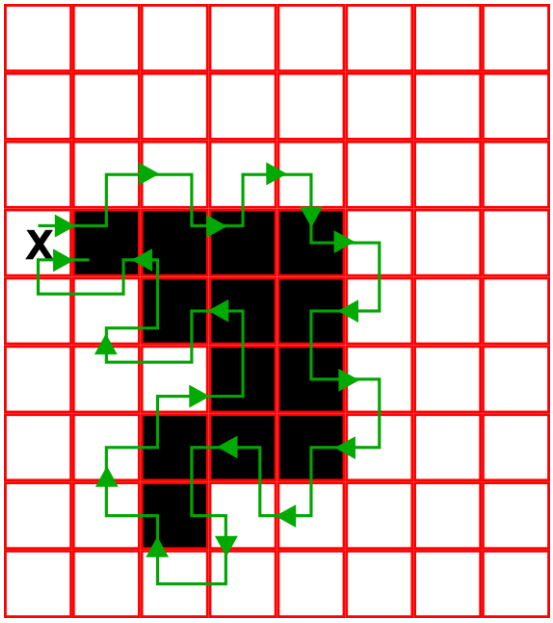
2.4.1 Algoritmus čtvercového trasování

Algoritmus čtvercového trasování (z anglického Square Tracing Algorithm) je jedním z nejjednodušších algoritmů sledování hranice. Jeho výstupem je posloupnost hraničních pixelů oddělujících 1-komponentu vnořenou do 0-komponenty, nebo naopak, 0-komponentu vnořenou do 1-komponenty, které ve svém 4-sousedství obsahují alespoň jeden 0-pixel.

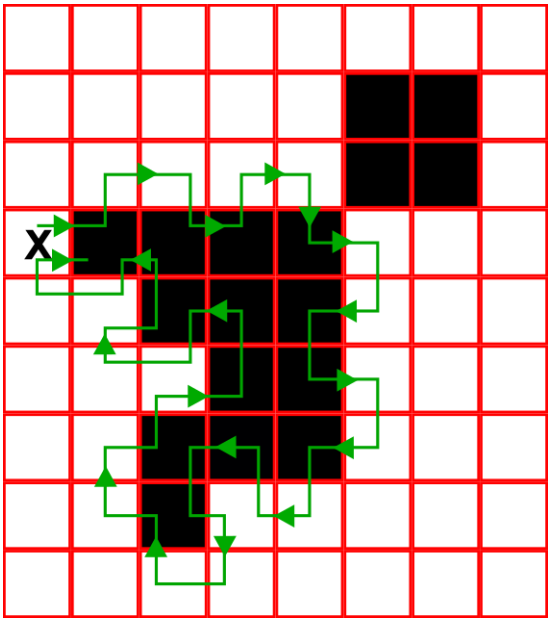
Představme si, že se pohybujeme po pixelech binárního obrázku. Začínáme na počátečním hraničním pixelu p_0 a bez újmy na obecnosti jsme natočeni čelem k horní hraně obrázku. Algoritmus je pak založen na jednoduchém pravidlu. Pokud se nacházíme na 1-pixelu, otočíme se čelem doleva a posuneme se o 1 pixel vpřed, v opačném případě se otočíme doprava a posuneme se o pixel vpřed. Opět vyhodnotíme intenzitu pixelu a pohneme se v souladu s pravidlem algoritmu. Veškeré 1-pixely, na které během chodu algoritmu vstoupíme pak ukládáme do posloupnosti pixelů hranice.

Algoritmus má různé podmínky ukončení. První z nich může být podmínka, který zastaví algoritmus v případě, že vstoupíme na pixel, na kterém jsme se už jednou (obecně n-krát) nacházeli. Dalším kritériem může být zastavení algoritmu v případě, že jsme vstoupili na počáteční hraniční pixel ze stejného směru podruhé. Tuto podmínku nazýváme Jacobovo kritérium zastavení. Můžeme si všimnout, že posun z pixelu p na pixel doleva, respektive doprava je relativní a je vztažen ke směru, ze kterého jsme na pixel p přišli. Proto většina algoritmů sledování hranice ve své implementaci s touto skutečností pracuje. Algoritmus čtvercového trasování je zázorněn na následujícím obrázku 2.33. Algoritmus začínáme tím, že z pixelu X o souřadnicích (x, y) vstupujeme na počáteční 1-pixel o souřadnicích $(x, y + 1)$. Zároveň algoritmus končí ve chvíli, kdy na počáteční pixel $(x, y + 1)$ vstoupíme podruhé ze stejného směru – Jacobovo kritérium zastavení.

Modifikujeme-li 1-komponentu z obrázku 2.33 přidáním čtyř pixelů způsobem znázorněným na obrázku 2.34, ukáže se, že algoritmus čtvercového trasování nefunguje správně a přidané čtyřy pixely, které jsou zajisté součástí kontury, nezaznamená. Proto si představíme robustnější algoritmus sledování hranice.



Obrázek 2.33: Chod algoritmu čtvercového trasování



Obrázek 2.34: Nedostatky algoritmu čtvercového trasování

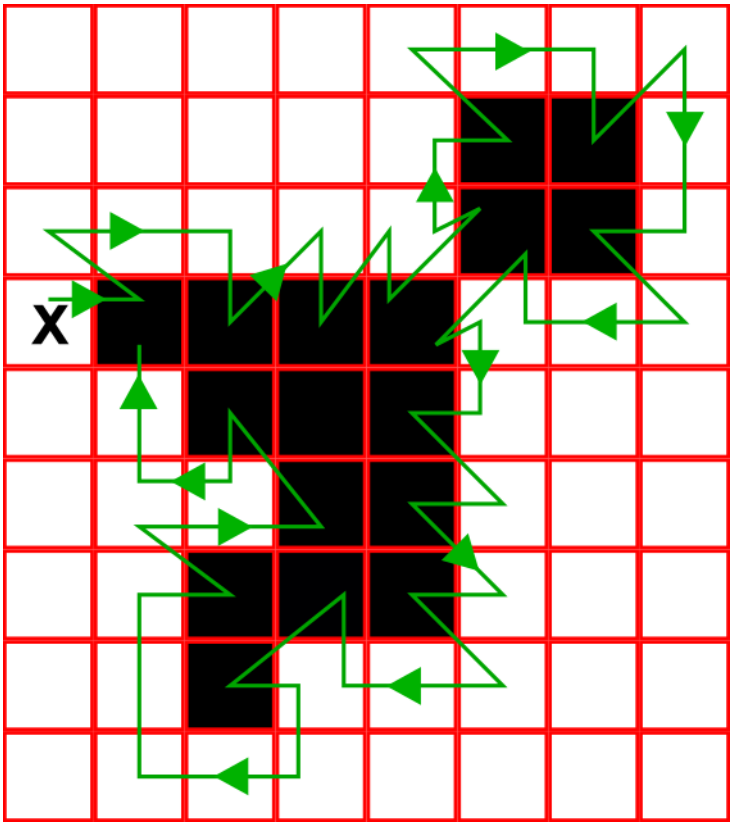
2.4.2 Mooreův algoritmus sledování souseda

Na rozdíl od algoritmu čtvercového trasování je výstupem Mooreova algoritmu sledování souseda posloupnost hraničních pixelů, které ve svém 8-sousedství obsahují alespoň jeden 0-pixel. Moorův algoritmus sledování souseda lze identicky zaměnit s algoritmem sledování hranice s anglickým názvem Radial Sweep, oba algoritmy totiž probíhají stejně a produkují stejné výsledky.

Algoritmus probíhá následovně. Opět začínáme na počátečním pixelu p_0 a vždy si při vtupu na nový pixel p_{n+1} pamatujeme směr, ze kterého jsme na nový pixel p_{n+1} vstoupili. Postupně pak testujeme pixely kolem pixelu p_{n+1} ve směru, respektive proti směru hodinových ručiček s tím, že začínáme testování vždy pixelem, který leží jako následující při pohybu ve směru, respektive proti směru hodinových ručiček od pixelu p_n, ze kterého jsme na pixel p_{n+1} vstoupili. Testujeme tímto způsobem pixely do té doby, než narazíme na 1-pixel. Na nově nalezený 1-pixel se pak posuneme, uložíme ho jako hraniční do posloupnosti pixelů hranice a opakujeme algoritmus do té doby, než není splněno ukončovací kritérium. Ukončovací kritéria jsou podobná jako v případě algoritmu čtvercového trasování.

- zastavení algoritmu v případě, kdy na stejný pixel vstoupíme podruhé (obecně n-krát)
- zastavení algoritmu v případě, kdy na počáteční hraniční pixel vstoupíme podruhé ze stejného směru (Jacobovo kritérium zastavení)

Uvažujme 1-komponentu z obrázku 2.34, kde selhal algoritmus čtvercového trasování. Opět z pixelu X o souřadnicích (x, y) vstupujeme na počáteční 1-pixel o souřadnicích $(x, y + 1)$. Zvolíme podmínku ukončení takovou, aby algoritmus skončil, když vstoupíme na počáteční pixel $(x, y + 1)$ podruhé. Chod Mooreova algoritmu sledování souseda je pak znázorněn na následujícím obrázku 2.35.



Obrázek 2.35: Mooreův algoritmus sledování souseda

2.4.3 Suzukiův algoritmus

Pokročilým algoritmem hledání kontur v celém obrazu a tvoření jejich hierarchické struktury je pak Suzukiův algoritmus [22], na kterém je založena implementace funkce **findContours()** z Python knihovny OpenCV [24].

V implementaci Suzukiho algoritmu je pak možné využít robustnější Mooreův algoritmus sledování souseda jakožto algoritmus sledování vnější, respektive vnitřní hranice. Pod vnitřní hranicí rozumíme hranici, která odděluje 1-komponentu od 0-komponenty ve formě díry, která se nachází uvnitř 1-komponenty (je 1-komponentou obklopená). Dále algoritmus využívá techniku rastrového snímání obrazu, využívanou většinou počítačových bitmapových systémů pro ukládání a přenos obrazu.

2.4.3.1 Algoritmus

Algoritmus pak probíhá následovně. Na počátku nastavíme hodnotu čítače kontur c na hodnotu 1. Metodou rastrového snímání procházíme obraz po liniích snímání (řádcích pixelů) od levého horního rohu do pravého dolního rohu, kde se algoritmus ukončí. Během rastrového snímání pak pro každý pixel (x, y) vyhodnocujeme následující dvě podmínky počátečních pixelů vnitřní, respektive vnější hranice

1. pokud pixel $(x, y-1)$ je 0-pixel a pixel (x, y) je 1-pixel, pak je pixel (x, y) počátečním pixelem vnější hranice a směrový pixel, ze kterého jsme na pixel (x, y) vstoupili nastavíme právě pixel $(x, y-1)$

2. pokud pixel (x, y) má intenzitu větší nebo rovno jedné a pixel $(x, y + 1)$ je 0-pixel, pak je pixel (x, y) počátečním pixelem vnitřní hranice a směrový pixel, ze kterého jsme na pixel (x, y) vstoupili nastavíme právě pixel $(x, y + 1)$

Jak již bylo řečeno v 2.4.1 algoritmy sledování hranice častokrát pracují se znalostí pixelu, ze kterého jsme na zkoumaný pixel přišli, proto je důležité informaci o tomto směrovém pixelu poskytovat souběžně s informací o počátečním. Pokud využíváme algoritmus Rastrového snímání s výše definovaným směrem snímání, je pak tento pixel, ze kterého jsme na zkoumaný pixel (x, y) vstoupili, vždy $(x, y - 1)$.

V případě, že pixel (x, y) právě snímaný rastrovým snímačem jednu z podmínek zmíněnou výše, pozastavíme rastrové snímání, zvýšíme hodnotu čítače kontur a použijeme jeden vybraný z dostupných algoritmů sledování hranice na počáteční pixel (x, y) a k němu příslušný směrový (vybraný na základě splněné podmínky). Algoritmus sledování hranice pak jednotlivé hraniční pixely označuje vybranými znaky (více v 2.4.3.2) a hraniční pixely ukládá do posloupnosti pixelů hranice, které pak ukládá do příslušné hierarchické struktury (více v 2.4.3.3). Po skončení algoritmu sledování hranice opět pokračuje rastrové snímání. Celý algoritmus pak končí, když rastrový snímač dosáhne pixelu v pravém dolním rohu.

2.4.3.2 Notace hraničních pixelů

Inovativním přístupem Suzukiho algoritmu je pak způsob notací jednotlivých kontur. Při chodu algoritmu sledování hranice jsou nejen jednotlivé 1-pixely (x, y) ukládány do posloupnosti hraničních pixelů, ale také jim je přiřazována celočíselná hodnota intenzity $f(x, y)$, kterou algoritmus sledování hranice vybírá ze dvou možností. Buď to c, nebo $-c$ dle následujících pravidel

$$f(x, y) = \begin{cases} -c, & \textbf{podmínka 1} \\ c, & \textbf{podmínka 2} \end{cases} \tag{2.31}$$

kde podmínky jsou definované následovně

podmínka 1: když aktuálně sledovaná hranice odděluje 0-komponentu obsahující pixel $(x, y + 1)$ a 1-komponentu obsahující pixel (x, y)

podmínka 2: v opačném případě a zároveň pokud se pixel (x, y) nenachází na již dříve sledované hranici

Díky této notaci pixelů společně s podmínkami počátečních pixelů sledování hranice se nemůže stát, že by byl pixel (x, y) příslušící vnější, respektive vnitřní hranici počátečním pixelem pro algoritmus sledování vnější, respektive vnitřní hranice.

2.4.3.3 Hierarchická struktura algoritmu

Jak již bylo řečeno, Suzukiův algoritmus je schopen nalezené hranice řadit hierarchicky například do struktury n-árních stromů. Kde kořenem stromu je vždy rám obrazu tvořen z prvního a posledního řádku pixelů a prvního a posledního sloupce pixelů obrazu. Rám je pak vnitřní rodičovskou hranicí (předkem) pro všechny další nalezené hranice.

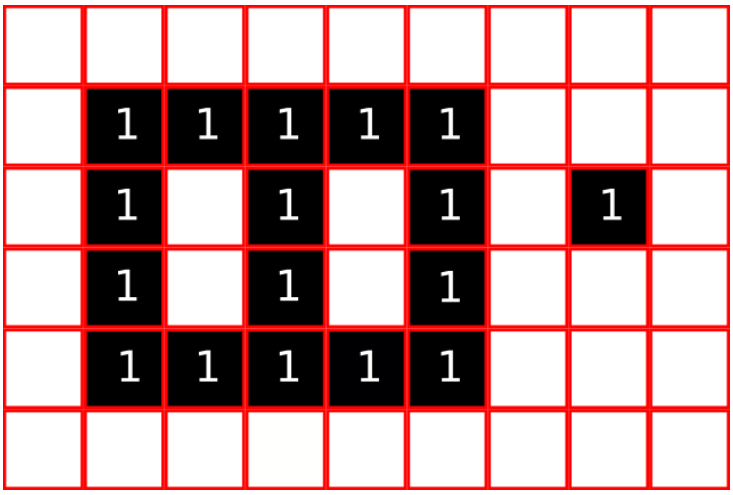
Při snímání pixelů rastrovým snímačem a sledování hranic si po nalezení počátečního pixelu hranice H ukládáme čítač kontur c, ale také hodnotu čítače kontur předchozí nalezené hranice H', který označíme d. Tato předchozí nalezená hranice H' je pak rodičovská hranice nově nalezené hranice H, nebo hranice, která má s nově nalezenou hranicí H společného rodiče. O této skutečnosti pak rozhoduje následující tabulka. Podle ní algoritmus rozhoduje, která z hranic je rodičovskou hranicí hranice H.

Typ H \ Typ H'	vnější hranice	vnitřní hranice
vnější hranice	rodičovská hranice hranice H'	hranice H'
vnitřní hranice	hranice H'	rodičovská hranice hranice H'

Tabulka 2.1: Pravidlo pro určování rodičovské hranice hranice H

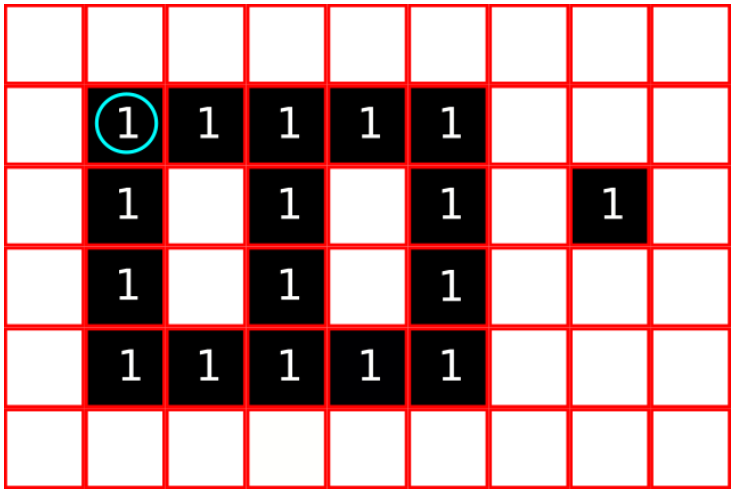
Číslo d (hodnota čítače předchozí nalezené hranice H' před nalezením H) příslušící hranici H nám pak slouží k orientaci a ke správnému umístění hranice H do námi zvolené hierarchické stuktury (nejčastěji však v podobě n-árního stromu).

Suzukiho algoritmus lze pak shrnout v podobě následujících schémat. Uvažujme 1-komponentu znázorněnou na následujícím obrázku 2.36.

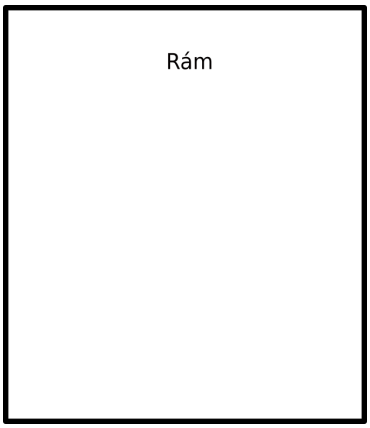


Obrázek 2.36: Počáteční 1-komponenta

Vždy na obrázku vlevo je znázorněn chod Suzukiho algoritmu a vpravo příslušná hierarchická struktura kontur. Rastrovým snímáním procházíme pixely obrazu po řádcích, dokud nenarazíme na první 1-pixel (označený tyrkysovou barvou).

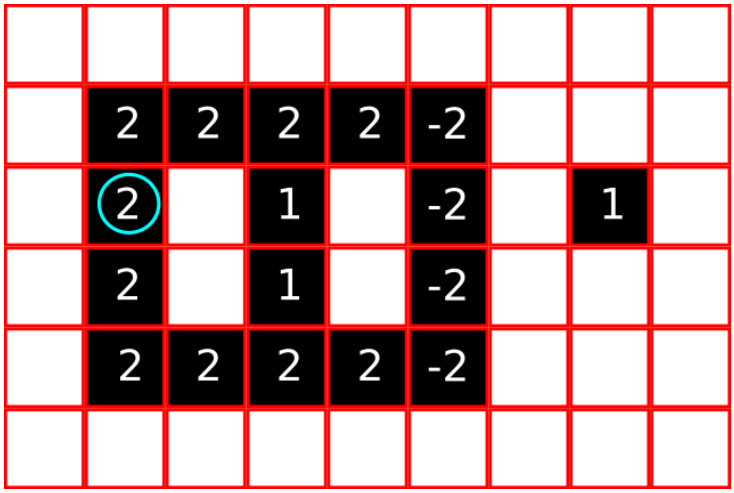


Obrázek 2.37: Chod Suzukiho algoritmu

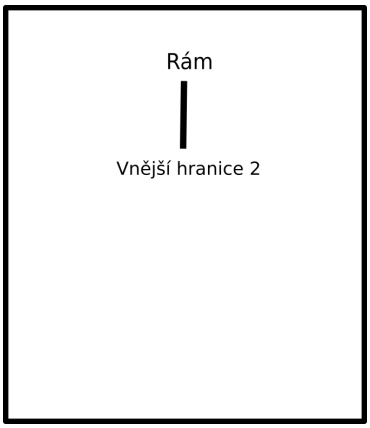


Obrázek 2.38: Hierarchická struktura kontur

Tento tyrkysově označený pixel splňuje podmínku pro počáteční pixel vnější hranice. Následně sledujme celou vnější hranici a anotujme jednotlivé hraniční pixely sekvenčním číslem 2, respektive -2 (dle 2.31) a následně umisťme nalezenou vnější konturu do hierarchické struktury kontur. Poslední nalezenou hranicí je v tomto případě rám obrazu se sekvenčním číslem 1, tudíž hranici umístíme do hierarchické struktury jakožto potomka rámu. Obnovme snímání rastrovým snímačem. Jelikož se na druhém řádků nenachází žádný další pixel, který by splňoval podmínku počátečního pixelu hranice, narazíme počáteční pixel až na řádku třetím.



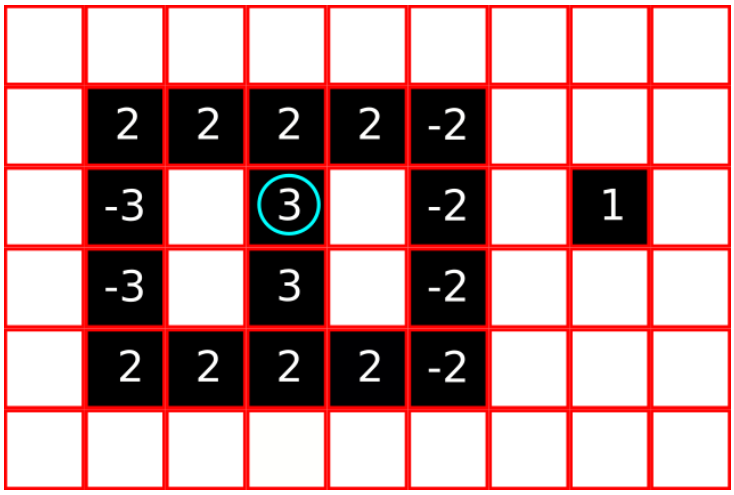
Obrázek 2.39: Chod Suzukiho algoritmu



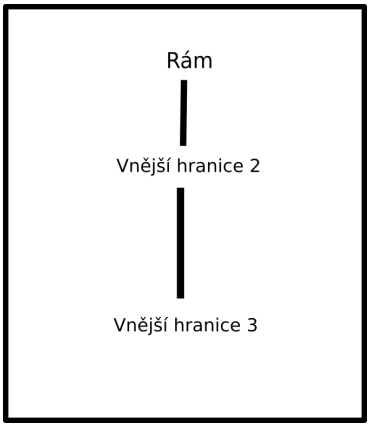
Obrázek 2.40: Hierarchická struktura kontur

Tento tyrkysově označený pixel splňuje podmínku počátečního pixelu vnitřní hranice. Sledujme tuto vnitřní hranici a anotujme příslušné hraniční pixely dle 2.31. Zároveň poslední nalezená hranice je vnější, tudíž dle tabulky výše rozhodneme, že rodičovskou hranicí nově nalezené vnitřní hranice je právě poslední nalezena vnější hranice se sekvenčním číslem 2. Obnovíme dále rastrové snímání, dokud nena-

razíme na další pixel splňující podmínku počátečního pixelu hranice. Ten opět označíme tyrkysovou barvou.

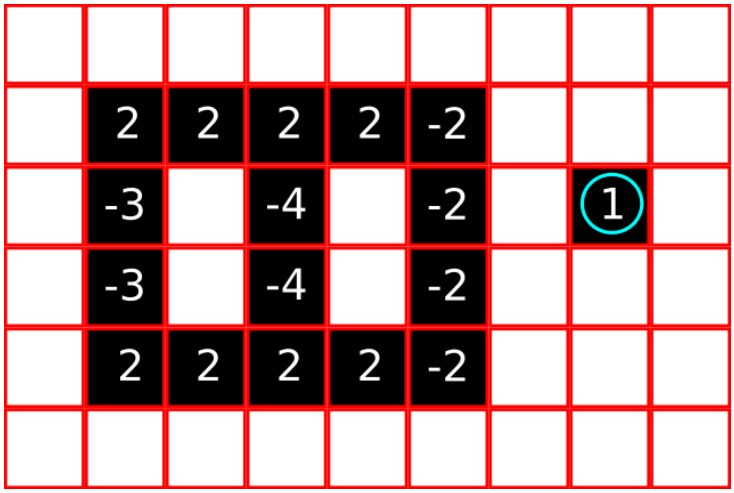


Obrázek 2.41: Chod Suzukiho algoritmu

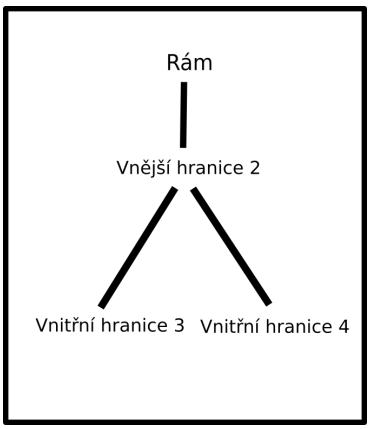


Obrázek 2.42: Hierarchická struktura kontur

Tento počáteční pixel splňuje podmínku počátečního pixelu vnitřní hranice. Opět sledujeme hranici a anotujeme příslušné hraniční pixely dle 2.31 a jelikož je poslední nalezená hranice vnitřní hranicí, na základě tabulky výše rozhodneme, že rodičovskou hranicí této nově nalezené hranice je rodičovská hranice poslední nalezené hranice, tudíž vnější hranice 2. Obnovíme opět rastrové snímání, které probíhá do té doby, než narazíme na pixel splňující podmínku počátečního pixelu hranice.

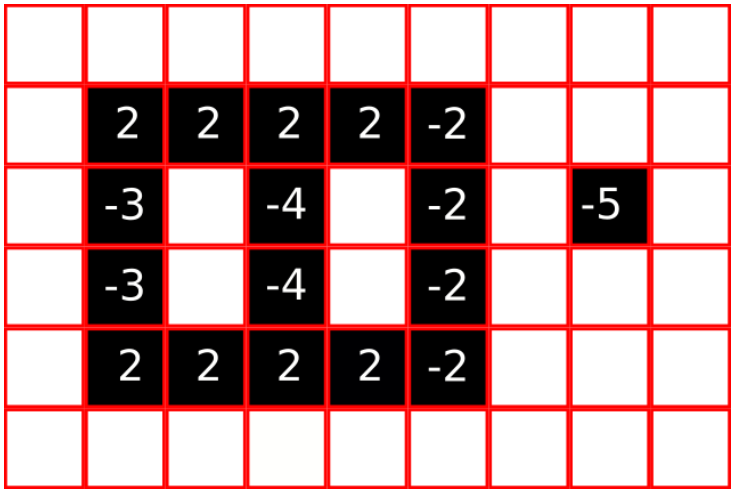


Obrázek 2.43: Chod Suzukiho algoritmu

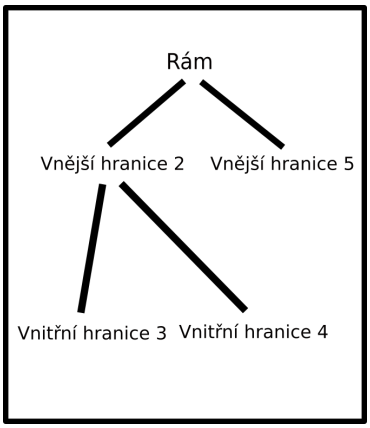


Obrázek 2.44: Hierarchická struktura kontur

Výsledná podoba anotovaných pixelů a hierarchické struktury hranic po skončení Suzukiho algoritmu vypadá následovně.



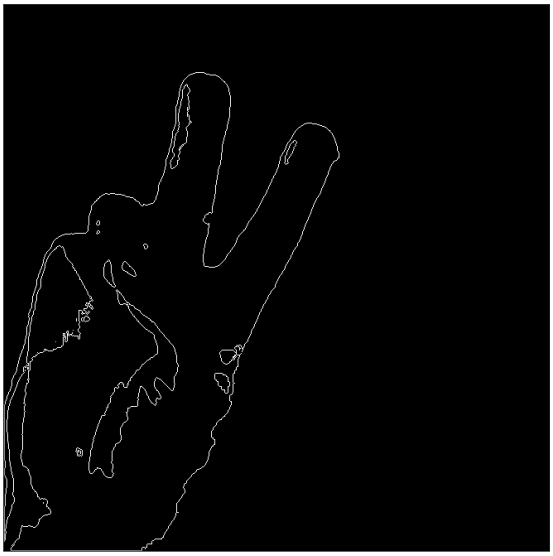
Obrázek 2.45: Chod Suzukiho algoritmu



Obrázek 2.46: Hierarchická struktura kontur

Výstupem knihovní fuknce findContours() z Python knihovny OpenCV, založené na Suzukiho algoritmu, je právě hierarchická struktura kontur. Při její aplikaci na binární obraz 2.47 dostáváme všechny vnější i vnitřní kontury znázorněné na obrázku 2.48.

Obrázek 2.47: Binární obraz gesta



Obrázek 2.48: Výsledek funkce findContours()

2.5 Fourierovy deskriptory

2.5.1 Odvození diskrétní Fourierovy transformace

Cílem této kapitoly je na základě poznatků z Matematické analýzy odvodit předpis pro diskrétní Fourierovu transformaci – zkráceně DFT z anglického discrete Fourier transform. DFT obecně převádí konečnou posloupnost rovnoměrně rozložených vzorků funkce na stejně početnou posloupnost opět rovnoměrně rozložených vzorků Fourierovy transformace v diskrétním čase (zkratka DTFT – z anglického

discrete-time Fourier transform), což je obecně komplexní funkce frekvence. Pro potřeby této kapitoly budeme uvažovat j jakožto komplexní jednotku.

2.5.1.1 Fourierova transformace

Fourierova transformace (zkratka FT z anglického Fourier transform) je integrální transformace, která slouží k dekompozici funkce do jejích frekvenčních komponentů. Nejčastěji se využívá k převedení časového spektra signálu na jeho frekvenční. Pod pojmem Fourierova transformace pak obecně rozumíme zobrazení $\mathcal{F}$, které mapuje spojitou funkci $f(t)$ na spojitou funkci $F(\mu)$. Funkci $F(\mu)$ pak nazýváme Fourierovou transformací funkce $f(t)$. Fourierova transformace je definována dle předpisu

$$\mathcal{F}(f(t)) = F(\mu) = \int_{-\infty}^{+\infty} f(t) \cdot e^{-j2\pi\mu t} \, \mathrm{d}t \tag{2.32}$$

Obdobně lze pak definovat inverzní Fourierovu transformaci, která mapuje Fourierovu transformaci funkce f zpět na f.

$$\mathcal{F}^{-1}(F(\mu)) = f(t) = \int_{-\infty}^{+\infty} F(\mu) \cdot e^{j2\pi\mu t} \, \mathrm{d}\mu \tag{2.33}$$

Funkci $f(t)$ nazýváme originál a funkci $F(\mu)$ obraz. Společně pak $f(t)$ a $F(\mu)$ nazýváme dvojice Fourierovy transformace.

2.5.1.2 Konvoluce

Konvoluce je binární matematický operátor. Spojitá konvoluce $\star$ dvou jednorozměrných funkcí $f(x)$ a $g(x)$ je definována vztahem

$$(f \star g)(x) = \int_{-\infty}^{+\infty} f(t)g(x-t) \, \mathrm{d}t \tag{2.34}$$

Funkci $g(x)$ pak nazýváme konvoluční jádro. Konvoluce je často využívaná například v algoritmech uplatňovaných v teorii zpracování obrazu, kde má diskrétní konvoluce daný předpis

$$(f \star g)(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} f(x-i, y-j) \cdot h(i, j) \tag{2.35}$$

kde dvourozměrnou funkci $h(x, y)$ lze chápat jako matici (konvoluční masku), kterou postupně přikládáme na jednotlivé pixely obrazu. Při každém přiložení vždy vynásobíme intenzity na sebe přilehlých pixelů a provedeme součet všech těchto součinů. Výslednou hodnotu pak uložíme jako intenzitu pixelu, na který jsme konvoluční masku přiložili.

Fourierovou transformací konvoluce dvou spojitých funkcí $f(x)$ a $g(x)$ je pak součin jejich Fourierových transformací.

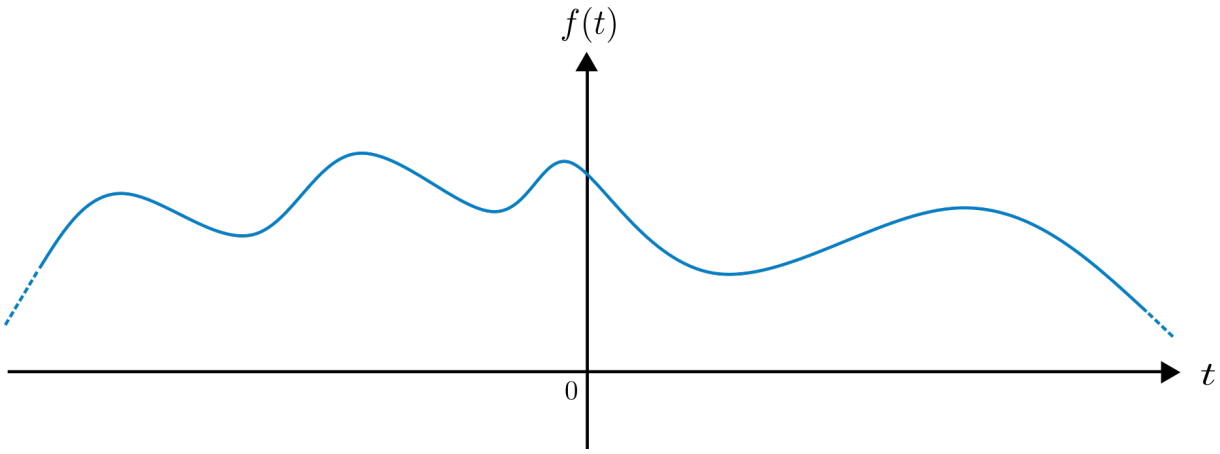
$$\mathcal{F}\Big((f \star g)(x)\Big) = (F \cdot G)(\mu), \quad \text{kde } \mathcal{F}\Big(f(x)\Big) = F(\mu) \text{ a } \mathcal{F}\Big(g(x)\Big) = G(\mu) \tag{2.36}$$

Obdobně pak Fourierova transformace součinu spojitých funkcí $f(x)$ a $g(x)$ je spojitá konvoluce jejich Fourierových transformací.

$$\mathcal{F}\Big((f \cdot g)(x)\Big) = (F \star G)(\mu), \quad \text{kde } \mathcal{F}\Big(f(x)\Big) = F(\mu) \text{ a } \mathcal{F}\Big(g(x)\Big) = G(\mu) \tag{2.37}$$

2.5.1.3 Vzorkování funkce

Pro manipulaci se spojitými funkcemi na počítači je nutné převést je na posloupnost jejich funkčních hodnot, protože nejsme schopni uložit do paměti počítače spojité množství hodnot. Členy těchto posloupností jsou pak funkční hodnoty původních spojitých funkcí vyčíslené v diskrétně a zpravidla ekvidistantně rozložených bodech (vzorcích). Tuto diskretizaci získáme následujícím vzorkováním (diskretizováním) dané funkce. Uvažujme spojitou funkci $f(t)$ na $(-\infty, +\infty)$, kterou chceme vzorkovat v ekvidistantních bodech. Její graf vypadá následovně.



Obrázek 2.49: Graf funkce $f(t)$

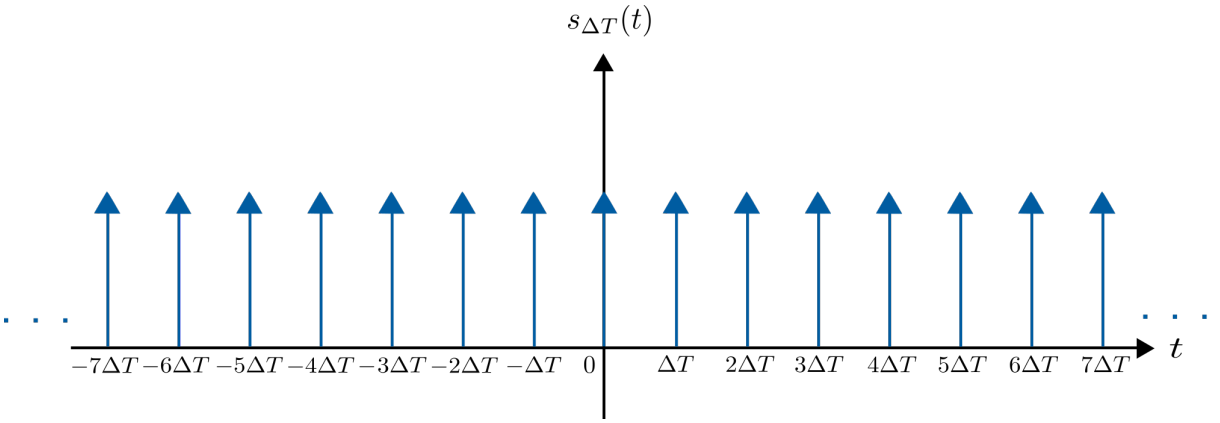
Označme ΔT vzdálenost dvou sousedních vzorků. Vzorkovaná funkce $\tilde{f}$ funkce f je pak definována následovně

$$\tilde{f}(t) = f(t)s_{\Delta T}(t) = \sum_{n=-\infty}^{+\infty} f(t) \cdot \delta(t - n\Delta T) \tag{2.38}$$

kde δ je Diracovo delta (v teorii zpracování signálu též nazývané jako Diracův jednotkový impuls). Diracovo delta je definované jako

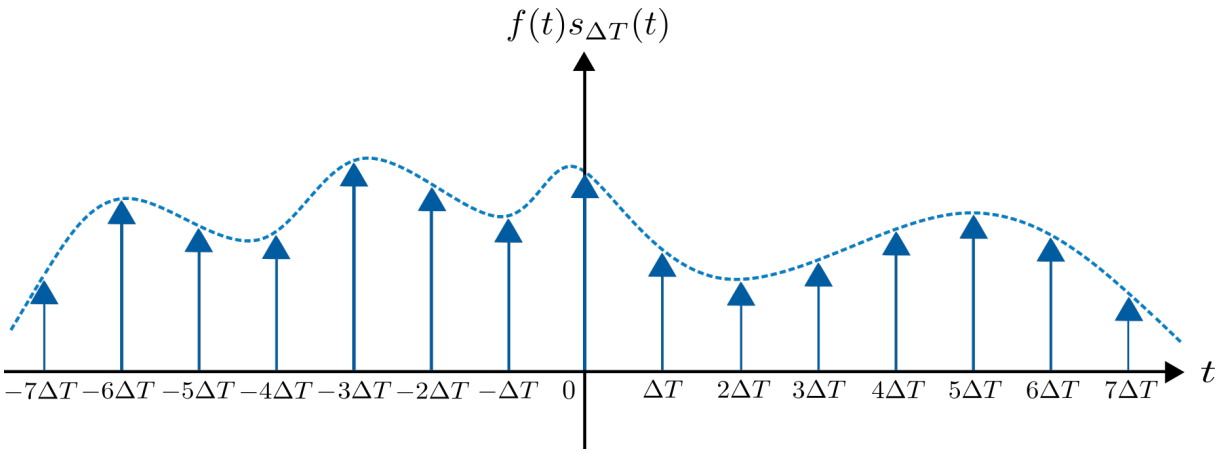
$$\delta(x) = \begin{cases} +\infty & \text{pro } x = 0 \\ 0 & \text{pro } x \neq 0 \end{cases} \tag{2.39}$$

Funkce $s_{\Delta T}(t)$ má následující podobu ve formě impulsů o velikosti jedna.



Obrázek 2.50: Graf funkce $s_{\Delta T}(t)$

Potom funkce $\tilde{f} = f(t)s_{\Delta T}(t)$ vypadá následovně

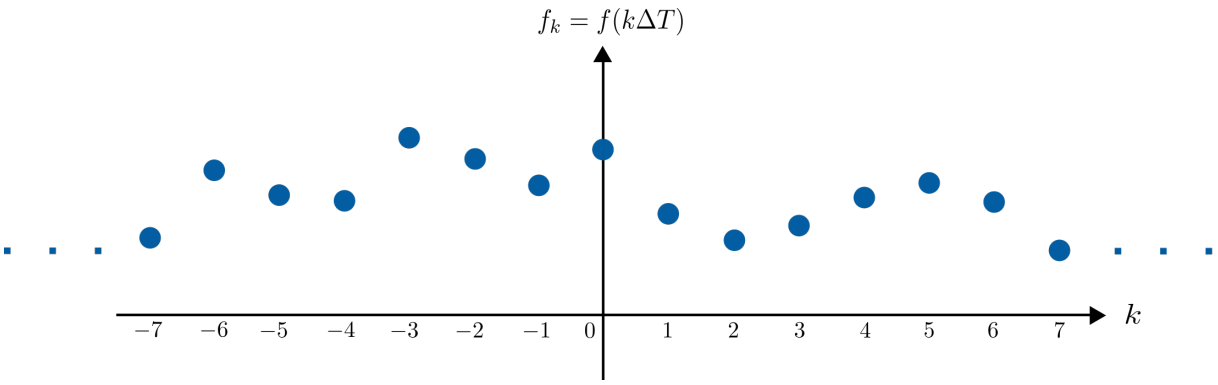


Obrázek 2.51: Graf funkce $\tilde{f} = f(t)s_{\Delta T}(t)$

Zároveň velikost k-tého impulsu f_k odpovídá následujícímu vztahu

$$f_k = \int_{-\infty}^{+\infty} f(t) \cdot \delta(t - k\Delta T)\,\mathrm{d}t = f(k\Delta T), \quad \text{kde } k \in \mathbb{Z} \tag{2.40}$$

Jednotlivé impulsy f_k, kde $k \in \mathbb{Z}$ jsou zobrazeny na následujícím obrázku 2.52.



Obrázek 2.52: Podoba impulsů f_k

S využitím znalosti konvoluce a rovnice (2.37) pak můžeme odvodit, jak bude vypadat Fourierova transformace vzorkované funkce $\tilde{f}$.

$$\mathcal{F}\left(\tilde{f}(t)\right) = \mathcal{F}\left(f(t) \cdot s_{\Delta T}(t)\right) = (F \star S)(\mu) \tag{2.41}$$

kde $S(\mu)$ má následující předpis

$$S(\mu) = \frac{1}{\Delta T} \sum_{n=-\infty}^{+\infty} \delta\left(\mu - \frac{n}{\Delta T}\right) \tag{2.42}$$

Potom dosazením předpisu pro $S(\mu)$ z (2.42) do pravé strany rovnice (2.41) dostáváme

$$(F \star S)(\mu) = \int_{-\infty}^{\infty} F(\tau)S(\mu - \tau)\,\mathrm{d}\tau = \frac{1}{\Delta T} \int_{-\infty}^{+\infty} F(\tau) \sum_{n=-\infty}^{\infty} \delta\left(\mu - \tau - \frac{n}{\Delta T}\right)\mathrm{d}\tau \tag{2.43}$$

Následnými úpravami potom

$$\frac{1}{\Delta T}\int_{-\infty}^{+\infty} F(\tau)\sum_{n=-\infty}^{\infty}\delta\left(\mu-\tau-\frac{n}{\Delta T}\right)\mathrm{d}\tau=\frac{1}{\Delta T}\sum_{n=-\infty}^{\infty}F\left(\mu-\frac{n}{\Delta T}\right) \tag{2.44}$$

Z výrazu na pravé straně rovnice (2.44) plyne, že Fourierova transformace vzorkované funkce $\tilde{f}$ je spojitá a periodická funkce s periodou $\frac{1}{\Delta T}$.

2.5.1.4 Diskrétní Fourierova transformace

V následující sekci využijeme poznatky ze sekcí předešlých a pokusíme se odvodit předpis pro diskrétní Fourierovu transformaci. V předešlé sekci 2.5.1.3 jsme vyjádřili předpis pro Fourierovu transformaci vzorkované funkce $\tilde{f}$ v závislosti na Fourierově transformaci původní funkce f. Nyní se pokusme vyjádřit $\tilde{F}=\mathcal{F}(\tilde{f})$ v závislosti právě na f. Aplikujme tedy Fourierovu transformaci na $\tilde{f}$ dle předpisu (2.32)

$$\tilde{F}=\int_{-\infty}^{+\infty}\tilde{f}(t)\cdot e^{-j2\pi\mu t}\,\mathrm{d}t \tag{2.45}$$

Dále rozepišme $\tilde{f}$ dle předpisu (2.38)

$$\int_{-\infty}^{+\infty}\tilde{f}(t)\cdot e^{-j2\pi\mu t}\,\mathrm{d}t=\int_{-\infty}^{+\infty}\left(\sum_{n=-\infty}^{+\infty}f(t)\delta(t-n\Delta T)e^{-j2\pi\mu t}\right)\mathrm{d}t \tag{2.46}$$

Dále zaměníme sumu a integrál

$$\int_{-\infty}^{+\infty}\left(\sum_{n=-\infty}^{+\infty}f(t)\delta(t-n\Delta T)e^{-j2\pi\mu t}\right)\mathrm{d}t=\sum_{n=-\infty}^{+\infty}\int_{-\infty}^{+\infty}f(t)\delta(t-n\Delta T)e^{-j2\pi\mu t}\,\mathrm{d}t \tag{2.47}$$

Nyní využijeme vztahu pro velikost k-tého impulsu vzorkované funkce (2.40) (konkrétně vlastnosti, že Diracovo delta působí při integraci jako jednotkový operátor – anglicky "sifting property").

$$\sum_{n=-\infty}^{+\infty}\int_{-\infty}^{+\infty}f(t)\delta(t-n\Delta T)e^{-j2\pi\mu t}\,\mathrm{d}t=\sum_{n=-\infty}^{+\infty}f_n\cdot e^{-j2\pi\mu n\Delta T} \tag{2.48}$$

Tím dostáváme výsledný předpis pro Fourierovu transformaci vzorkované funkce $\tilde{f}$

$$\tilde{F}=\sum_{n=-\infty}^{+\infty}f_n\cdot e^{-j2\pi\mu n\Delta T} \tag{2.49}$$

Ze sekce 2.5.1.3 víme, že $\tilde{F}$ je spojitá periodická s periodou $\frac{1}{\Delta T}$. Stačí nám proto k charakterizaci $\tilde{F}(\mu)$ pouze znalost jedné její periody.

Vezměme si bez újmy na obecnosti funkci $\tilde{F}(\mu)$ na intervalu $(0,\frac{1}{\Delta T})$ a vzorkujme tuto funkci na zvoleném intervalu M vzorky. To odpovídá dělení intervalu s dělícími body $\mu_0,...,\mu_{M-1}$ definovanými následovně

$$\mu_i=\frac{i}{M\Delta T},\quad i\in\{0,...,M-1\} \tag{2.50}$$

Dosazením předpisů pro jednotlivé μ_i za μ do předpisu (2.49) dostáváme výslednou podobu diskrétní Fourierovy transformace

$$\tilde{F}_i=\sum_{n=0}^{M-1}f_n e^{\frac{-j2\pi in}{M}} \tag{2.51}$$

Pro zjednodušení značení budeme nadále psát $\tilde{F}$ bez vlnky. Máme-li množinu $\{f_n\}_{n=0}^{M-1}$ obsahující M vzorků funkce $f(t)$, použitím diskrétní Forierovy transformace (2.51) dostaneme posloupnost $\{F_m\}_{m=0}^{M-1}$ M obecně komplexníxh čísel, které odpovídají Fourierově transformaci vstupním vzorkům $\{f_n\}_{n=0}^{M-1}$ funkce $f(t)$.

Na druhou stranu je možné z obecně komplexní posloupnosti $\{F_m\}_{m=0}^{M-1}$ získat posloupnost $\{f_n\}_{n=0}^{M-1}$, použitím inverzní diskrétní Fourierovo transformace definované předpisem

$$f_n = \frac{1}{M} \sum_{m=0}^{M-1} F_m \cdot e^{j\frac{2\pi m}{M}n}, \quad n \in \{0, ..., M-1\} \tag{2.52}$$

Substitucí f_n v (2.51) z rovnice (2.52) a dalšími úpravami zjistíme, že $F_m \equiv F_m$. Analogicky substituce F_m v (2.52) za předpis pro F_m z (2.51) dává $f_n \equiv f_n$. Z toho plyne, že f_n a F_m tvoří dvojici diskrétní Fourierovy transformace a co víc, existuje diskrétní Fourierova transformace i inverzní diskrétní Fourierova transformace pro libovolnou posloupnost vzorků s konečnými hodnotami.

Při popisu diskrétní Fourierovy transformace jsme využívali indexy n a m pro popis diskrétních hodnot f_n a F_m. Ovšem v teorii zpracování obrazu je více intuitivní značení x a y, zejména pro celočíselné nezáporné souřadnice pixelů v digitálním obrazu. Analogicky u a v jsou celočíselné nezáporné frekvenční proměnné. Proto se pro budoucí použití hodí následující přeznačení diskrétní Fourierovy transformace (2.51)

$$F(u) = \sum_{x=0}^{M-1} f(x) \cdot e^{-j\frac{2\pi u}{M}x}, \quad u \in \{0, ..., M-1\} \tag{2.53}$$

a analogicky přeznačení inverzní diskrétní Fourierovy transformace (2.52)

$$f(x) = \frac{1}{M} \sum_{u=0}^{M-1} F(u) \cdot e^{j\frac{2\pi u}{M}x}, \quad x \in \{0, ..., M-1\} \tag{2.54}$$

V praxi diskrétní Fourierovu transformaci (i její inverzi) počítáme pomocí FFT algoritmu (z anglického fast Fourier transform).

2.5.2 Fourierovy deskriptory

Fourierovy deskriptory, odvozené z Fourierových řad, jsou obecně komplexní čísla používaná k charakterizaci složitosti tvaru a dalších geometrických atributů kontury objektu. Uvažujme hranici (posloupnost hraničních pixelů)

$$H := \left\{(x_i, y_i)\right\}_{i=0}^{K-1} \tag{2.55}$$

Kde x a y jsou souřadnice pixelů v obrazu a K je délka (počet pixelů) kontury H. Uvažujme dále zobrazení $x(i) := x_i$ a $y(i) := y_i$, která indexu $i \in \{0, ..., K-1\}$ přiřazují x-ovou, respektive y-ovou souřadnici pixelu hranice s indexem i. Tímto způsobem a využitím znalosti komplexních čísel lze vyjádřit hranice H jako posloupnost hraničních pixelů $\{h(i)\}_{i=0}^{K-1}$, jejíž členy mají následující předpis

$$h(i) = x(i) + jy(i) \qquad i \in \{0, ..., K-1\} \tag{2.56}$$

Kde j je komplexní jednotka a zobrazení $x(i)$ a $y(i)$ jsou pak průměty souřadnic i-tého pixelu hranice do směru reálné, respektive imaginární osy v komplexní rovině. Tímto přeformulováním redukujeme dvourozměrný problém (každý pixel hranice je charakterizován dvěma souřadnicemi) na jednorozměrný (každý pixel hranice lze vyjádřit příslušným komplexním číslem v komplexní rovině).

Zároveň z kapitoly 2.5.1 máme předpis pro diskrétní Fourierovu transformaci hranice H vyjádřenou pomocí posloupnosti pixelů v komplexní reprezentaci

$$a(u) = \sum_{i=0}^{K-1} h(i) \cdot e^{-j\frac{2\pi i}{K}u}, \quad u \in \{0, ..., K-1\} \tag{2.57}$$

Komplexní koeficienty $a(u)$ nazýváme Fourierovy deskriptory hranice. Pomocí inverzní Fourierovy transformace Fourierových deskriptorů můžeme rekonstruovat původní hranici $H = \{h(i)\}_{i=0}^{K-1}$ následovně

$$h(i) = \frac{1}{K} \sum_{u=0}^{K-1} a(u) \cdot e^{j\frac{2\pi i}{K}u}, \quad i \in \{0, ..., K-1\} \tag{2.58}$$

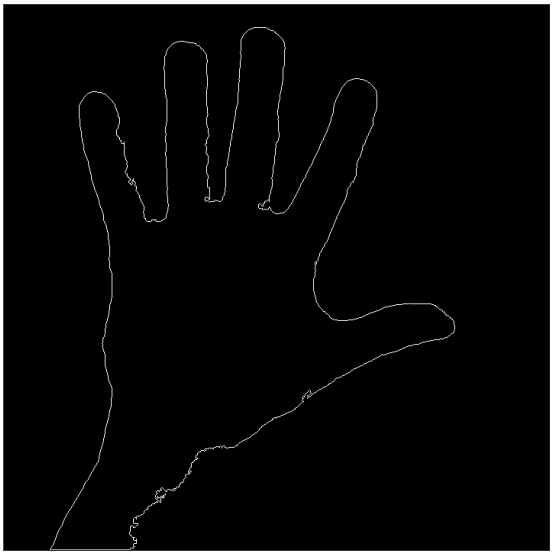
Uvažujme ovšem prvních P Fourierových deskriptorů. Jinými slovy Fourierovy deskriptory $a(u)$ pro $u \geq P$ položíme rovno nule. Výsledkem inverzní Fourierovy transformace aplikované na prvních P Fourierových deskriptorů hranice H pak není identická rekonstrukce hranice H, nýbrž pouze její aproximace $\widehat{H} := \{\hat{h}(i)\}_{i=0}^{K-1}$, jejíž členy jsou inverzní Fourierovou transformací definované následovně

$$\hat{h}(i) = \frac{1}{K} \sum_{u=0}^{P-1} a(u) \cdot e^{j\frac{2\pi i}{K}u}, \quad i \in \{0, ..., K-1\} \tag{2.59}$$

Přestože jsme pro konstrukci $\widehat{H}$ využili pouze P Fourierových deskriptorů hranice H, je délka (počet pixelů) posloupnosti $\widehat{H}$ rovna délce H.

V praxi vysokofrekvenční Fourierovy deskriptory v jistém smyslu reprezentují detail kontury, kdežto nízkofrekvenční členy mapují její celkový tvar. Uvažujme binární obrázek gesta (viz obrázek 2.53), jeho vnější kontura je znázorněna na obrázku 2.54.

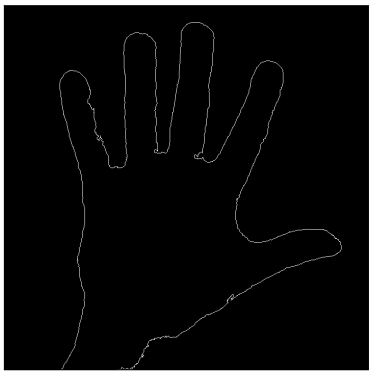
Obrázek 2.53: Binární obraz gesta



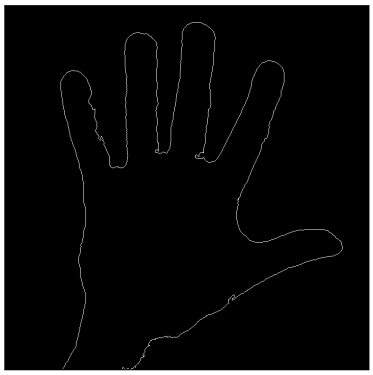
Obrázek 2.54: Kontura gesta

Podívejme se, jak bude vypadat rekonstuovaná kontura při zachování různých počtů prvních P Fourierových deskriptorů (zbylé nastavíme na nulu). Konturu tvoří celkem 3363 pixelů. Spočítáme-li Fourierovy deskriptory celé kontury, dostaneme 3363 Fourierových deskriptorů. Rekonstrujme nyní zpět kontury ruky při zachování prvních 3363 deskriptorů, 1682 deskriptorů, 338 deskriptorů, 68 deskriptorů,

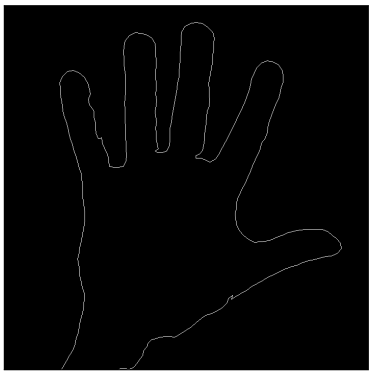
34 deskriptorů, 18 deskriptorů, které odpovídají 100%, 50%, 10%, 2%, 1% a 0,5% z celkového počtu 3363 deskriptorů původní kontury. Můžeme si všimnout, že při ponechání menšího počtu deskriptorů se snižuje úroveň detailu v rekonstruované kontuře gesta.



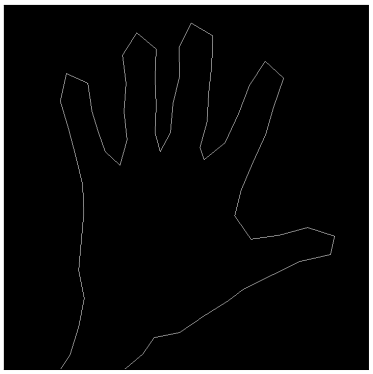
Obrázek 2.55: Rekonstrukce kontury z prvních 3363 deskriptorů (100% deskriptorů gesta)



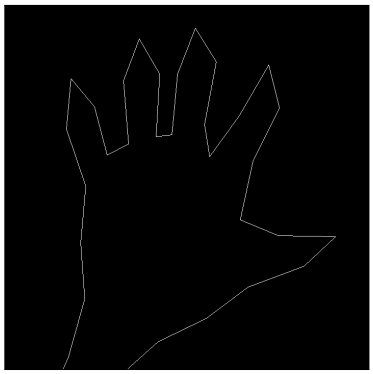
Obrázek 2.56: Rekonstrukce kontury z prvních 1682 deskriptorů (50% deskriptorů gesta)



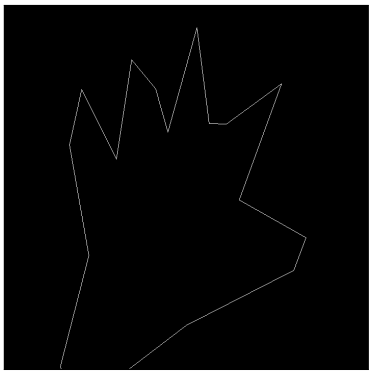
Obrázek 2.57: Rekonstrukce kontury z prvních 338 deskriptorů (10% deskriptorů gesta)



Obrázek 2.58: Rekonstrukce kontury z prvních 68 deskriptorů (2% deskriptorů gesta)



Obrázek 2.59: Rekonstrukce kontury z prvních 34 deskriptorů (1% deskriptorů gesta)



Obrázek 2.60: Rekonstrukce kontury z prvních 18 deskriptorů (0,5% deskriptorů gesta)

Při klasifikaci gest je kladen důraz právě na celkový tvar daného gesta, proto je žádoucí porovnávat kontury pouze na základě jejich nízkofrekvenčních Fourierových deskriptorů a vysokofrekvenční členy neuvažovat (nulovat). Platí, že pokud jsou Fourierovy deskriptory seřazeny podle jejich velikosti v absolutní hodnotě, pak čím méně prvních P členů vezmeme, tím více detailu hranice ztratíme. Nulování vysokofrekvenčních Fourierových deskriptorů pak má svou analogii ve filtrování Fourierovo transformace ideálním lowpass filterem.

Při klasifikaci na základě Fourierových deskriptorů kontury je žádoucí, aby byly pokud možno co nejvíce invariantní vůči translaci, rotaci, škálování a také vůči volbě počátečnímu pixelu kontury. Ačkoliv Fourierovy deskriptory obecně nejsou vůči těmto transformacím invariatní, jsou tyto transformace kontury jednoduše vyjádřitelné v podobě transformací jednotlivých deskriptorů. Uvažujme například rotaci komplexního čísla c v komplexní rovině podle počátku souřadného systému o úhel θ. Rotované komplexní číslo c' pak získáme přenásobením c konstantou $e^{j\theta}$. Tímto způsobem můžeme rotovat každý pixel kontury H a tím získat rotovanou konturu H' podle počátku o úhel θ. Fourierovy deskriptory kon-

tury H' pak budou vypadat následovně

$$a_r(u) = \sum_{i=0}^{K-1} h(i) \cdot e^{j\theta} \cdot e^{-j\frac{2\pi i}{K}u} = a(u) \cdot e^{j\theta}, \quad u \in \{0, ..., K-1\} \tag{2.60}$$

Z toho plyne, že rotace kontury H o úhel θ podle počátku ovlivní Fourierovy deskriptory kontury H jejich přenásobením faktorem $e^{j\theta}$. Obdobně se dají odvodit i zbylé transformace, jejichž podobu shrnuje následující tabulka

transformace	$\boldsymbol{h(i)}$	$\boldsymbol{a(u)}$
identita	$h(i)$	$a(u)$
rotace	$h_r(i) = h(i)e^{j\theta}$	$a_r(u) = a(u)e^{j\theta}$
translace	$h_t(i) = h(i) + \Delta_{xy}$	$a_t(u) = a(u) + \Delta_{xy}\delta(u)$
škálování	$h_s(i) = \alpha h(i)$	$a_s(u) = \alpha a(u)$
počáteční pixel kontury	$h_p(i) = h(i - i_0)$	$a_p(u) = a(u)e^{-j\frac{2\pi i_0}{K}u}$

Tabulka 2.2: Transformace Fourierových deskriptorů

Kde $\Delta_{xy} := \Delta x + j\Delta y$. Potom značení $h_t(i) = h(i) + \Delta_{xy}$ indikuje translaci posloupnosti v následujícím smyslu

$$h_t(i) = (x(i) + \Delta x) + j(y(i) + \Delta y) \tag{2.61}$$

Jak je vidět z předpisu $a_t(u) = a(u) + \Delta_{xy}\delta(u)$, kde δ je Diracovo delta, translace kontury H o Δ_{xy} má vliv pouze na nultý Fourierův deskriptor. Značení $h_p(i) = h(i - i_0)$ znamená $h_p(i) = x(i - i_0) + jy(i - i_0)$ a v praxi posouvá index počátečního pixelu kontury z $i = 0$ na $i = i_0$. Z tabulky lze pozorovat, že změna počátečního pixelu kontury H postihuje Fourierovy deskriptory kontury H různým, avšak známým způsobem (ve smyslu, že násobení jednotlivých Fourierových deskriptorů $a(u)$ faktorem $e^{-j\frac{2\pi i_0}{K}u}$ závisí na u).

Kapitola 3

Klasifikace gesta s využitím klasických metod zpracování obrazu a knihovny MediaPipe

Cílem této kapitoly je popsat segmentaci a následnou klasifikaci gesta na základě metod zpracování obrazu. Následně pak porovnat výsledky klasifikace gest s využitím tohoto přístupu, s výsledky klasifikace běžnějším způsobem výhradně pomocí neuronových sítí. Pro segmentaci gesta využijeme znalosti intenzit pixelů kůže obličeje. Pro zjednodušení postupu segmentace gesta využijeme pro detekci obličeje v obrazu knihovnu MediaPipe [13], která je založena na modelu neuronových sítí. Popis zmíněného přístupu je rozdělen do dvou chronologicky navazujících podkapitol. Pod pojmem gesto rozumíme i nadále gesto statické. Implementace všech metod z celé praktické části byla provedena v jazyce Python.

3.1 Segmentace gesta

Základem úspěšné klasifikace gesta je jeho optimální segmentace. Navržený postup se opírá o článek [10], ve kterém autoři ze znalosti intenzity pixelů detekovaného obličeje byli schopni na základě zpětné projekce histogramu namapovat pixely gesta (kůže na ruce). K detekci obličeje využijeme implementaci knihovny MediaPipe v jazyce Python. Na základě znalosti pozice obličeje pak metodami zpracování obrazu odfiltrujeme vzorek referenčních pixelů kůže na obličeji, na základě kterých pak pomocí modifikované zpětné projekce histogramu detekujeme pixely gesta (kůže na ruce). Tyto pixely následně segmentujeme a dostaneme binární obraz, kde bílé pixely reprezentují gesto (pixely kůže) a černé pixely pozadí. Dále využijeme algoritmy hledání kontur v binárních obrazech pro získání kontury gesta. Kontura pak následně poslouží ke klasifikaci gesta.

3.1.1 MediaPipe

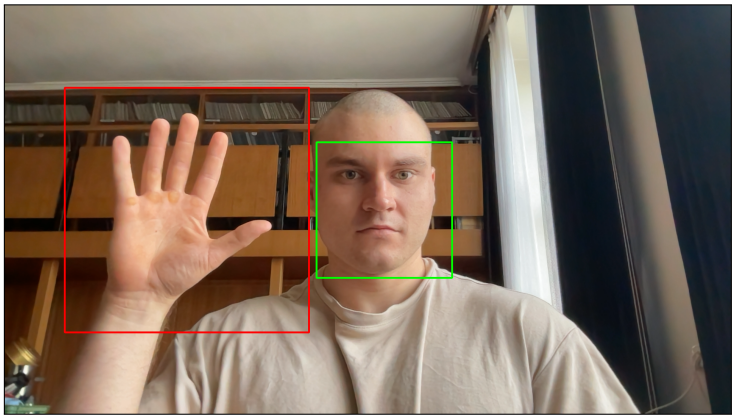
MediaPipe [13] je open source multiplatformní knihovna využívající metod strojového učení pro řešení úloh typu detekce obličeje, trasování rukou, detekce objektů, odhad postury těla a v neposlední řadě je také schopna umisťovat polyhedrální sítě na povrch obličeje. Ačkoliv knihovna MediaPipe využívá k detekci modely neuronových sítí, její využití nám poslouží k výraznému ulehčení segmentace gesta. Navíc znalost pozice obličeje funkcemi, které detekci obličeje umožňují, z knihovny MediaPipe pak jednak slouží jako orientační bod pro dron, jednak pro umístění oblasti vedle obličeje, ve které se pak gesto segmentuje a klasifikuje. Tato oblast, ve které gesto hledáme, zároveň slouží jako bezpečnostní

opatření. Kdybychom detekovali gesto v celém obrazu, který nám poskytuje kamera dronu, mohlo by dojít k falešné detekci a klasifikaci gesta, aniž by to uživatel zamýšlel.

Výstupem modulu pro detekci obličeje knihovny MediaPipe může být ohraničující obdélník (anglicky bounding box) nebo jiné klíčové body například v oblasti očí, nosu, úst nebo uší. Cílem je vyfiltrovat pixely obličeje, které jsou s co nejvyšší pravděpodobností pixely kůže a zbavit se tak pixelů v oblasti vlasů, uší, vousů a ostatních periferních oblastí. Proto právě zmíněný ohraničující obdélník a klíčový bod v oblasti úst využijeme. Obdélník z obou bočních stran zúžíme o 30 pixelů, horní stranu obdélníku též posuneme dolů o 10 pixelů a dolní hranu obdélníku bude tvořit klíčový bod úst posunutý nahoru o 50 pixelů. Výsledkem tohoto ořezu pak bude obraz, který bude obsahovat oblast kolem očí, nosu a tváří. Tento obraz vypadá následovně.

Obrázek 3.1: Výřez obličeje

Výřez obličeje před dalším zpracováním rozmažeme s využitím Gaussova filtru [6] a tím eliminujeme případné zašumění obrazu. Zároveň vedle ohraničujícího obdélníku obličeje umístíme další obdélník s adaptivní velikostí, který slouží jako výřez obrazu pro gesto. V tomto obdélníku budeme v následujícím postupu gesto segmentovat. Celkový obraz pak vypadá následovně.



Obrázek 3.2: Obraz z kamery dronu obsahující ohraničující obdélníky obličeje a gesta

3.1.2 Segmentace referenčních pixelů kůže

Právě na tvářích a v oblasti kolem očí se nachází nejvíce optimálních referenčních pixelů kůže, proto je žádoucí jich co možno nejvíce segmentovat. K jejich segmentaci využijeme prahování Otsuovo metodu. Nejprve je třeba obraz převést z barevného na 8-bitový, následně ho lze prahovat Otsuovo metodou. Při vnějším osvětlení, působící na obličej ze strany, vrhá nos a prohlubně kolem očí neostré stíny, které mají nepříznivý vliv na prahování celého obrazu obličeje Otsuvo metodou. Proto může vlivem prahování dojít ke ztrátě velké části pixelů kůže. Pro zlepšení výsledku prahování za takových podmínek také využijeme regionální podobu Otsuovy metody. Obraz rovnoměrně rozdělíme na dílčí obdélníky například v takovém poměru aby bylo umístěno 5 obdélníků vertikálně a 4 horizontálně. Na každý z těchto obdélníků pak použijeme Otsuovu metodu. Výsledkem je pak do jisté míry regionální segmentace jednotlivých pixelů kůže. Bitovým součtem regionálně prahovaného obrazu Otsuovo metodou s obrazem, který celý prahujeme Otsuovo metodou, dostaneme výsledný obraz. V takto vzniklém binárním obraze, kde bílé pixely reprezentují prahované pixely kůže, se ovšem častokrát nachází pixely světlých periferních částí obočí. Ty eliminujeme použitím morfologické tranformace Eroze na prahovaný binární obraz. Bílé pixely ve výsledném obraze pak nahradíme původními barevnými v jejich RGB reprezentaci. Jak již bylo zmíněno, knihovna MediaPipe poskytuje informaci o pozici očí. Abychom nemuseli obtížně prahovat bělmo očí a duhovky, využijeme tuto informaci a kolem očí umístíme černé kruhy s adaptivně měnícím se poloměrem, který závisí na velikosti obdélníku, který ohraničuje obličej. Výsledný obraz pak vypadá následovně.

Obrázek 3.3: Segmentovaný výřez obličeje

3.1.3 Srovnání barevných prostorů

S přihlédnutím k faktu, že scéna je často ovlivněna vnějším přírodním nebo umělým osvětlením, jeví se užitečné použít prostor barev, který je maximálně invariatní vůči tomuto osvětlení. Dle článku [11] se pixely kůže příhodněji shlukují v prostorech barev, kde lze oddělit luminanční (jasovou) složku od té chromatické. Takové prostory pak jsou méně citlivé na vnější osvětlení než například RGB prostor barev. Jedním z takových je YCrCb, kde Y je luminance (jas) a Cr, respektive Cb je chrominanční červená, respektive modrá složka. Využitím tohoto modelu je možné se do určité míry zbavit nežádoucích efektů luminance (způsobené různými typy vnějšího osvětlení) zanedbáním právě luminanční složky Y a uvažováním pouze složek Cr a Cb.

3.1.4 Modifikovaná zpětná projekce histogramu

Převedeme výsledný obraz (obraz obličeje) ze sekce 3.1.2 do zmíněného YCrCb prostoru barev. Stejně tak ze sekce 3.1.1 převedeme výřez pro gesto (obraz gesta) v podobě obdélníkový obrazu do YCrCb prostoru barev. Sestavíme normalizovaný histogram zpracovaného obrazu obličeje v CrCB rovině (Y složku neuvažujeme). Dále neuvažujeme odfiltrované nežádoucí pixely kůže, kterým jsme nastavili černou barvu (vynulujeme v histogramu hodnotu odpovídající Cr=128 a Cb=128). A provedeme zpětnou projekci histogramu složek Cr a Cb obrazu gesta s využitím normalizovaného histogramu obrazu obličeje v CrCb rovině.

Kdybychom histogram obrazu obličeje předem nijak neupravili, namapovaly by se s nenulovou hodnotou (pravděpodobností výskytu) jen ty pixely z obrazu gesta, které svou intenzitou odpovídají pixelům z obrazu obličeje. Přirozeně má ovšem kůže na ruce vlivem prokrvení odlišný odstín od kůže na obličeji. Navíc častokrát se stává, že obličej je jinak nasvícen než samotné gesto a tím se v závislosti na spektrálním složení působícího světla mění i hodnoty chromatičnosti Cr a Cb. Proto je třeba histogram obrazu obličeje v CrCb rovině před zpětnou projekcí histogramu modifikovat. Všimneme si, že dlaně jsou obecně více prokrvené, než obličej a mají červenější odstín. Konvolujme proto histogram obrazu obličeje vhodným strukturním elementem. Jako tento strukturní element zvolíme matici, kde jsou jedničky uspořádány do tvaru elipsy pootočené o 45° proti směru hodinových ručiček. Strukturní element pak vypadá jako následující matice s rozměry 12×20.

0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0

Tabulka 3.1: Strukturní element ve tvaru rotované elipsy

Vlivem konvoluce se histogram obrazu obličeje rozmaže a pixely s intenzitou v okolí shluku pixelů obličeje, které jsou svou intezitou podobné pixelům obličeje pak dostanou nenulovou hodnotu v histogramu obrazu obličeje. Dále zmíněnou volbou strukturního elementu se zároveň "roztáhneme"histogram ve směru jeho diagonály. Tímto způsobem při zpětné projekci histogramu namapujeme v obraze gesta výrazně více pixelů kůže, které mají většinou červenější odstín než pixely obličeje. Ovšem zvýší se tím i počet namapovaných pixelů pozadí podobných pixelům kůže, které ve skutečnosti pixely kůže nejsou. Počet těchto falešně detekovaných pixelů pozadí však výrazně snížíme při hledání kontur u binárního obrazu gesta (viz sekce 3.1.5). Výsledkem zpětné projekce histogramu je pak obraz s hodnotami intenzit na intervalu $[0, 1]$. Tyto hodnoty jednotlivých pixelů pak reprezentují relativní zastoupení pixelu o daných složkách Cr a Cb v obrazu obličeje. Normalizováním těchto hodnot na intervalu $[0, 1]$, přeškálováním hodnotou 255 a zaokrouhlením na nejbližší celé číslo dostáváme 8-bitový obraz. Poté prahováním určitou hodnotou prahu P pak dostaneme binární obraz gesta, kde pixely s hodnotou 1 reprezentují gesto a pixely s hodnotou 0 reprezentují pozadí.

3.1.5 Hledání kontury gesta

Na binární obraz gesta pak aplikujeme funkci findContours() z knihovny OpenCV, která využívá Suzukiův algoritmus hledání kontur. Výstupem funkce findContours() je hieararchická struktura kontur (posloupností hraničních pixelů). Uvažujme, že právě gesto je vždy co do obsahu největší homogenní objekt, který se nachází v analyzované oblasti. Proto lze z hierarchické struktury kontur najít tu, která má největší obsah a tu považovat za konturu gesta. Tímto způsobem výběru eliminujeme značnou část falešně detekovaných pixelů z pozadí, které se svou intenzitou podobají pixelům kůže.

3.2 Výsledky segmentace

Výsledky předchozího postupu segmentace byly porovnány s manuálně anotovaným datasetem gest.

3.2.1 Dataset

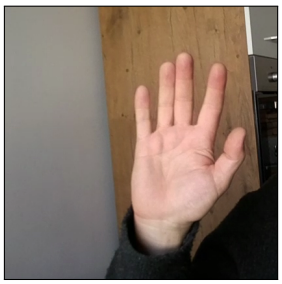
Dataset byl anotován s využitím platformy CVAT (Computer Vision Annotation Tool). CVAT je open source software sloužící k anotaci dat, které mohou být následně zpracovány pomocí algoritmů počítačového vidění. CVAT nám mimo jiné umožňuje manuálně anotovat gesta pomocí polygonu, jehož klíčové body jsou pak s dalšími metadaty uloženy. Tímto způsobem anotovaný dataset gest lze pak exportovat v různých formátech. Volili jsme formát COCO 1.0, jehož výstupem je soubor JSON obsahující identifikační čísla obrázků gest v datasetu, jejich rozlišení, název, souřadnice vrcholů ohraničujících polygonů gest, jejich obsah a další. Manuálně anotované polygony ohraničující jednotlivá gesta nazveme pro další účely jako referenční polygony gest. Příklady referenčních polygonů gest jsou pak znázorněny růžovou barvou na následujících obrázcích.

Obrázek 3.4: Referenční polynom gesta "Circle"

Obrázek 3.5: Referenční polynom gesta "Rocknroll"

Dataset pak obsahuje celkem 1602 anotovaných obrázků gest rozdělených do osmi tříd (typů gest). Třídy odpovídají jednotlivým názvům gest: "Five", "Four", "Three", "Two", "One", "Circle", "Rocknroll"a "Thumb". V každé třídě se pak nachází zbruba 200 obrázků gest pořízených z kamery dronu od deseti

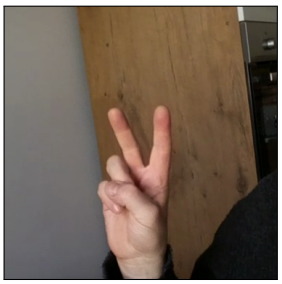
různých subjektů za odlišných světelných podmínek. Zástupci každé z tříd (od stejného subjektu) jsou znázorněny na následujících obrázcích.



Obrázek 3.6: Ukázka gesta "Five"

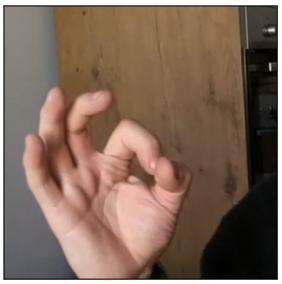
Obrázek 3.7: Ukázka gesta "Four"

Obrázek 3.8: Ukázka gesta "Three"



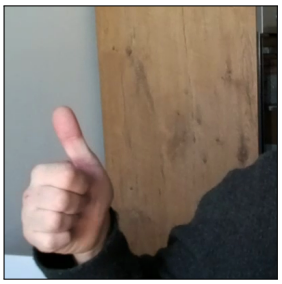
Obrázek 3.9: Ukázka gesta "Two"

Obrázek 3.10: Ukázka gesta "One"



Obrázek 3.11: Ukázka gesta "Circle"

Obrázek 3.12: Ukázka gesta "Rocknroll"

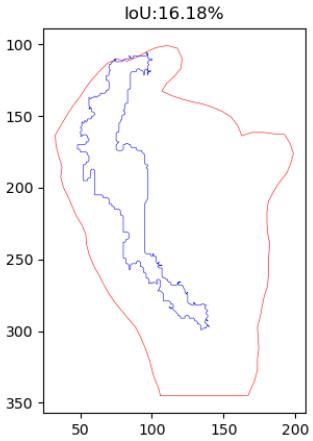


Obrázek 3.13: Ukázka gesta "Thumb"

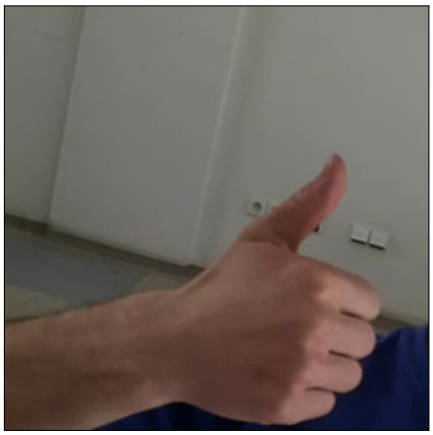
K vyhodnocení kvality segmentace využijeme konturu gesta, kterou jsme získali postupem popsaným v sekci 3.1.5. Tuto konturu gesta využijeme k hodnocení segmentace pomocí metody Intersection over Union (IoU). Metoda IoU počítá poměr mezi obsahem průniku dvou polygonů a jejich sjednocením, kde zmíněné dva polygony jsou právě referenční polygon daného gesta a jeho segmentací získaná kontura.

IoU referenčního polygonu a kontury gesta napočítáváme pro každé gesto z datasetu. Na následujících obrázcích jsou znázorněny dva případy segmentace a jejich příslušná hodnota IoU. Referenční polygon gesta je znázorněn červenou barvou, kdežto kontura gesta barvou modrou. Můžeme vidět, že pixely dlaně u špatně segmentovaného gesta na obrázku 3.14 jsou zastíněné, což je příčinou takovéto špatné segmentace.

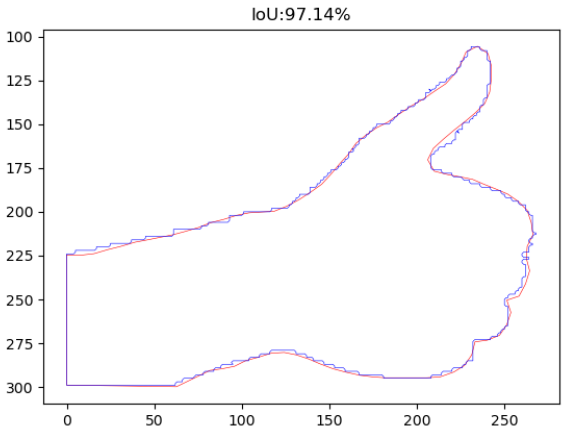
Obrázek 3.14: Původní gesto



Obrázek 3.15: Špatná segmentace původního gesta na obrázku 3.14

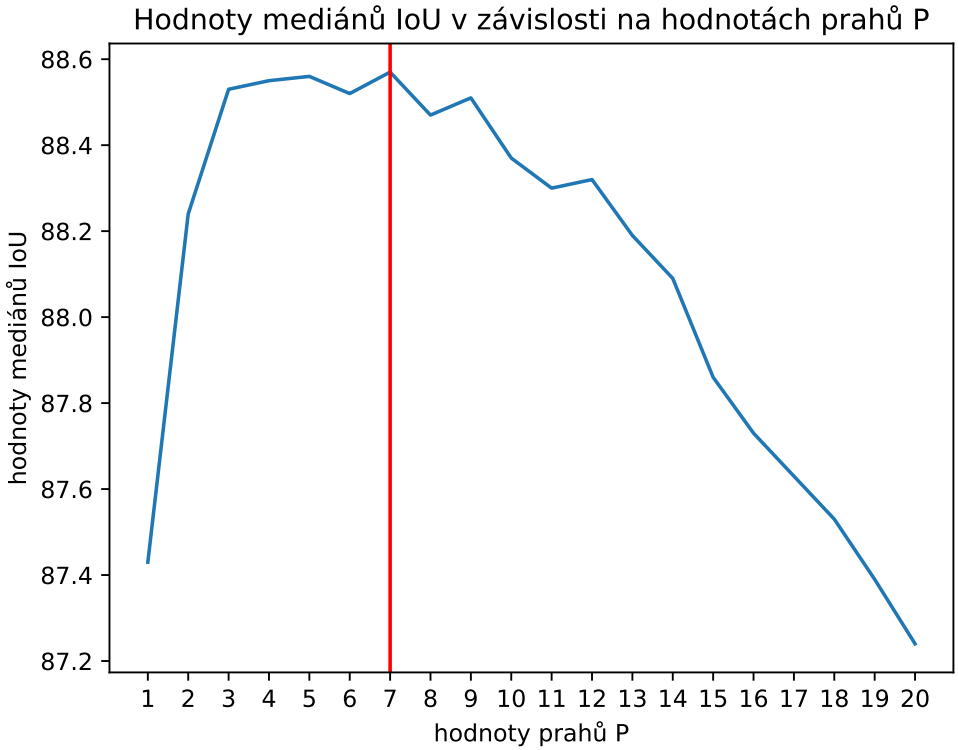


Obrázek 3.16: Původní gesto



Obrázek 3.17: Optimální segmentace původního gesta na obrázku 3.16

Jediným relevantním parametrem v postupu segmentace je hodnota prahu P v sekci 3.1.4, kterým prahujeme 8-bitový obraz gesta získaný pomocí zpětné projekce histogramu. Ukazuje se, že nejvyšší hodnotu mediánu IoU celého datasetu dosáhneme nastavením hodnoty prahu $P = 7$ s hodnotou mediánu IoU $88,57\%$. Hodnoty mediánů IoU pro celý dataset byly naměřeny v závislosti na hodnotách prahů P od 1 do 20. Výsledné hodnoty ukazuje následující graf.



Obrázek 3.18: Graf hodnot mediánů v závislosti na hodnotách prahů P

3.3 Klasifikace gesta

U nalezené kontury gesta pak spočítáme její Fourierovy deskriptory, kterých uvažujeme prvních 20 a zbylé vynulujeme. Volba ponechání prvních 20 Fourierových deskriptorů se jeví jako ideální kompromis mezi výpočetní rychlostí a úspěšností klasifikace gest. U těchto prvních 20 Fourierových deskriptorů také neuvažujeme nultý deskriptor, který má význam translace kontury gesta. Takto vybrané Fourierovy deskriptory kontury zkoumaného gesta vždy porovnáváme se (stejným způsobem) vybranými Fourierovy deskriptory kontur referenčních gest ve vytvořeném datasetu. Pro dvojici gest (vždy zkoumané a nějaké referenční) napočítáme jejich Eukleidovskou vzdálenost d definovanou následovně

$$d = \sqrt{\sum_{i=1}^{20} (x_i - y_i)^2} \tag{3.1}$$

kde x_i je i-tý Fourierův deskriptor kontury referenčního gesta a y_i je i-tý Fourierův deskriptor kontury zkoumaného gesta. Poté můžeme zkoumané gesto považovat za gesto, které je v Eukleidovské vzdálenosti nejblíže ke zkoumanému. Využijeme ovšem metody klasifikace k nejbližších sousedů [4]. Tato jednoduchá metoda klasifikuje zkoumané gesto na základě většinového hlasování jeho k nejbližších sousedů (k v Eukleidovské vzdálenosti nejbližších gest). Tudíž zkoumané gesto bude přiřazeno třídě, která je v množině jeho k nejbližších sousedů nejčastěji zastoupena. Měřením se zjistilo, že nejvyšší průměrné úspěšnosti klasifikace dosáhneme pro $k = 5$.

Dataset sloužící k testování úspěšnosti klasifikace je odlišný od datasetu, který sloužil k testování segmentace. Obsahuje celkem 11963 párů výřezu obličeje a výřezu gesta (snímky byly pořízeny ve stejnou chvíli) od jednoho subjektu a na 3 rozdílných pozadích, kde ke každé třídě přísluší přibližně 1500 párů těchto výřezů (přibližně 500 z každého pozadí). Dataset byl nafocen na kameru z notebooku s rozlišením 1920×1080 (1080p). Výřezy obličeje a gesta jsou znázorněné na následujících obrázcích.

(a) Výřez obličeje

(b) Výřez gesta

Obrázek 3.19: Ukázka výřezů obličeje a gesta z datasetu

3.3.1 Výsledky klasifikace

Klasifikace byla testována rozdělením datasetu v poměru 4:1 na dvě části – referenční a testovací. Množina referenčních obrázků gest obsahuje 9572 vzorků, kdežto množina testovacích obrázků gest obsahuje 2391 vzorků. Každé gesto ze skupiny testovacích gest bylo klasifikováno pomocí představeného postupu a byla sledována úspěšnost klasifikace. Volbou ponechání prvních 20 Fourierových deskriptorů a zvolením $k = 5$ v klasifikační metodě k nejbližších sousedů dosáhneme již zmíněného kompromisu mezi výpočetním časem a úspěšností klasifikace. Při této volbě úspěšnost klasifikace testovací skupiny gest dosahuje v průměru $83,67\%$. Procentuální úspěšnosti klasifikace v závislosti na volbě k v metodě k nejbližších sousedů a počtu požadovaných Fourierových deskriptorů ukazuje následující tabulka. Z tabulky je zřejmé, že nejvyšší průměrné úspěšnosti klasifikace dosáhneme volbou $k = 5$. Naopak, volba ponechání prvních 20 Fourierových deskriptorů je pak právě zmíněným kompromisem mezi úspěšností klasifikace a výpočetním časem.

počet deskriptorů	počet nejbližších sousedů						
	1	3	5	7	9	11	průměr
5	70,30	73,75	76,52	76,70	76,58	76,06	74,98
10	81,00	82,06	82,77	83,03	83,01	82,42	82,38
15	82,26	82,89	83,34	83,71	83,28	82,66	83,02
20	82,29	83,01	83,67	83,71	83,28	83,14	83,18
25	82,71	83,43	83,75	83,53	83,54	83,38	83,39
30	82,66	83,59	84,04	83,66	83,50	83,52	83,49
35	82,78	83,59	83,92	83,66	83,49	83,51	83,49
40	82,65	83,72	84,12	83,74	83,41	83,71	83,55
45	82,78	83,76	84,09	83,70	83,45	83,68	83,57
50	82,82	83,81	84,00	83,70	83,53	83,64	83,58
průměr	81,22	82,36	83,02	82,91	82,70	82,57	

Tabulka 3.2: Závislost úspěšnosti klasifikace na počtu zachovaných deskriptorů gesta a počtu nejbližších sousedů

Kapitola 4

Klasifikace gesta s využitím neuronových sítí

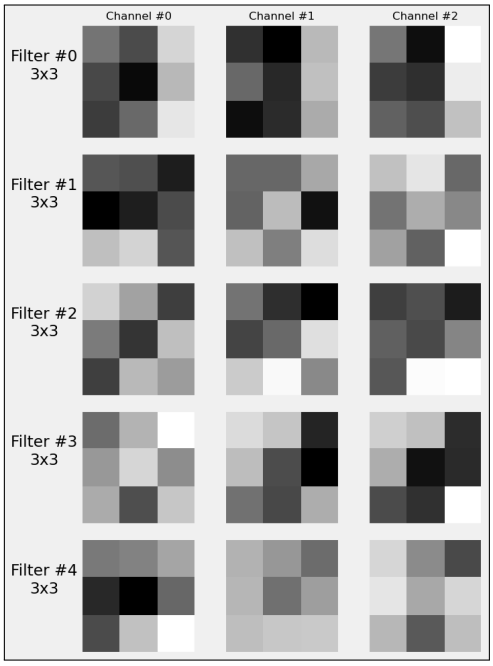
Jak již bylo zmíněno, provedeme srovnání námi navržené metody klasifikace s metodou klasifikakace, která využívá výhradně neuronové sítě. Neuronové sítě nejsou předmětem této bakalářské práce a slouží pouze jako porovnání s námi navrženou metodou, proto zmíníme pouze základní parametry modelu bez bližších podrobností. Vycházíme z knihy "Deep Learning with PyTorch Step-by-Step: A Beginner's Guide"[7]. Součástí knihy je také kód napsaný v jazyce Python, který využívá knihovnu PyTorch [16]. Ve zmíněné knize je šestá kapitola věnována rozpoznávání gest, která figurují ve hře „kámen, nůžky, papír". Předpřipravený model neuronových sítí z této kapitoly využijeme a napasujeme na dataset ze sekce 3.3.1, který jsme využívali pro testování úspěšnosti klasifikace námi navrženou metodou.

Připomeneme, že tento dataset obsahuje celkem 11963 párů obrázku gesta a k němu příslušný obrázek obličeje. Obrázky obličeje v tomto případě neuvažujeme a k naučení modelu neuronové sítě využijeme pouze obrázky gest. Obrázky z datasetu byly pořízeny na 3 rozdílných pozadích od jednoho subjektu. Každá třída obsahuje přibližně 1500 obrázků gesta. Množiny validačních a trénovacích (referenčních) obrázků volíme stejně jako při testování úspěšnosti klasifikace námi navrženou metodou. Validační množina obrázků gest pro neuronovou síť odpovídá testovací množině ze sekce 3.3.1.

Prvně jsou obrázky z datasetu zmenšeny na rozlišení 28x28, převedeny do RGB prostoru barev a dále převedeny do tenzorů, které poskytuje knihovna PyTorch. Dále pro každý obrázek z trénovací množiny napočítáme střední hodnotu a směrodatnou odchylku pro každý barevný kanál. Dále určíme průměr těchto veličin pro každý kanál ze všech obrázků datasetu. Tím získáme střední hodnotu a směrodatnou odchylku každého z kanálů RGB prostoru barev v trénovacím datasetu. Těmito hodnotami normalizujeme celý dataset.

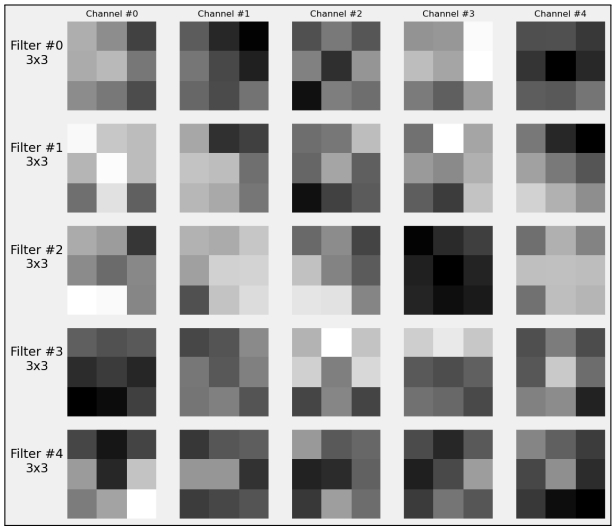
Aby měl náš model vyšší úspěšnost klasifikace na datech, které nikdy předtím nezpracovával, využijeme Dropout [20]. Dropout je důležitou součástí modelů hlubokého učení. Používá se jako regulátor, to znamená, že se snaží zabránit přetrénování sítě (overfitting) tím, že nutí model najít více než jednu cestu k dosažení cíle. Dropout během trénování vynechává určitý počet výstupů vrstvy. To má za následek, že vrstva vypadá jako vrstva s jiným počtem uzlů a konektivitou než předchozí vrstva a je s ní tak zacházeno. Počet vynechaných výstupů vrstvy závisí na parametru pravděpodobnosti $p \in [0, 1]$.

Celý model je pak složen z celkem pěti vrstev. Obsahuje dvě konvoluční vrstvy, dvě lineární a Dropout. První z konvolučních vrstev má parametry 5x3x3x3. To znamená, že obsahuje 5 konvolučních filtrů na každý ze tří kanálů o rozměrech 3x3. Filtry první vrstvy vypadají následovně.



Obrázek 4.1: Ukázka filtrů první konvoluční vrstvy

Druhá konvoluční vrstva má rozměry 5x5x3x3 a její filtry vypadají následovně.



Obrázek 4.2: Ukázka filtrů druhé konvoluční vrstvy

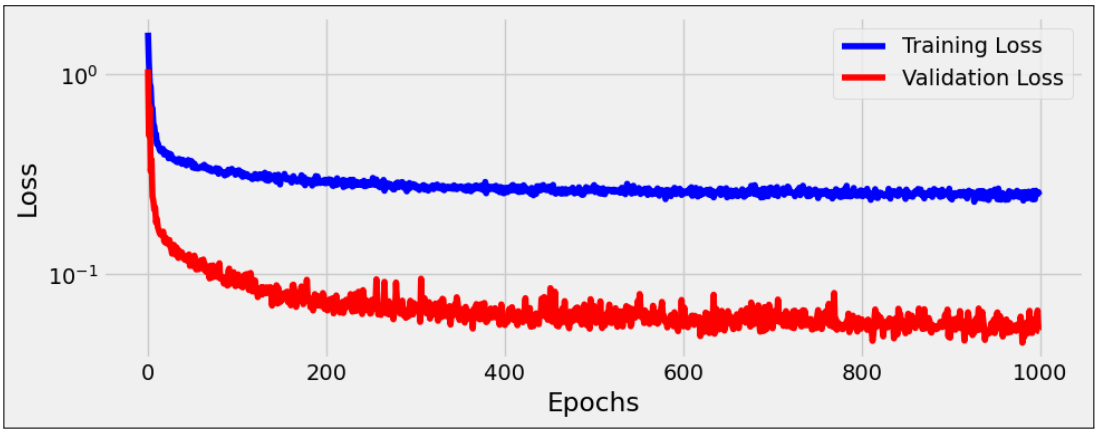
Třetí vrstvu pak tvoří skrytá (hidden) lineární vrstva s rozměry 50x125 a čtvrtou pak výstupní (output) lineární vrstva s rozměry 8x50. Součástí modelu je také již zmíněný Dropout s parametrem $p = 0,5$. Dropout vrstvy se nachází vždy před každou lineární vrstvou.

Jakožto optimalizátor zvolíme SGD (z anglického Stochastic Gradient Descent) s modifikací „Nesterov Momentum“ [21]. Parametry optimalizátoru zvolíme následovně.

$$\text{learning rate} = 0,05$$
$$\text{momentum} = 0,9$$

Jako scheduler (plánovač) zvolíme „CyclicLR“ implementovaný v knihovně PyTorch a založený na článku „Cyclical Learning Rates for Training Neural Networks“ [19].

Model se natrénoval na zařízení Apple MacBook M1 Pro (8-core CPU, 14-core GPU) po 1000 epochách s úspěšností klasifikace na trénovací množině obrázků gest 99,35% a úspěšností klasifikace na validační množině 99,03%. Graf Training Loss a Validation Loss vypadá následovně.



Obrázek 4.3: Training Loss a Validation Loss v závislosti na počtu epoch

Závěr

Cílem této bakalářské práce bylo rozpoznat gesto v obraze, který poskytuje kamera dronu, navrhnout metodu klasifikace gest a na základě klasifikovaného gesta dále ovládat dron.

V první kapitole jsme se zaměřili na krátké shrnutí konstrukce, ovládání a pohybu dronu.

Druhá kapitola je věnována přehledu metod, které dále využíváme v praktické části. Zaměřili jsme se na prahování obrazu iterativním algoritmem a následným vylepšením hledání optimální hodnoty prahu v podobě Otsuovy metody. Druhá kapitola se mimo jiné věnuje metodě zpětné projekci histogramu, která umožňuje namapovat pixely gesta tak, aby svou intenzitou odpovídaly pixelům kůže na obličeji. Následuje pak přehled metod sledování kontur a pokročilejší Suzukiho algoritmus, na kterém je založena funkce findContours() z knihovny OpenCV. Poslední část kapitoly je věnována teoretickému pozadí Fourierových deskriptorů.

Předmětem třetí kapitoly je samotné praktické provedení segmentace a klasifikace gesta s využitím metod z druhé kapitoly. Jedinou výjimkou je knihovna MediaPipe, kterou využíváme pro detekci obličeje v obrazu. Úspěšnost segmentace vyhodnocujeme pomocí metody IoU a porovnáváme s manuálně anotovaným datasetem gest. Nejvyšší hodnotu mediánu IoU jsme naměřili 88,57%. Klasifikace gest námi navrženou metodou s využitím Fourierových deskriptorů hranice dosahuje úspěšnosti 83,67%. Volby parametrů pro metody segmentace a klasifikace gesta byly testovány na dvou rozdílných datasetech gest.

Čtvrtá kapitola se pak zabývá porovnáním úspěšnosti klasifikace námi navržené metody s metodou klasifikace pomocí neuronových sítí. Výsledný model neuronové sítě dosahuje úspěšnosti klasifikace 99,35% na trénovací množině obrázků gest a 99,03% na validační množině obrázků gest. Klasifikace gest s využítím modelu neuronových sítí je tedy výrazně úspěšnější než klasifikace pomocí námi navržené metody.

V budoucnu by se chtěl autor této práce věnovat právě využití neuronových sítí v oblasti zpracování obrazu a počítačového vidění.

Odkaz na oba datasety je k dispozici zde:
`https://kaggle.com/datasets/adamnovozmsk/hands`
Odkaz na kódy využité v bakalářské práci:
`https://github.com/michmanprusek/Gesture-controlled-drone`

Literatura

[1] S. Beucher. The watershed transformation applied to image segmentation. *Scanning Microscopy*, 1992(6), 1992.

[2] A. Boyat and B. Joshi. A review paper: Noise models in digital image processing. *Signal & Image Processing : An International Journal*, 6, 05 2015.

[3] A. Buades, B. Coll, and J. M. Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65 vol. 2, 2005.

[4] P. Cunningham and S. J. Delany. k-nearest neighbour classifiers: 2nd edition (with python examples). *CoRR*, abs/2004.04523, 2020.

[5] D. F. Escoté. Knihovna djitellopy. Dostupné na `https://github.com/damiafuentes/DJITelloPy/tree/master`, version 1.6.0.

[6] E. Gedraite and M. Hadad. Investigation on the effect of a gaussian blur in image filtering and segmentation. pages 393–396, 01 2011.

[7] D. V. Godoy. *Deep Learning with PyTorch Step-by-Step: A Beginner's Guide*. Independently published, 1 2022.

[8] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Pearson, 2018.

[9] M. Hansard, S. Lee, O. Choi, and R. Horaud. *Time of Flight Cameras: Principles, Methods, and Applications*. 10 2012.

[10] W. Jing, Y. Ze, and Z. Xuechao. Adaptive skin color detection based on human face under complex background. *MATEC Web of Conferences*, 22:01006, 07 2015.

[11] J. Kovac, P. Peer, and F. Solina. Human skin color clustering for face detection. In *The IEEE Region 8 EUROCON 2003. Computer as a Tool.*, volume 2, pages 144–148 vol.2, 2003.

[12] T. Kumar and M. I. Hasan. Physics of quadcopter and itssurveillance application: A review, 2020.

[13] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C. Chang, M. G. Yong, J. Lee, W. Chang, W. Hua, M. Georg, and M. Grundmann. Mediapipe: A framework for building perception pipelines, 2019.

[14] C. Ma and H. Shih. Human skin segmentation using fully convolutional neural networks. In *2018 IEEE 7th Global Conference on Consumer Electronics (GCCE)*, pages 168–170, 2018.

[15] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[17] J. G. Proakis and D. G. Manolakis. *Digital Signal Processing (3rd Ed.): Principles, Algorithms, and Applications*. Prentice-Hall, Inc., USA, 1996.

[18] A. Raj, S. Gupta, and N. K. Verma. Face detection and recognition based on skin segmentation and cnn. In *2016 11th International Conference on Industrial and Information Systems (ICIIS)*, pages 54–59, 2016.

[19] Leslie N. Smith. Cyclical learning rates for training neural networks, 2017.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[21] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[22] S. Suzuki and A. Keiichi. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

[23] M. J. Swain and D. H. Ballard. Indexing via color histograms. In *[1990] Proceedings Third International Conference on Computer Vision*, pages 390–393, 1990.

[24] OpenCV team. Oficiální webová stránka knihovny opencv, Jul 2023.

[25] Ryze Tech. Oficiální webová stránka modelu dronu tello. Dostupné na `https://www.ryzerobotics.com/tello`.

[26] G. Tofighi, S. A. Monadjemi, and N. Ghasem-Aghaee. Rapid hand posture recognition using adaptive histogram template of skin and hand edge contour. In *2010 6th Iranian Conference on Machine Vision and Image Processing*, pages 1–5, 2010.

[27] V. P. Yadav, G. Singh, Md. I. Anwar, and A. Khosla. Periodic noise removal using local thresholding. In *2016 Conference on Advances in Signal Processing (CASP)*, pages 114–117, 2016.

Seznam obrázků

Seznam tabulek