VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

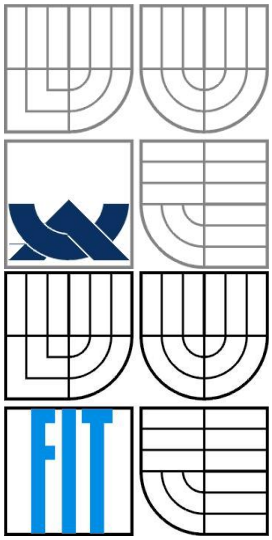
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

OVLÁDÁNÍ POZEMNÍHO ROBOTA GESTY

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE MICHAL CHLUD
AUTHOR

BRNO 2014
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

OVLÁDÁNÍ POZEMNÍHO ROBOTA GESTY

NÁZEV BAKALÁŘSKÉ PRÁCE ANGLICKY

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — MICHAL CHLUD
AUTHOR

VEDOUCÍ PRÁCE — Ing. Vítězslav Beran, Ph.D.
SUPERVISOR

BRNO 2014

Abstrakt

Bakalářská práce popisuje hlavní metody, které slouží k úspěšnému rozpoznání gest ruky pomocí běžné RGB kamery. Zabývá se jednotlivými kroky, které vedou ke klasifikaci ruky ve videosekvenci jako je segmentace vstupního obrazu na základě modelu barvy kůže. Dále popisuje metody sledování a analýzy gest. Cílem práce je vytvoření aplikace na ovládání pozemního robota gesty, která využívá těchto znalostí. Práce navrhuje řešení a poté popisuje jednotlivé kroky implementace.

Klíčová slova

Počítačové vidění, rozpoznání gest, gesta rukou, ovládání robota, model barvy kůže, OpenCV, ROS

Abstract

This bachelor’s work is focused on basic methods leading to successful gesture recognition in video sequence using standard RGB camera. Key steps required for hand classification are described, mainly segmentation of input video based on skin color model. Next there are discussed methods for object tracking and gesture analysis. Main goal of this work is to demonstrate this knowledge by creating application which controls robot’s movement by hand gesture analysis. Solution is proposed and individual implementation steps are described.

Keywords

Computer vision, gesture recognition, hand gesture, robot control, skin color model, OpenCV, ROS

Citace

Chlud Michal: Ovládání pozemního robota gesty. Brno, 2014, bakalářská práce, FIT VUT v Brně.

Ovládání pozemního robota gesty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Vítězslava Berana Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Michal Chlud
21.05.2014

Poděkování

Tímto bych rád poděkoval svému vedoucímu Ing. Vítězslavu Beranovi Ph.D. za cenné rady, trpělivost a vedení při tvorbě této práce.

Obsah

1 Úvod

V dnešní době získávají na popularitě inovativní řešení v oblasti ovládání počítačů a robotů. To umožňuje především pokrok v oblasti technologií. Zvyšování výkonu běžných počítačů a mobilních zařízení. Tento pokrok umožnuje implementaci náročnějších metod. Počítače tvoří velkou součást každodenního života. Lze předpokládat, že tento trend bude pokračovat a interakce s počítačem bude stále častější. I vlivem tohoto převládá silná tendence objevovat nové a intuitivní způsoby jakými může člověk komunikovat s počítačem, protože komunikace pomocí gest může nabídnout spoustu výhod oproti klasickým metodám. Jednou z motivací je právě snaha o větší podobnost s mezilidskou interakcí. Mezi největší klady patří bezkontaktnost, jednoduchost a přirozenost. Ovládání pomocí gest může obohatit velkou řadu oblastí.

Jako příklad ze skupiny těch komerčně úspěšnějších a známějších lze uvést způsob ovládání her gesty, který byl zpopularizován pomocí zařízení Microsoft Kinect. Ale jde také o oblast zdravotnictví, kde je jednou z výhod právě bezkontaktnost, která slouží k větší hygieně. Aplikace mohou pomáhat například při rehabilitaci nebo mohou chirurgům nabízet při operaci důležité informace atp. Ovládání gesty také slibuje velké možnosti v pomoci handicapovaným. Další oblastí je ovládání strojů, kde lze využít přirozenosti a podobnosti některých gest s běžnými gesty lidí. Uživatel může intuitivně, pomocí pohybu ruky, ovládat robotické rameno nebo může gesty řídit pohyb robota. Vzdálené řízení pohybu robota by mohlo nalézt využití ve vojenských aplikacích nebo při krizových situacích jako je prohledávání sutin robotem po zemětřesení apod.

Rozpoznávání gest v obraze stojí na několika oborech, jde hlavně o počítačové vidění a zpracování obrazu. Při zpracování gest rukou je klíčový způsob detekce ruky, její sledování a vyhodnocení gest. Přestože taková řešení nejsou v současnosti úplně běžně rozšířená, lze se domnívat, že se jich bude objevovat stále více a možná budou jednoho dne naprosto běžnou součástí našeho života.

Cílem této bakalářské práce je vytvoření aplikace, která slouží k ovládání pozemního robota pomocí gest ruky. První kapitola se zabývá metodami, které lze pro tento účel využít. Druhá kapitola se snaží nabídnout vlastní řešení tohoto problému a třetí kapitola popisuje implementaci řešení.

2 Teorie

Tato kapitola poskytuje stručný náhled do problematiky týkající se oblasti zpracování gest. Také popisuje základní metody v jednotlivých krocích, které se v této oblasti používají k dosažení požadované funkčnosti.

2.1 Gesta jako uživatelské rozhraní

Neverbální gesta jsou základním kamenem mezilidské komunikace. Protože jsou pro člověka naprosto přirozená, je jedním z velkých cílů oborů zabývajících se počítačovým viděním přenést intuitivnost a jednoduchost komunikace mezi lidmi do oblasti interakce s počítačem [7][5]. Toto úsilí je podporováno stále se zvyšujícím výkonem výpočetní techniky a rozvojem technologií. Gesto může být z hlediska lidské komunikace velice široký pojem. Může jít o výraz tváře, postoj těla, pohyb očí, ale nejvýraznější jsou gesta rukou.

V některých ohledech nemohou gesta nahradit klasické ovládací prvky jako myš nebo klávesnici. Ale v mnoha situacích by mohlo být využití gest daleko efektivnějším a intuitivnějším způsobem ovládání. Mnoho oborů by mohlo těžit z tohoto inovativního přístupu jako třeba medicínské aplikace, zábavní průmysl, pomoc postiženým a právě komunikace člověka s robotem [7]. Možnost řídit robota gesty by mohla otevřít úplně nové možnosti. Ať už jde přímo o řízení pohybu samotného robota nebo o manipulaci s rameny, robotickou rukou atp. Mezi největší výhody použití gest patří:

- Bezkontaktní použití. Výhodou je, že nemusí dojít k fyzickému kontaktu člověka s ovládacím prvkem.
- Kvalitně navržený interface může poskytnout velice intuitivní a pohodlný způsob ovládání komplexních aplikací, který není potřeba se učit. Což může pomoct například starým lidem nebo takovým, kteří s ovládáním nemají zkušenosti.
- Usnadnění přístupu handicapovaným. Může pomoct lidem, kterým fyzická omezení brání ve využití klasických ovládacích prvků.

Ke zpracování gest se používá nejčastěji obyčejná RGB kamera, která je dnes dostupná téměř na všech počítačích. Dalším pokročilejším zdrojem dat je hloubkový senzor nebo stereo vize. Mezi hlavní požadavky kladeny na ovládání pomocí gest patří [7].

- **Cena.** Kvalita nahrávacích zařízení nesmí omezovat správné fungování aplikace. Také hardware, na kterém aplikace „běží“, musí dostačovat. Aplikace zpracovávající gesta mohou být často dost náročné.
- **Odezva**. Tento bod úzce souvisí s předchozím. Aplikace musí mít dostatečně rychlou reakční dobu, jinak je používání nepraktické.
- **Přesnost**. Přesnost aplikace musí odpovídat oblasti použití. Například na aplikaci, která bude pomáhat chirurgům, budou kladeny větší nároky, než na herní aplikace.
- **Intuitivnost**. Gesta by měla být dobře specifikována a způsoby jejich použití by se neměli překrývat, aby nedocházelo k matení uživatele.

- **Pohodlnost**. Gesta by neměla od uživatele vyžadovat úkony, u kterých hrozí únava svalů. Musí být navržena s ohledem na očekávanou souvislou aktivní dobu použití aplikace.

2.1.1 Dělení gest z pohledu jejich zpracování

Forma gesta může být rozdělena na statická a dynamická [17]. U statických gest nehraje při klasifikaci roli trajektorie nebo pohyb. Většinou jde o kombinaci aktuální konfigurace prstů a celkové orientace ruky. Může jít o prstovou abecedu nebo (v jednodušší formě) rozpoznání tohoto kolik uživatel ukazuje prstů nebo zda má ruku v pěst apod. Naopak u dynamických gest pracuje klasifikace s informací o změně polohy sledované ruky v čase. Často není moc brán zřetel na aktuální konfiguraci ruky. Ta může sloužit jako signál k započnutí dynamického gesta. Hlavním vodítkem je sledování polohy objektu napříč videosekvencí.

Základní rozdělení v přístupu k rozpoznávání gest rukou je na metody invazivní a neinvazivní [17]. Metody invazivní využívají dalších nástrojů nebo pomůcek, které napomáhají k identifikaci ruky ve videosekvenci, protože právě detekce ruky může být jedna z nejproblematičtějších částí celého procesu. Nejčastěji jde o barevné rukavice nebo rukavice s akcelerometry apod. za účelem věrnějšího zachyceni dynamických dějů. Přístupy neinvazivní spoléhají výhradně na detekci založené na specifických vlastnostech ruky nebo prostředí (barva kůže, detekce pohybu apod.).

Pro rozpoznání ruky ve videosekvenci se používají následující přístupy [7]:

- **Barva**. Jedna z nejrozšířenějších metod. Využívá modelů barvy kůže k segmentaci obrazu. Tento přístup nemusí správně fungovat při proměnlivých světelných podmínek. Modely barvy kůže jsou často učeny pomocí rozsáhlé tréninkové sady. Pro zvýšení spolehlivosti mohou být využity barevné rukavice.
- **Detekce pohybu**. V nejjednodušší formě jde o diferenci dvou po sobě jdoucích snímků. Hodí se pouze pro prostředí se statickým pozadím, ale jde o hardwarově nenáročnou metodu. Lze sem zařadit i metody, které využívají substrakce pozadí.
- **Hloubková mapa**. Využívá segmentaci na základě hloubkové mapy. Podmínkou je senzor, který to umožnuje. Jde o stabilní metodu, protože není závislá na osvětlení. Často je kombinovaná s modelem barvy kůže.
- **Tvar**. Snaží se klasifikovat objekty na základě tvaru. Pokud není kombinována s jinou metodou tak lze použit zpravidla pouze v prostředí s jednoduchým pozadím. Většinou využívají detekci hran.
- **Jiné.** Lze sem zařadit Haaruv klasifikátor, který je velmi populární při detekci obličejů. Při detekci ruky tento klasifikátor tak úspěšný není.

2.2 Počítačové vidění

Počítačové vidění [12] je obor, který se snaží napodobit funkci biologického oka. Schopnost počítače „vidět“ je postavená na několika sousedních oborech jako je zpracování obrazu, rozpoznávání atd. Tento obor se začal rozvíjet zhruba v 60. letech 20. století. Vývoj hardware velkou měrou přispívá k rozvoji tohoto oboru, protože je potřeba zpracovat velké množství dat.

Zajímavé jsou aplikace v robotice. Vidění umožnuje robotům autonomní navigaci, protože jsou schopni rozpoznat prostředí, ve kterém se nacházejí. Mapovat okolí, určit svoji polohu a tyto informace použít v efektivním pohybu prostředím. Vidění jim umožnuje vyhýbat se překážkám. Také mohou rozpoznávat předměty, se kterými mohou následně interagovat. Tyto schopnosti mají velkou škálu

využití třeba ve vojenském průmyslu nebo při záchranných akcích, kde mohou operovat v prostředí, které by bylo člověku nebezpečné.

Dalším zajímavou oblastí použití je rozšíření možností interakce člověka s počítačem[1][2], anglicky se tento obor nazývá Human-Computer Interaction. Používá se běžná zkratka HCI. Počítačové vidění poskytuje nové možností interakce. Velkou inspirací je subkultura sci-fi, kde je často k vidění jak uživatel efektivně komunikuje s počítačem pomocí různých gest. Jako známý současný příklad lze uvést způsob ovládání her, které umožnil Microsoft Kinect[3].

Bohužel reakce aplikací často nejsou ideální, protože zpracování dat je složité a také není jednoduché navrhnout ovládání, tak aby bylo jednoznačné, přesné apod. Úkolem je zvyšovat přesnost těchto metod a snižovat jejich nároky na hardware a prostředí.

Počítačové vidění je úzce spojeno se zpracováním obrazu. Úkol zpracovaní obrazu je obraz vylepšit nebo transformovat. Počítačové vidění následně obraz analyzuje a na jeho základě činí nějaké rozhodnuti. Z pohledu této práce je tedy počítačové viděni nadřazené zpracovaní obrazu. Dále následuje stručný přehled použitých metod.

2.2.1 Barevné prostory

Obraz může být zakódován pomocí různých barevných prostorů. Každý barvu reprezentuje jinak. Zpravidla jde o vektor o třech dimenzích. Barevné prostory mají své výhody a nevýhody. Následuje stručný přehled [12][16].

RGB

RGB je základní barevný prostor. Barvy jsou kódovány pomocí tří základních barev – červené (R, Red), zelené (G, Green) a modré (B, Blue). Parametry nabývají hodnoty z intervalu <0,1>. Pokud jsou kódovány v jednom bytu, tak nabývají celočíselných hodnot z rozsahu 0 až 255.

Normalizované RGB

Normalizované RGB má za cíl linearizovat oblasti s jiným nasvícením, ale stejnou barvou. Jinými slovy by měla tato transformace odstranit vliv nerovnoměrného nasvícení scény. Převod je následující:

$$r = \frac{R}{R + G + B}$$

$$g = \frac{G}{R + G + B}$$

$$b = \frac{B}{R + G + B}$$

$$r + g + b = 1$$

[1] Ming-Hsuan Yang and Narendra Ahuja. 2012. *Face Detection and Gesture Recognition for Human-Computer Interaction*. Springer Publishing Company, Incorporated.

[2] Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale. 1997. *Human-Computer Interaction*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[3] http://www.microsoft.com/en-us/kinectforwindows/

Další výhodou tohoto prostoru je, že barva může být reprezentována pouze pomocí dvou dimenzí, protože platí, že součet všech tří proměnných je jedna. K tomuto účelu se většinou používají hodnoty *r* a g.

HSV

Prostor HSV definuje barvu jako trojici složek. První složkou je *hue* (barevný tón), druhou *saturation* (sytost) a třetí *value* (jasová hodnota). Barevný tón určuje základní barvu. Nabývá hodnot 0 až 360 a je tvořen barevným kruhem, kde 0 stupňů odpovídá červené, dále následuje přechod do žluté, zelené, modré, fialové a pak opět červené. Hodnota sytosti může nabývat hodnot z intervalu <0,1> a určuje příměs jiných barev. Hodnota jasu určuje příměs bílého světla. OpenCV používá následující transformací:

$$V = \max(R, G, B)$$

$$S = \begin{cases} \dfrac{V - \min(R,G,B)}{V} & pokud\ V \neq 0 \\ 0 & jinak \end{cases}$$

$$H = \begin{cases} \dfrac{60(G-B)}{V - \min(R,G,B)} & pokud\ V = R \\ 120 + \dfrac{60(B-R)}{V - \min(R,G,B)} & pokud\ V = G \\ 240 + \dfrac{60(R-G)}{V - \min(R,G,B)} & pokud\ V = B \end{cases}$$

YC_BC_R

Třísložkový barevný model YC_BC_R reprezentuje barvu pomocí parametru Y (luminance), Cb (modrý chrominanční komponent a Cr (červený chrominanční komponent). Výhoda tohoto modelu je, že odděluje informaci o osvětlení od informací o barvě.

$$Y = 0.299R + 0.587G + 0.114B$$
$$Cb = (B - Y) \cdot 0.564$$
$$Cr = (R - Y) \cdot 0.713$$

Šedotónový obraz

Základní transformace. Barevný obraz je převeden do šedotónového, který reprezentuje úrovně jasu v obraze. Tento obraz je v podstatě to samé jako složka Value v HSV a Y (luminance) z YC_BC_R. Šedotonový obraz se využívá například při detekci hran.

$$Y = 0.2126R + 0.7152G + 0.0722B$$

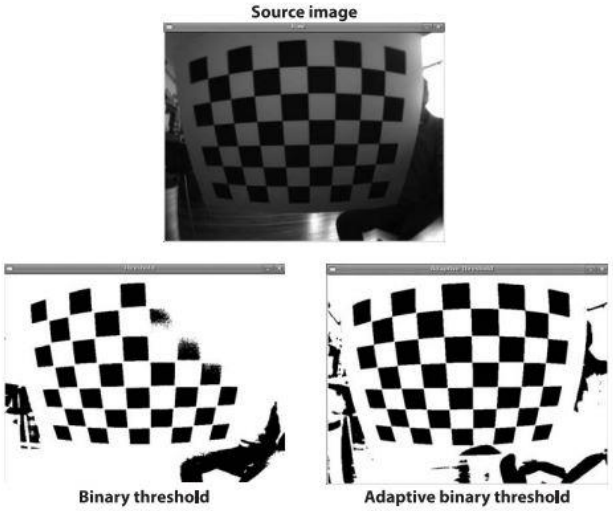
2.2.2 Prahování

Jednou ze základních metod segmentace obrazu je prahování na základě nějaké jasové konstanty [12]. Prahování je většinou prováděno na šedotónovém obrazu. Obraz je pak na základě tohoto prahu

globálně segmentován. Prahování je nejstarší a nejjednodušší segmentační metodou. Úspěšnost segmentace je závislá na určení správné hodnoty prahu. Ten může být určen buď manuálně, nebo automaticky. Práh by měl být určen tak, aby rozdělil histogram obrazu tak, že hodnoty menší než práh odpovídají pozadí (bude jim přiřazena 0) a hodnoty větší než práh odpovídají popředí (těm je přiřazena log. 1). Tímto způsobem je vytvořena maska, která reprezentuje oblasti zájmu. Matematický zápis:

$$dst(x,y) = \begin{cases} 1\ pokud\ src(x,y) > práh \\ 0\ jinak \end{cases}$$

$dst(x,y)$ je logická hodnota výstupní masky. $src(x,y)$ značí jasovou hodnotu vstupního šedotónového snímku.



*Obr. 1: **Příklad prahování**. Vstupní obrázek (nahoře). Jednoduché prahování (vlevo dole). Adaptivní prahování (vpravo dole). Obr. z [2].*

Automatické určování globálního prahu většinou pracuje na základě histogramu. Základním předpokladem je, že objekty zájmu budou od pozadí jasově odlišeny. Pak by měl histogram obsahovat dva vrcholy (bimodální histogram). Úkolem automatické metody je najít co nejideálnější práh. Ten by měl splňovat podmínku minimální segmentační chyby.

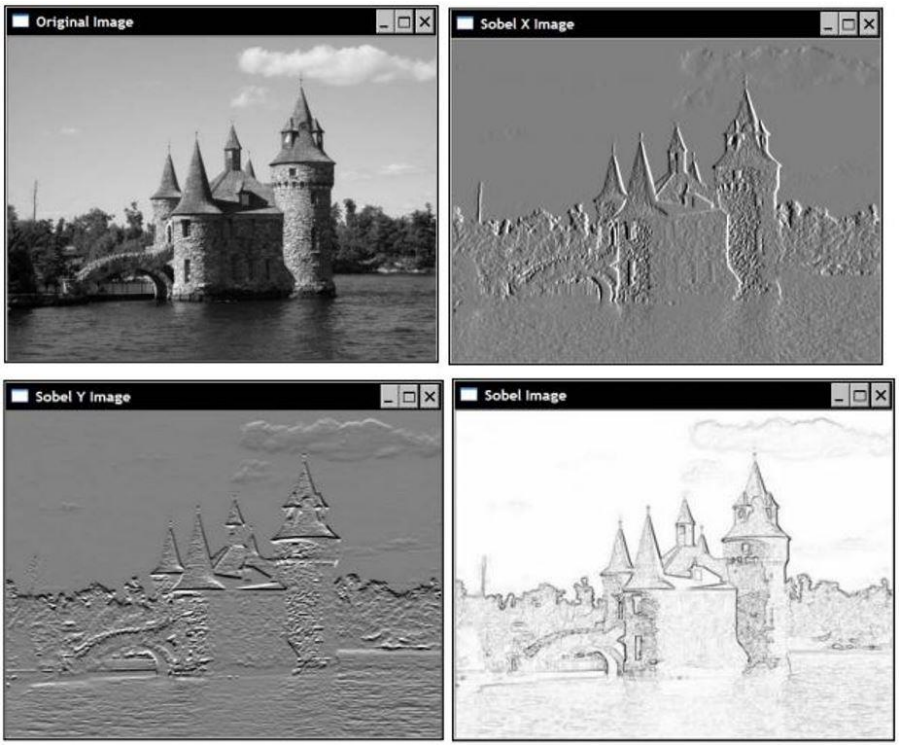
Jednou z možnosti jak určit globální práh automaticky je **Otsuova metoda [2].** Ten právě předpokládá bimodální histogram. Ideální práh volí takový, který minimalizuje vnitřní rozptyl dvou skupin pixelů (pozadí a popředí).

Často však nelze určit práh globálně, protože obraz není rovnoměrně nasvícen apod. Proto byly vytvořeny metody, které provádějí segmentaci lokálně. Mezi automatické metody tohoto typu patří **adaptivní prahování** [2]. V této metodě je obraz prahován po malých oblastech. Práh je vypočítán pro každou oblast zvlášť. Například to může být průměrná hodnota jasu z vybrané oblasti.

2.2.3 Detekce hran

Další užitečnou operací v rámci zpracování obrazu je detekce hran [12]. Tato informace může být využita při rozdělování obrazu na oblasti nebo při zjišťování hranic objektů. Princip je založen na

předpokladu, že hrana odpovídá změně intenzity jasu sousedních pixelů. Hrana je vektorová veličina a je určena velikostí a směrem.



*Obr. 2: **Detekce hran**. Z levého horního rohu: Původní obrázek. Po aplikaci jádra pro y-osu. Po aplikaci jádra pro x-osu. Obraz po složení obou předchozích a prahování. (zdroj:[14])*

Detekce hran v praxi probíhá pomocí konvoluce snímku s vhodným jádrem. Často se používá Sobelův operátor, který aproximuje první parciální derivace.

$$G_x = \begin{vmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{vmatrix} * I \qquad G_y = \begin{vmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{vmatrix} * I$$

G_x je jádro pro detekci hran ve směru x-ové osy a G_y je jádro pro detekci ve směru osy y. Symbol I reprezentuje vstupní šedotónový obrázek. Na Obr. 2 je ukázána aplikace jednotlivých masek a složení výsledného obrazu. Následujícím způsobem lze vypočítat sílu hrany a její orientaci:

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

$$\Theta = \arctan(\frac{G_x}{G_y})$$

Symbol G odpovídá síle hrany a Θ její orientaci.

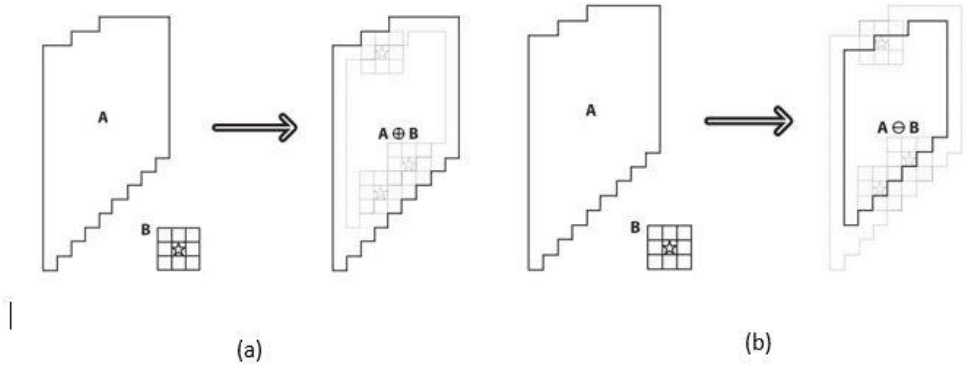
V praxi se ještě používá Laplaceův operátor, který pracuje s druhou derivaci nebo Cannyho hranový detektor. Ten používá několika kroků k dosažení co nejoptimálnějšího výsledku.

2.2.4 Morfologické operace

Matematická morfologie poskytuje nástroje pro předzpracování obrazu a je vhodná pro analyzování tvaru objektů [12][2]. Nejčastěji se používá pro odstranění šumu, zjednodušení tvaru objektů a zdůraznění struktury objektů. Může sloužit také k detekci hran nebo k segmentaci obrazu. Vznikla v 60. letech ve Francii. Výrazně se rozšířila v 80. letech 20. století díky ucelené teorii a jednoduchosti implementace na počítačích.

Princip je takový, že probíhá relace obrazu s tzv. strukturním elementem. Matematickou morfologii lze aplikovat jak na binární obrazy, tak na obrazy s více úrovněmi jasu. Ale v kontextu této práce budou tyto operace používány pouze na binární obrazy.

Příklad strukturního elementu je na Obr. 3, kde je označen jako *B*. Často se používá čtverec o velikosti 3x3 apod.



Obr. 3: ***Demonstrace dilatace (a). Eroze (b)****. (zdroj:[2])*

Základní operace jsou eroze a dilatace. **Eroze (**Obr. 3) skládá dvě bodové množiny pomocí rozdílu vektorů.

$$X \ominus B = \{d \in E^2 : d + b \in X \; pro \; \forall \; b \in B\}$$

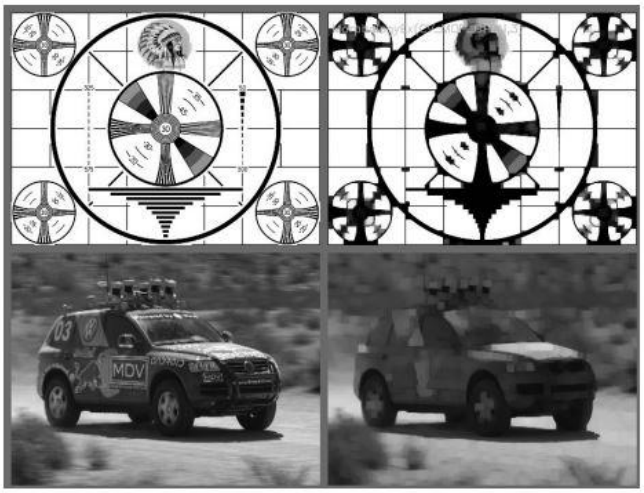
Symbol X značí bodovou množinu, na které má být provedena operace **(**na Obr. 3 je označena jako A). B je strukturní element (také bodová množina) a E^2 je 2D euklidovský prostor.

Dilatace (Obr. 3) naopak skládá body dvou množin pomocí vektorového součtu. Je to duální transformace k dilataci. (není to inverzní transformace).

$$X \oplus B = \{d \in E^2 : d = x + b, x \in X, b \in B\}$$

Protože eroze a dilatace nejsou inverzní operace, lze jejich kombinací vytvořit další významné operace. A to uzavření a otevření. Aplikace eroze a následně dilatace se nazývá **otevření**. Příklad je na Obr. 4. Definice otevření:

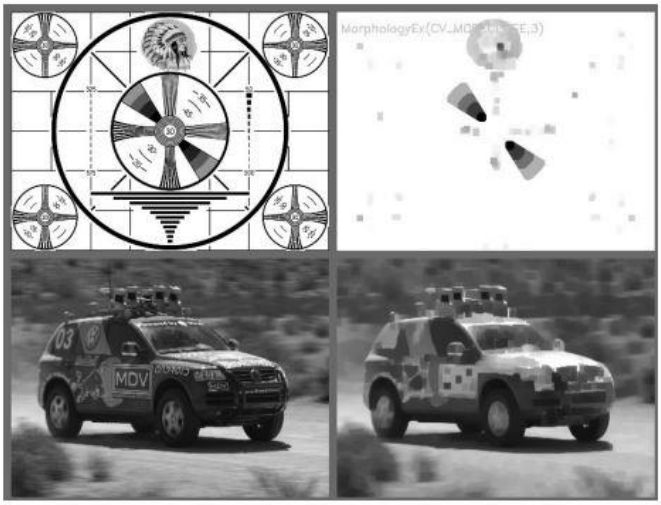
$$X \circ B = (X \ominus B) \oplus B$$



*Obr. 4: **Morfologická operace otevření** (zdroj:[2])*

Opačně aplikace eroze po dilatace se nazývá **uzavřením (**Obr. 5**):**

$$X \cdot B = (X \oplus B) \ominus B$$



*Obr. 5: **Morfologická operace uzavření** (zdroj:[2])*

2.3 Segmentace obrazu za účelem získání ruky

Základním krokem na cestě k rozpoznání gesta ruky je její detekce v obraze [11][17][15]. Tomuto tématu věnuji samotnou kapitolu, protože je to stěžejní krok na cestě k rozpoznání gest. Běžný uživatel má nejčastěji přístup k obyčejné RGB kameře (webkamery). V nedávné době se rozmohla detekce založená na zpracování hloubkových dat, především díky dostupnosti senzoru Kinect. Tento krok má zásadní vliv na výslednou kvalitu aplikace. Pokud je segmentace špatná, nemusí být ruka v obraze vůbec detekována. Nebo pokud je nekvalitní, tak některé metody, které pracují třeba s tvarem kontury ruky, nemusí poskytovat adekvátní výsledky. Tato kapitola se bude zabývat výhradně segmentací obrazu na základě modelů barvy kůže, protože na tomto principu bude fungovat i výsledné řešení (viz. v 3.2 - Specifikace výsledného řešení).

Hloubková mapa umožnuje segmentaci vstupního obrazu na základě vzdálenosti objektů od snímače. Nejjednodušší přístupy využívají toho předpokladu, že ruka je nejbližší objekt ke snímači. Microsoft také poskytuje vývojové nástroje, které dokáží analýzou hloubkové mapy vytvořit virtuální skeleton. Efektivní je využití kombinace hloubkové mapy a RGB vstupu.

Detekce využívající samotného RGB obrazu se nejčastěji snaží pracovat tak, že segmentuje obraz pomocí daného modelu barvy kůže [1][15]. Jinými slovy se dá říct, že podle barvy pixelu určí, zda pixel odpovídá barvě kůže nebo ne.

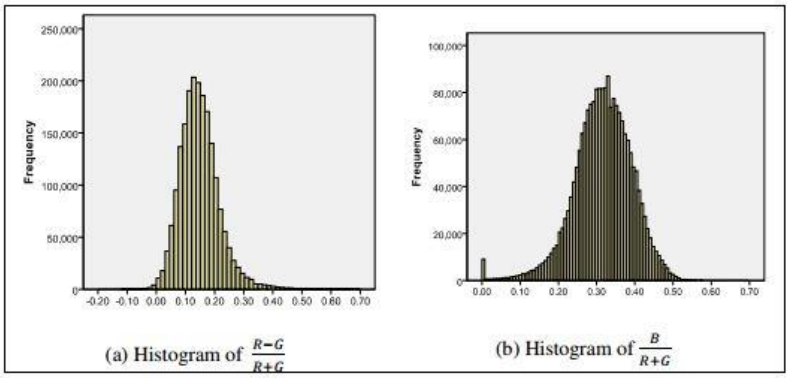
2.3.1 Jednoduché statické modely

Nejjednodušší modely barvy kůže jsou statické, které definují pouze interval a omezující pravidla v barevném prostoru, které by měly specifikovat oblast reprezentující barvu kůže. Právě Kovač a kol. [8] definovali interval a několik dalších podmínek v barevném prostoru RGB. Pravidla jsou následující:

$$Pravidlo\ 1: R > 95, G > 40\ a\ B > 20$$
$$Pravidlo\ 2: Max(R,G,B) - Min(R,G,B) > 15$$
$$Pravidlo\ 3: |R - G| > 15$$
$$Pravidlo\ 4: R > G\ a\ R > B$$

Jejich úspěšnost s tímto přístupem byla 88.39 TP (true positive) a 17.16 FP (false positive). Osman a kol. [13] použili trénovací sadu o velkosti 250 snímků obsahujících barvu kůže. Vytvořili dva histogramy (Obr. 6). Na jejich základě definovali oblast barvy kůže jako:

$$0.0 < \frac{R-G}{R+G} \leq 0.5\ a\ zároveň\ \frac{B}{R+G} \leq 0.5$$



*Obr. 6: **Histogramy reprezentující barvu kůže** (zdroj:[13])*

Úspěšnost segmentace byla v tomto případě 94.91 TP a 33.07 FP.

Problém těchto jednoduchých metod je takový, že vykazují velmi špatné výsledky při nepravidelném osvětlení. Velkou překážkou je také to, že barevný výstup kamery bývá často změněn balancováním bílé a podobnými algoritmy.

2.3.2 Parametrické modely

Parametrické modely [15], pomocí dané funkce, aproximují rozložení barvy kůže ve vybraném barevném prostoru. Výhodou těchto metod je, že nepotřebují tolik paměti. Naopak nevýhodou je, že funkce, na základě kterých je cílový model vytvářen, někdy nevykazují dostatečnou přesnost.

Gaussovo rozdělení

Základní model je založen na aproximaci distribuce pomocí jednoho Gaussova rozložení:

$$p(c|skin) = \frac{1}{2\pi|\Sigma_s|^{1/2}} \cdot e^{-\frac{1}{2}\cdot(c-\mu_s)^T{\Sigma_s}^{-1}(c-\mu_s)}$$

Kde *c* je barevný vektor pixelu. μ_s s je střední hodnota normálního rozdělení a Σ_s je kovarianční matice. Tyto parametry mohou být odvozeny z dostatečně velké trénovací sady. Výsledkem je $p(c|skin)$, neboli pravděpodobnost s jakou daný pixel odpovídá barvě kůže.

Výhodou tohoto modelu je, že jednoduchý a rychlý. Může poskytovat kvalitní výsledky [16]. Ale v komplexnějším prostředí, kde se ve scéně nacházejí objekty barvou podobné kůži, nemusí stačit.

Guassian Mixture Model

Na stejném principu pracuje metoda založená a modelování rozložení pomocí více Gaussových funkcí, tzv. Gaussian Mixture Model (GMM) [3][15]. Tato metoda je přesnější, protože aplikací více Gaussových funkcí lze věrněji aproximovat požadované rozložení. Stejně jako u předchozí metody je model naučen z dostatečně velké trénovací sady. Trénování probíhá prostřednictvím EM algoritmu (Expectation-Maximization).

$$p(c|skin) = \sum_{i=1}^{k} \pi_i \cdot p_i(c|skin)$$

$$p_i(c|skin) = \frac{1}{2\pi|\Sigma_{si}|^{1/2}} \cdot e^{-\frac{1}{2}\cdot(c-\mu_{si})^T{\Sigma_{si}}^{-1}(c-\mu_{si})}$$

Kde *i* je váha *i*-tého Gaussova rozložení a *k* je celkový počet normálních rozložení použitých v modelu. Při aplikaci je potřeba vybrat vhodný počet rozložení, které budou tvořit GMM. Pokud jich bude málo, tak nebude segmentace kvalitní. A pokud jich bude moc, tak může být proces segmentace pomalá, protože výpočet modelu zabere hodně času.

2.3.3 Bayesův klasifikátor

Mezi neparametrické metody patří modely založené na Baeysově klasifikátoru [15][1]. Pravděpodobnost, že daný pixel odpovídá kůži je dán vztahem:

$$p(skin|c) = \frac{p(c|skin)p(skin)}{p(c|skin)P(skin) + p(c|\neg skin)}$$

Daný vztah lze jiným způsobem vyjádřit přímo jako poměr mezi $p(c|skin)$ a $p(c|\neg skin)$, tedy pravděpodobnosti, že daný pixel náleží kůži a pravděpodobnosti, že náleží pozadí:

$$\Theta = \frac{p(c|skin)}{p(c|\neg skin)}$$

Z předchozího vyplývá, že hodnoty pravděpodobnosti pro každý pixel musí být někde uloženy. Vhodné je použití 2D histogramu [1], protože barva je většinou reprezentována pomocí 2D vektoru. Často se využívá histogram H-S (Hue a Saturation z HSV) nebo Cr-Cb z YCrCb.

Tato metoda je velmi oblíbená, protože je zároveň jednoduchá a také dostatečně efektivní. Snímek videosekvence je segmentován na základě vhodně zvoleného prahu proměnné Θ.

Výhodou této metody je, že je velmi rychlá a také pokud jsou histogramy vhodně „naučeny“, tak poskytuje velmi dobré výsledky[4].

2.4 Rozpoznání ruky v obraze

Rozpoznávání slouží k identifikaci objektů. Složitost tohoto problému spočívá v tom, že vlastnosti objektu, který by měl být rozpoznán, jsou často velice variabilní a navíc se podobných objektů může ve scéně nacházet spousta. Základem úspěchu je vybrat co nejvhodnější metodu pro dané podmínky, protože rozpoznávacích metod je mnoho, stejně jako způsobů popisu objektů. Objekt může být popsán tvarem, maskou, barvou, velikostí apod [17]. Popis vlastností objektu se nazývá deskriptor. Složitější deskriptory mohou být tvořeny 3D modely, kombinací jednodušších deskriptorů atp. Úlohou rozpoznávací metody je pak hledat podobnost deskriptoru objektu, který má být vyhledán s deskriptory kandidátů

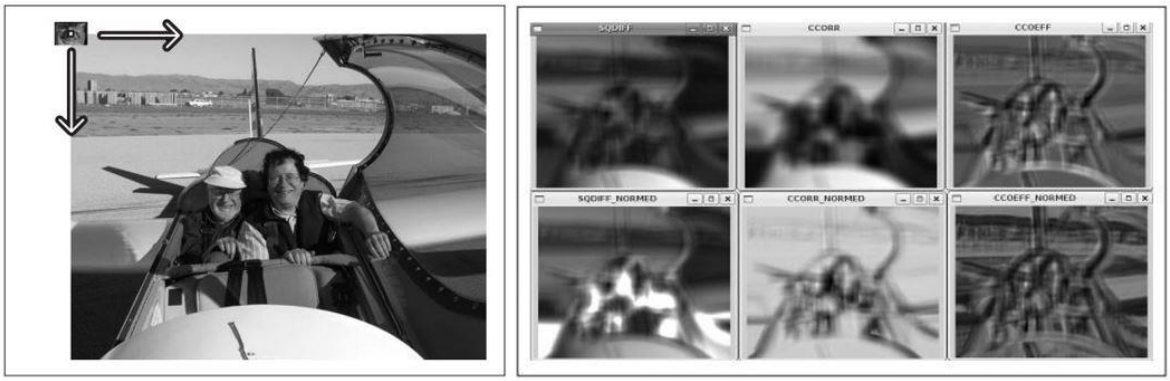
Problémem je, že jednodušší deskriptory nejsou většinou invariantní vůči rotaci, změně měřítka ani perspektivy. Obecně platí pravidlo, že čím je deskriptor robustnější, tím je potom algoritmus pomalejší. Neinvariantnost deskriptorů vůči výše zmíněním vlastnostem může být řešena vytvořením databáze deskriptorů daného vzoru, která obsahuje objekt v různých natočeních, velikostech apod. Invariantnost vůči změně měřítka (velikosti) je často zajištěna pomocí toho, že je vstupnímu snímku změněna velikost na několik úrovní a z nich se vytvoří tzv. pyramida. Jedním z invariantních deskriptorů jsou Huovy momenty. Je to vektor, který obsahuje 7 hodnot popisujících objekt, které jsou vypočítány pomocí momentů.

2.4.1 Template Matching

Jedním z nejjednodušších způsobů jak implementovat rozpoznávání je počítaní podobnosti dvou snímků [17]. Jeden snímek tvoří vzor (příklad na Obr. 7) a metoda zkoumá podobnost s druhým snímkem. Podobnost se dá počítat pomocí několika metod jako je vzájemná korelace (cross-corelation).

Nevýhodou této metody je, že není invariantní vůči rotaci objektu, ani vůči změně měřítka. Invariantnost vůči změně měřítka lze vytvořit pomocí (již výše zmíněných) pyramid. Nebo částečně zajistit pomocí normalizace snímků na předem danou velikost. Rozpoznání objektu v jiném natočení lze nejjednodušeji zajistit tak, že databáze bude obsahovat tento objekt v různých natočeních.

[4] Hsieh, C. C., Liou, D. H., & Lai, W. R. (2012). Enhanced Face-Based Adaptive Skin Color Model. *Journal of Applied Science and Engineering*, *15*(2), 167-176.



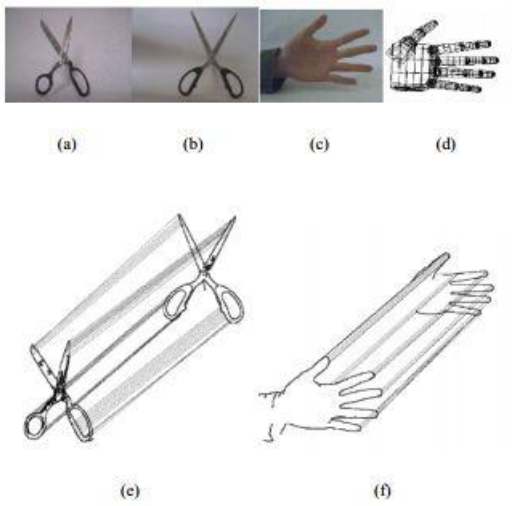
Obr. 7: ***Příklad template matchingu****. Vlevo nahoře vzor. Vpravo výsledky pro různé metody (zdroj:[2])*

2.4.2 Chamfer Matching

Chamfer Matching [10] je podobná metoda jako Template Matching. Rozdíl je v tom, že nepracuje s maskou, ale s popisem (deskriptorem) tvaru. Deskriptor tvoří pouze hranice objektu. Princip metody je takový, že pro každý pixel porovnávaného objektu je nalezen nejbližší pixel vzorového objektu. Všechny vzdálenosti jsou pak sečteny. Výsledek je většinou průměr těchto vzdáleností. Čím je výsledek menší, tím větší je podobnost mezi vzorem a porovnávaným objektem.

V praxi se algoritmus implementuje pomocí vzdálenostní transformace (distance transform, dále jen DT). Databáze vzorových objektů je tvořena jejich DT. Hranice porovnávaného objektu je pak korelována s danou DT.

Výhodou této metody je její rychlost a jednoduchost. Také je tolerantní k malým odchylkám ve vzájemném natočení objektů a změně měřítka. Situace je složitější, pokud je tato metoda aplikována na obraz, kde se nachází „více hran“ okolních (cizích) objektů.



Obr. 8: ***Identifikace objektů pomocí Chamfer Matching****: Porovnávaní dvojice objektů (a) a (b). Porovnání ruky s modelem (c) a (d). Identifikace jednotlivých částí při porovnávání (e) a (f) (zdroj:[10])*

2.5 Tracking

Sledování objektů neboli tracking je dalším podstatným krokem k rozpoznání gest [17][11]. Nejdůležitějším úkolem trackingu je schopnost identifikovat sledovaný objekt napříč videosekvencí. To umožňuje sledovat pohyb objektu, určit a analyzovat trajektorii, ale hlavně zachovat kontext sledovaného objektu.

Tracking je základem pro aplikace, které nabízejí nějakou interaktivitu. Může být hardwarově náročný. Jeho robustnost se odvíjí od typu aplikace. Pokud je odlišení objektů složitější, využívá tracking metod rozpoznání objektů (jako v 2.4). Pokročilejší metody jsou schopny zachovat kontext i u rychle se pohybujících objektů nebo objektů, které na chvíli zmizí ze scény a pak se zase objeví. Zřejmými příklady mohou být aplikace jako bezpečnostní systémy (sledování osob) nebo kontrola provozu (sledování aut).

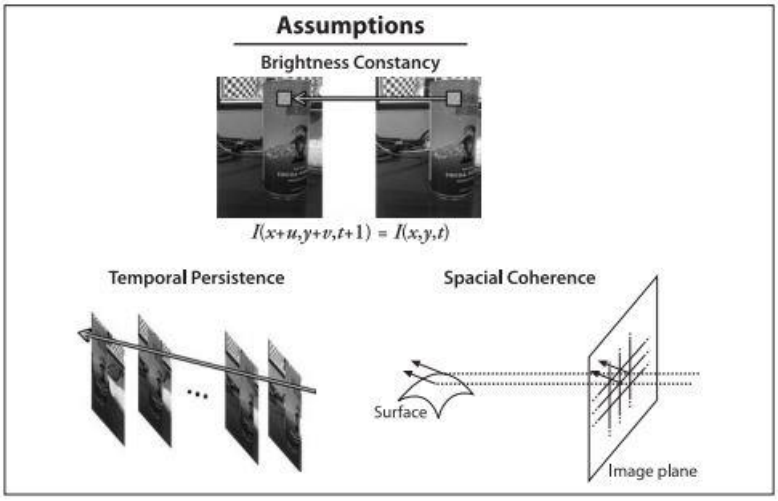
2.5.1 Optický tok

Optický tok (Optical Flow) [12][17] patří mezi základní metody trackingu. Pokud je aplikován na objekt, vypočítá ve dvou po sobě jdoucích snímcích tzv. displacement vektor. Ten určuje směr, kterým se vektor pohybuje a velikost vektoru reprezentuje míru změny (resp. rychlost).

Sledovaný objekt je často reprezentován jedním nebo více pixely. Pixely, které reprezentují objekt, se vybírají tak, aby ony i jejich okolí měly co nejstabilnější vlastnosti [5]. Tyto pixely pak mohou být v následujících snímcích identifikovány pomocí běžně používané metody Lucas-Kanade.

Tato metoda předpokládá, že optický tok je pro blízké okolí pixelu neměnný, a že intenzity pixelů a jejích okolí jsou v následujících snímcích téměř stejné. Metoda Lukas-Kanade [2] (Obr. 9) pracuje s výřezem o velikosti 3x3 kolem sledovaného pixelu. Rovnici pro výpočet displacement vektoru pak řeší pro všech těchto 9 pixelů.

Výhodou tohoto přístupu je, že pro detekovaný objekt je vybrán určitý počet pixelů, které ho mají reprezentovat. Poté je prováděn tracking pouze na základě těchto informací. Takže není důležitá změna tvaru objektu apod. Sledování touto metodou je relativně hardwarově nenáročné.



*Obr. 9: **Předpoklady při použití Lukas-Kanade**. Nahoře: Konstantní jas sledované oblasti. Vlevo dole: Malá změna polohy mezi snímky. Vpravo: Sousední pixely stále zůstávají (zdroj:[2])*

[5] Shi, J.; Tomasi, C., "Good features to track," *Computer Vision and Pattern Recognition, 1994*

3 Návrh

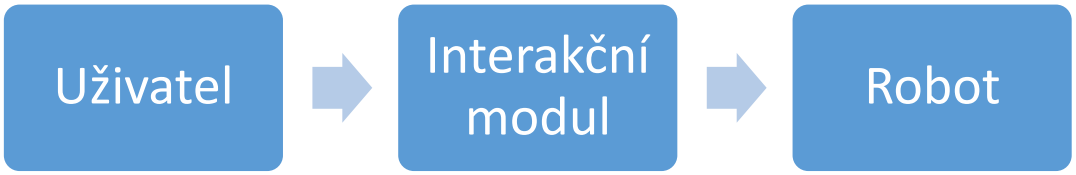
Tato kapitola se zabývá analýzou možností, které zadání nabízí. Na základě analýzy navrhuje řešení, specifikaci gest a požadavky na jednotlivé kroky řešení.

3.1 Analýza zadání

Ze zadání vyplývá, že spolu budou interagovat tři základní aktéři - uživatel, aplikace a robot (Obr. 11). Dále je v zadání uvedeno, že aplikace bude využívat nástrojů ROS a OpenCV. Aplikace bude navrhována pro funkci ve vnitřním prostředí. Dále následuje rozbor požadavků na každého aktéra a komunikace mezi nimi.

Obr. 10: ***Robot Ed***

Cílem práce je pohyb robotem. Aplikace bude vyvíjena pro komunikaci s obecným robotem pomocí systému ROS. Tento systém zajišťuje komunikaci mezi výstupem aplikace (příkazy pro robota) a řídící jednotkou robota. Díky univerzálnosti tohoto systému, by mělo být jednoduché použít řešení i na jiném zařízení.



Obr. 11: ***Základní schéma interakce***

V zadání není specifikováno, jaká gesta bude uživatel pro ovládání robota používat. Proto je bude nutné dobře specifikovat i s ohledem na standartní požadavky (2.1). Nejužitečnější se jeví použít neinvazivního typu gest. Takže uživatel nebude k použití aplikace potřebovat žádné další pomůcky. Důležité je se při návrhu také soustředit na to, aby bylo ovládání pro uživatele pohodlné a přirozené. Ze zadání nelze určit, jak dlouho bude uživatel s robotem pracovat. Z povahy aplikace lze předpokládat, že to bude několik minut až několik desítek minut. Bude potřeba také definovat vzdálenost uživatele od kamery a umístění kamery.

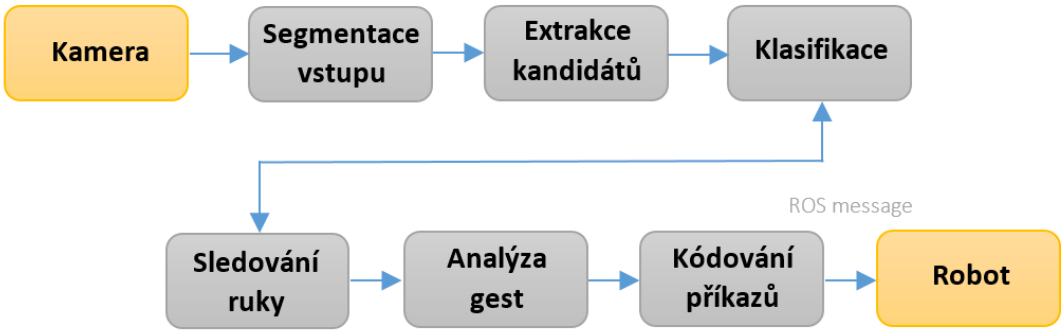
Pod pojmem interakční modul je v tomto případě myšlena samotná kamera, stejně jako hardware a software (aplikace), který data zpracovává. Použití samostatné RGB kamery je výhodné vtom, že je běžně dostupná (je jí vybaven skoro každý notebook). To znamená, že pokud by se podařilo vyvinout spolehlivý systém, mohla by být část aplikace rozšířena pro jiná použití.

Požadavky na robota jsou pouze takové, že ho lze ovládat pomocí ROS. V zadání je specifikováno, že cílem je ovládání pozemního robota. Díky univerzálnosti systému ROS by mělo jít ovládání teoreticky rozšířit na každého robota podobné povahy.

Způsob komunikace mezi robotem a snímačem závisí na celkovém uchopení aplikace. Kamera může být umístěna přímo na robotovi nebo může být umístěna v nějaké samostatné místnosti (např. operačním středisku). Také dnes získávají na popularitě systémy, které využívají kamer nošených přímo na těle (hlava nebo hruď). Komunikace interakčního modulu a robota může tedy probíhat přímo nebo prostřednictvím nějakého bezdrátového spojeni jako Wi-Fi, Bluetooth nebo internetu. Toto není v zadání přímo specifikováno. Výběr prostředí a umístění kamery může velice ovlivnit požadavky kladené na jednotlivé části řešení (např. detektor).

Pokud bude kamera umístěna na robotovi, tak uživatel zpětnou vazbu obdrží tak, že přímo uvidí pohyb robota. Pokud bude kamera sledující uživatele umístěna mimo robota, uživatel obdrží zpětnou vazbu buď tak, že na něho uvidí nebo na něm bude muset být umístěna nějaká další kamera, díky které bude uživatele vědět, kde se robot pohybuje (ale to je mimo rozsah této práce).

3.2 Specifikace výsledného řešení



*Obr. 12: **Základní blokové schéma***

Z analýzy zadání vyplývá, že v rámci návrhu řešení je potřeba se rozhodnout mezi několika variantami, které mohou výrazně ovlivnit složitost realizace, funkci a pojetí celé práce. Výsledné řešení bude vypadat následovně:

- **Aplikace**. Cílem práce je řešení, která v reálném čase zpracovává vstup z kamery a adekvátně reaguje na gesta uživatele. Uživatel s aplikací komunikuje pomocí dvou specifikovaných gest rukou (viz. kap. 3.3). Aplikace na základě rozpoznání a analýzy gest řídí pohyb robota. Zpracování obrazu probíhá pomocí knihovny OpenCV a komunikace s robotem prostřednictvím ROS.
- **Kamera**, která je spojená s počítačem zpracovávajícím data, bude umístěna mimo robota. Vstupem do aplikace je RGB výstup z jakékoliv kamery. Aplikace je navržena tak, aby pracovala ve vnitřním prostředí.
- **Rozhodl jsem se nevyužít data z hloubkového senzoru, protože mým cílem bylo vytvořit aplikaci, která bude dostupnější a nebude klást takové nároky na vybavení. Tím pádem pro její běh bude postačovat pouze obyčejná kamera**. Z toho plyne, že segmentace vstupního obrazu bude probíhat na základě modelu barvy kůže.
- **Uživatel** bude od kamery vzdálen **cca 80cm**. Měla by být umístěna přímo před ním (viz. kap. 3.3). Aplikace nebude vyžadovat žádné další nástroje jako barevné rukavice nebo podobné pomůcky. Bude ovládána čistě pomocí ruky. Uživatel bude k ovládání používat pouze jednu ruku. Předpokládá se, že v záběru bude celá hlava nebo její část a ruka, která slouží k ovládání. Vzdálenost 80cm odpovídá zhruba vzdálenosti kamery umístěné na monitoru a uživatele, který se trochu odsune od stolu. Pokud by měl být uživatel moc blízko, bude ruka zabírat moc velkou část obrazu a nebude místo k manipulaci.
- Sledování ruky bude započnuto tak, že uživatel nastaví svou ruku přímo na kameru. A její kontura nesmí splývat s obličejem.

Řešení lze logicky rozdělit na několik jednotlivých částí (Obr. 12). Toto dělení bude použito i v následujícím textu:

- **Segmentace na základě barvy kůže**. Výstupem této částí bude binární obraz (maska), který bude obsahovat oblasti odpovídající barvě kůže.

- **Klasifikace** má za úkol klasifikovat kandidáty extrahované z masky a určit zda odpovídají ruce nebo ne.
- **Tracking** (sledování ruky) sleduje klasifikovanou ruku ve videosekvenci.
- **Analýza gest** rozpoznává, kdy je které gesto aktivní a analýzou gesta vytváří odpovídající pokyny k ovládání robota.
- **Kódování příkazů** slouží k vytváření zpráv systému ROS.

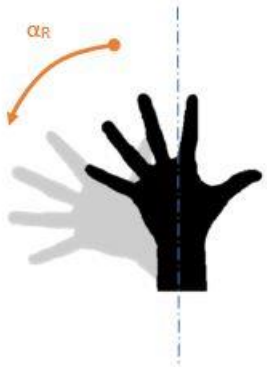
3.3 Specifikace gest a jejich použití

Protože gesta jsou jediným komunikačním prostředkem mezi uživatelem a robotem, je důležité je dobře specifikovat. Musí být jasné, jakou funkci budou plnit a jak se bude robot chovat. Gesta také nesmějí být moc „podobná“, aby se snížila pravděpodobnost, že je gesto špatně klasifikováno. Dalším úkolem je zřetelně určit, kdy gesto začíná a kdy končí. Lze použít aktivních zón, klidového stavu nebo speciální konfigurace ruky pro aktivaci gesta. Také by měla vyhovovat požadavkům v 2.1.

Statická gesta ke své klasifikaci nevyužívají informace o pohybu ruky, ale jsou náročnější na segmentaci. Výstup musí být alespoň natolik kvalitní, aby šlo určit, zda se jedná o otevřenou dlaň nebo zda je ruka v pěst.

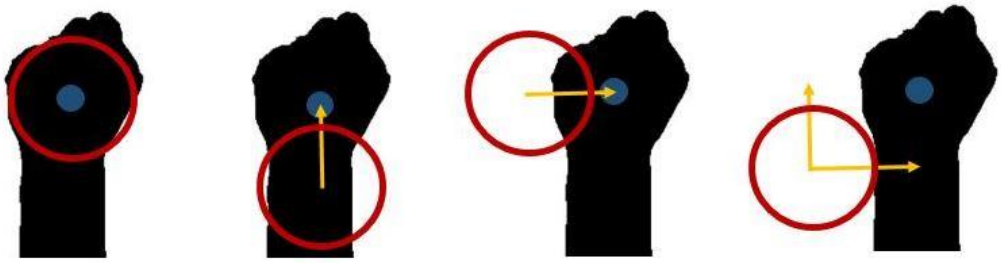
Gesta jsou navržena následovně:

- První gesto na Obr. 13 funguje tak, že uživatel ukazuje jednou rukou na kameru. To, do jaké strany a jak rychle se robot otáčí, je úměrné tomu, do jaké strany a jak moc je ruka natočena. Gesto je aktivní, pokud je ruka **otevřena**. Toto gesto umožnuje volný pohyb ruky v obraze. Pouze změna natočení ruky je chápána jako pokyn k pohybu robota.



*Obr. 13: **Ovládání pomocí natočení ruky***

- Pro aktivaci druhého gesta (Obr. 14) musí být ruka v **pěst**. Toto gesto funguje jako virtuální joystick, který má dvě osy (osa *x* a *y*). Červený kruh na obrázku (a) v oblasti dlaně tvoří mrtvou zónu. Modrý kruh reprezentuje střed dlaně. Na obrázku (b) lze vidět pohyb v kladném směru osy *y*. Žlutá šipka ukazuje míru vychýlení. Na (c) je pohyb v kladném směru osy *x*. Na (d) je složený pohyb v kladném směru osy *x* a *y*. Mrtvá zóna je umístěna v bodě, kde bylo gesto aktivováno. Toto gesto neumožnuje volný pohyb ruky. Každé vychýlení z mrtvé zóny je interpretováno jako pokyn k pohybu robota.



(a) (b) (c) (d)

Obr. 14: ***Gesto virtuální joystick***

3.4 Návrh implementace

V této podkapitole jsou probrány jednotlivé kroky výsledného řešení. Níže popsané metody tvoří jeden cyklus aplikace.

3.4.1 Segmentace obrazu

Prvním krokem celého řešení je segmentace vstupního obrazu. Výstupem bude maska, která by měla obsahovat oblasti reprezentující barvu kůže. Segmentace bude prováděna pouze na základě RGB vstupu. Řešení nebude používat informace z hloubkového senzoru (viz. 3.2).

Základem segmentace by měly být dva statické modely pro barvu pozadí a popředí (kůže). Kvalita tohoto statického modelu je kritická pro fungování celé aplikace. Model by měl být naučen pomocí rozsáhlé sady trénovacích dat, aby bylo dosaženo požadované robustnosti. Matice pravděpodobnosti pro vstupní snímek bude vypočítána pomocí následujícího vztahu, který udává, s jakou pravděpodobnosti daný pixel reprezentuje barvu kůže. Hodnoty jsou z intervalu <0,1>.

$$\Theta = \frac{p(c|skin)}{p(c|\neg skin)}$$

Jde o Bayesův klasifikátor popsaný v 2.3.3. Nyní je potřeba z matice pravděpodobnosti vytvořit masku. Úkolem masky je, aby reprezentovala pouze pozitivní případy, protože bude použita k učení dynamických modelů. Nejjednodušší bude využít prahování, možná bude lepší použít nějakou z automatických metod, jako třeba Otsuovu metodu (2.2.2).

Pomocí masky budou naučeny dva dynamické modely. Jeden pro barvu kůže a druhý pro pozadí. Modely budou implementovány pomocí histogramů. Dynamický model pozadí bude daleko přesnější, protože bude specifický právě pro aktuální situaci. Otázkou je, zda budou modely v každém cyklu učeny znovu, nebo zda bude histogram uchovávat kontext. Výstupní maska bude vytvořena podobně jako v případě statického modelu, avšak bude lepší použít jiné metody prahování.

3.4.2 Klasifikace objektů

Vstupem do klasifikace bude sada kandidátů, resp. kontury odpovídající objektům obsaženým v masce segmentace. Použitá metoda klasifikace bude jednoduchý Template Matching (2.4.1). Omezení metody vyhovují, protože bude stačit, když bude ruka klasifikována ve standartní vzdálenosti a s nulovým natočením.

Bude nutné vytvořit databází vzorů, která bude obsahovat dvě skupiny. Pro levou a pravou ruku. Pro každou ruku by měla databáze obsahovat alespoň 15 vzorů, které by měly dobře reprezentovat požadavky na specifikovanou vzdálenost uživatele, která je 80cm. Ruka nebude natočena, vždycky by měla být v poloze jako v Obr. 13.

Dalším úkolem bude správně stanovit práh úspěšnosti klasifikace. Výstupem metody bude informace určující, zda porovnávaný objekt odpovídá pravé, levé ruce nebo žádné.

3.4.3 Tracking ruky

Tracking bude mít za úkol v každém novém snímku identifikovat konturu ruky, protože zároveň s trackingem už nebude probíhat klasifikace. Protože nepředpokládám skokový pohyb ruky, jediné vodítko (clue) trackingu bude pozice sledované ruky v předchozím snímku. Objekt pak identifikuje nalezením nejmenší vzdáleností mezi objekty v současném snímku a pozicí ruky v předchozím snímku. Tato metoda by měla fungovat dostatečně. Standardně by měly být v obraze pouze dva objekty (hlava a ruka).

Pokud se sledovaná ruka hodně přiblíží jinému objektu, může dojít k „přeskočení“ sledování na daný objekt. Tomu lze zamezit tím, že bude specifikován práh minimální vzdálenosti. Pokud bude tomuto prahu vyhovovat více objektů, tak může být jako další vodítko použita například velikost kontury.

Také bude potřeba stanovit práh maximální vzdálenosti, aby pří zmizení ruky ze scény nebylo započato sledování jiného objektu, který by vyhověl té podmínce, že bude nejbližším objektem.

Občas se může stát, že sledovaný objekt, vlivem špatné segmentace, zmizí na chvíli z obrazu. Proto bude stanoven čas (počet snímků), po který bude probíhat sledování, přestože objekt zmizel.

3.4.4 Analýza gest

K rozlišení obou gest od sebe bude potřeba metoda schopná určit konfiguraci ruky. Na tuto úlohu se hodí metody k vyhledání prstů. Tyto metody jsou jednodušší a rychlejší než metody porovnávání vzorů. Další výhodou je, že není potřeba žádná databáze vzorů. Z počtu prstů se bude dát jednoduše určit aktuální konfigurace. Rozlišení je snadné, protože jedno gesto pracuje s otevřenou rukou (5 prstů) a druhé gesto s pěstí (žádné prsty).

První gesto (Obr. 13) pro svou funkci vyžaduje výpočet relativního natočení. Toho může být dosaženo pomocí metody *Phase Correlate*[6]. Implementace druhého gesta (Obr. 14) – virtuálního joysticku – nebude složitá. Jde pouze o počítání rozdílu aktuální polohy ruky od středu mrtvé zóny.

Jedním z požadavků na výstup je to, aby byl dostatečně modulární. To znamená, že formát výstupu půjde změnit nezávislé na implementaci celého předchozího řešení. Jak už bylo řečeno, robot bude ovládán pomocí systému ROS. To znamená, že výstupem aplikace budou takzvané ROS messages (zprávy). Formát zprávy závisí na tom, jaký typ zprávy cílový robot akceptuje.

[6] http://en.wikipedia.org/wiki/Phase_correlation

4 Implementace a testování

Tato kapitola se zabývá samotnou implementací jednotlivých kroků a otestováním některých částí řešení.

4.1 Implementační nástroje

4.1.1 ROS

ROS (Robot Operating Systém)[7] je flexibilní framework určený k tomu, aby vývojářům zjednodušil vytváření software pro roboty. ROS nabízí standartní služby operačního systému jako je abstrakce hardware, ovládání periferních zařízení, komunikace mezi procesy a správa balíků. ROS je v současné době open-source (BSD), a je neustále ve vývoji. Zhruba jednou za rok vyjde nová verze.

Jedním z cílů ROS je nabídnout vývojářům jednoduchou znuvupoužitelnost kódu. To usnadňuje systém procesů (nodes), který dovoluje, aby byl spustitelný kód vytvářen nezávisle na ostatních. Procesy mohou být pak seskupovány do balíčků (packages) nebo štosů (stacks), což umožnuje modularitu.

ROS přímo neobsahuje simulační nástroje, ale právě díky modulárnímu designu umožnuje využití externích nástrojů. Jedním z nejpoužívanější simulačních nástrojů je Gazebo [8]. Umožnuje komplexní simulaci díky využití několika fyzikálních enginů.

4.1.2 OpenCV

OpenCV[9] je veřejná open-source knihovna pro počítačové vidění. Je napsaná v C a C++. OpenCV bylo vytvořeno s tím úmyslem, aby umožnilo rychlé a efektivní zpracování obrazu pro aplikace pracující v reálném čase. Obsahuje více než 2500 metod běžně používaných ve zpracování obrazu a počítačovém vidění. Tyto metody jsou dobře optimalizované. OpenCV je také schopné využít vícejádrových procesorů.

První alfa verze byla vypuštěna v roce 2000. Jednou z motivací při tvorbě OpenCV byla jednoduchá a přehledná použitelnost. OpenCV je dnes aktivně používáno velkou spoustou velkých firem jako je Google, Yahoo!, Microsoft, Intel, IBM atd. Odhaduje se, že komunitu tvoří zhruba 47 tisíc lidí. Nedávno byla vypuštěna verze 2.4.9.

[7] http://www.ros.org/
[8] http://gazebosim.org/
[9] http://opencv.org/

4.2 Segmentace vstupu na základě modelu barvy kůže

4.2.1 Statický model barvy kůže

Statická segmentace je založena na modelu pro barvu kůže a modelu pro barvu pozadí. Protože nebylo v mých možnostech vytvořit robustní model na základě dostatečného množství trénovacích obrázků, tak jsem se rozhodl využít už vytvořený model [6-21s].

Oba modely jsou popsány 16 GMM v barevném prostoru RGB. Model pro pozadí byl vytvořen z trénovací databáze o velikosti 8965 snímků. Model barvy kůže byl vytvořen na základě 4675 manuálně označených snímků, které obsahují barvu kůže. Výpočet 32 GMM v každém cyklu algoritmu je časově velmi náročný. Počet GMM by se dal zmenšit. Bylo by možné odebrat modely s nejmenší váhou nebo použít metody, které by pomohly rozhodnout, které jednotlivé modely mají na celkovou přesnost nejmenší vliv (např. pomocí PCA).

Je daleko rozumnější oba tyto modely reprezentovat pomocí histogramu, protože vyhledání dané hodnoty v histogramu je nepochybně daleko rychlejší než výpočet výsledku v modelu pomocí GMM.

Zpočátku jsem se pokoušel model transformovat a uložit pomocí barevného prostoru YCrCb. S použitím hodnot Cr a Cb v rozsahu 0 až 255 pro každou dimenzi. Ale nepodařilo se mi najít způsob, jak vhodně vyřešit to, že více kombinací hodnot RGB se mapuje na stejné Cr-Cb souřadnice. Výsledkem byla špatná segmentace oblastí s nízkou nebo vysokou luminancí. Obdobně jsem se pokoušel uložit model pomocí normalizovaných souřadnic RGB se stejným problémem.

Nakonec jsem model uložil v původním prostoru RGB. Pokud by pro takový 3D histogram měla každá dimenze rozsah 255 hodnot a jednotlivé hodnoty by byly uloženy pomocí *float* (standardně 32bitů), tak by výsledný histogram měl velikost cca 500 MB. Což je prakticky nepoužitelné.

Proto jsem rozlišení modelu z 255 hodnot v každé dimenzi snížil na 85 hodnot. Výsledná velikost takového histogramu je cca 10 MB. Protože rozhodování zda se jedná o barvu kůže je vytvářeno na základě poměru hodnot pravděpodobnosti:

$$\Theta = \frac{p(c|skin)}{p(c|\neg skin)}$$

tak je pro ušetření místa a výpočetního času výhodnější ukládat přímo poměr $p(c|skin)$ a $p(c|\neg skin)$. Tím pádem používám pouze jeden histogram.

Výsledek segmentace je s porovnáním vůči výsledku dvou GMM modelů téměř totožný. Model je uložen v matici (*cv::Mat*). Tato matice je serializována a uložena ve formátu XML. Při každém spuštění programu je načtena do paměti.

Dalším krokem je vytvoření **masky** na základě matice obsahující pravděpodobnosti pro jednotlivé barevné pixely zdrojového snímku.

Pro tento účel využívám Otsuovu metodu prahování. Protože zpravidla takto vytvořená maska obsahuje pouze správně klasifikované pixely (Obr. 15). Tedy naopak neobsahuje pixely, které by byly nesprávně klasifikovány jako barva kůže. Tato maska je výstupem ze statické (offline) segmentace.

Segmentace pracuje v rozlišení 320x240px stejně jako celý zbytek aplikace. Toto rozlišení je dostatečně detailní a zároveň dostatečně malé, aby mohlo řešení fungovat v reálném čase.

Obr. 15: ***Segmentace stat. metodou****: Otsuova metoda prahování (a). Vstupní obrázek (b)*

4.2.2 Dynamické modely

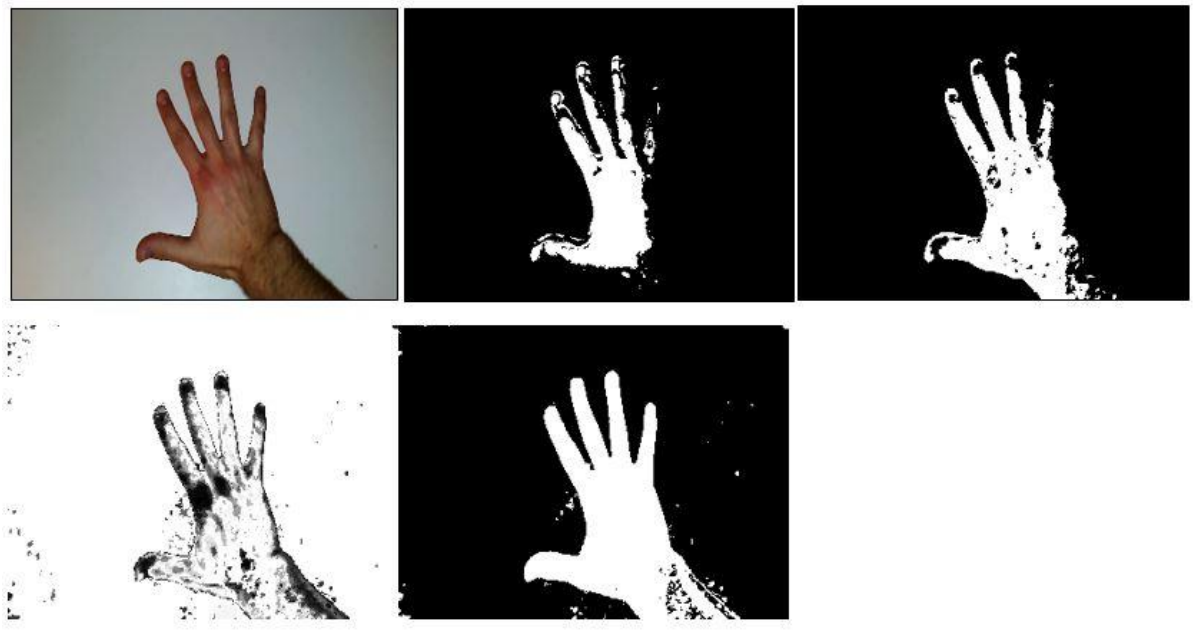
Dalším krokem segmentace je dynamické učení (online) histogramu. Jeden histogram pro barvu kůže a druhý jako model pozadí. Využívám vestavěné metody OpenCV a to *calcHist()* a *calcBackProject()*. Výhodou je jednoduchost použití a kvalitní optimalizace těchto metod. Motivací pro dynamické modely bylo to, že by měly daleko adekvátněji reprezentovat rozložení barvy kůže v dané situaci než statický model.

Jedním ze základních parametrů je volba barevného prostoru histogramu a počet binů. Zkoušel jsem barevné prostory HSV (Histogram H-S), YCrCb (Cr-Cb), Lab (a-b) a Luv (u-v). Po testování jsem vybral barevný prostor YCrCb. Počet binů má výrazný vliv na kvalitu segmentace. Zvolil jsem 64 binů pro model barvy kůže a 64 pro model pozadí.

Histogramy mohou fungovat ve dvou základních režimech. Prvním režim zachovává kontext a druhý ne (akumulace). Režim s akumulací funguje tak, že histogram není v každém cyklu programu přemazáván, ale zachovává kontext. A má tu výhodu, že může po čase konvergovat k modelu, který přesněji popisuje barvu kůže v daných podmínkách. Histogram většinou konverguje tím způsobem, že přiřazuje nejvyšší pravděpodobnost největším oblastem. Myslím, že by se toho dalo za určité řízené konvergence využít, ale nepřišel jsem na to, jak to spolehlivě udělat. Proto využívám režim, který histogramy neakumuluje.

Vstupní maska pro histogram barvy kůže je výstupní maska ze statické segmentace. Vstupní maska pro histogram pozadí je vytvořena negací masky, která je výstupem statické segmentace.

Posledním klíčovým úkolem je spojit výsledky těchto histogramů a vytvořit výslednou masku (Obr. 16). To jsem nakonec provedl tak, že snímek pravděpodobnosti modelu pozadí (váha 0.3) je negován a váhově sečten se snímkem pravděpodobnosti pro model kůže (s váhou 0.7). Následně je k tomuto složenému snímku přičten snímek pravděpodobnosti statického modelu. Takto složený snímek je prahován jeho průměrnou jasovou hodnotou. Výsledná maska je výstupem segmentace.



*Obr. 16: **Segmentace barvy kůže**. Z levého horního rohu: Vstupní obrázek (1). Prahování Otsuovou metodou (2). Dyn. model kůže (3). Dyn. model pozadí (4). Výsledná maska (5)*

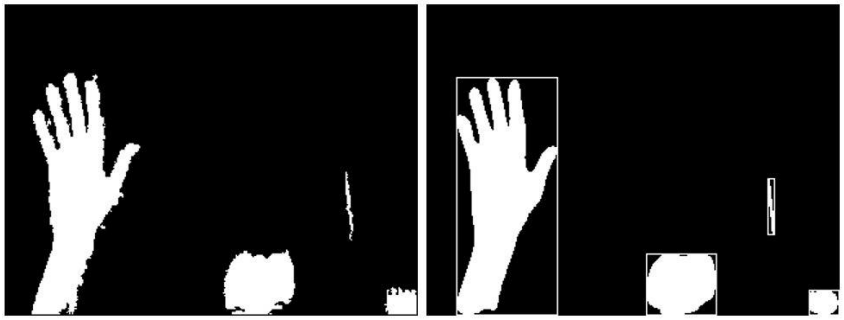
4.3 Extrakce kandidátů

Výstupní maska segmentace je před analýzou upravena morfologickou operací otevření pomocí vestavěných funkcí *erode()* a *dilate()* s 3x3 obdélníkovým strukturním elementem. Tento postup by měl vést k částečnému omezení vlivu šumu.

V takto upravené masce jsou vyhledány kontury pomocí vestavěné funkce *findContours()*. Použitá metoda ukládání je *CV_CHAIN_APPROX_NONE*. Která do výstupního vektoru ukládá všechny body kontury. Další možností jsou aproximace, které komprimují horizontální, vertikální a diagonální segmenty, ale tímto se komplikuje následná analýza kontury.

Jedním ze základních kroků klasifikace kontur je jejich filtrace na základě velikosti. Podstatné je omezení minimální velikosti, aby se odstranili kontury reprezentující šum nebo objekty vzdálenější od kamery. Jednoduchým experimentem jsem zjistil, že velikost kontury ruky v maximální předpokládané vzdálenosti od kamery je zhruba 200px. Takže jsem dolní práh velikosti s určitou rezervou nastavil na 100px. Horní práh kontury se nedá tak jednoduše určit, protože v této fázi kontura obsahuje ruku i část paže. Také není stanovena minimální vzdálenost ruky od kamery. Navíc toto prahování nemá logický význam.

V této fázi algoritmus postupně v cyklu prochází jednotlivé kontury. Prvním krokem je extrakce kontur. To provádím pomocí funkce *boundingRect()*. Dále je kontura vyhlazena metodou *blur()*. Vyhlazení je důležité, protože tvar kontury je často negativně ovlivněn šumem a nedokonalostí segmentace. Po vyhlazení je kontura zpravidla jednodušší reprezentací požadovaného tvaru, což usnadňuje klasifikaci a především alg. pro získání počtu prstů (Obr. 17).



*Obr. 17: **Vyhlazení a extrakce kontury**. Zleva: Maska před zpracováním (1). Kontury po vyhlazení (2).*

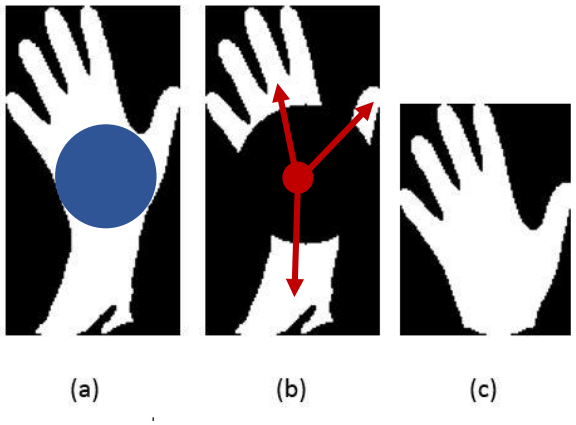
4.3.1 Segmentace paže

Pro segmentaci předloktí od zbytku ruky jsem navrhl vlastní postup. Nevím, zda je originální, ale nikde jsem na něj zatím nenarazil. Jeden z klasických postupů využívá předpokladu toho, že šířka ruky v zápěstí je nejmenší. Největší problém, na který jsem narazil, je správně najít osu, podle které měřit šířku ruky. Zkoušel jsem pracovat s metodou PCA nebo na stejném principu fungující metodou OpenCV – *fitEllipse()*. Stačila ale malá odchylka od reálné hodnoty a segmentace paže produkovala špatné výsledky. Vím, že ještě existují podobné postupy založené na stejném předpokladu, ale stejně jsem měl pocit, že jsou tyto metody málo robustní.

Využil jsem předpokladu, že největší vepsaný kruh na ruce se bude nacházet zhruba v oblasti dlaně. Střed tohoto kruhu se dá najít pomocí vzdálenostní transformace (Distance Transform, dále jen DT). Tedy jako souřadnice **maximální** hodnoty této transformace, kde poloměr dlaně odpovídá přímo této hodnotě. Tento předpoklad nemusí vždy platit. Může být porušen, pokud je paže blíž kameře než dlaň nebo v důsledku šumu, nekvalitní segmentace atp.

Po nalezení středu a velikosti (poloměru) dlaně je tento poloměr zvětšen o 35% (empiricky určená hodnota) a vepsán do masky aktuální kontury. Dá se říct, že algoritmus pracuje graficky. Zvětšení má za cíl separovat konturu palce, předloktí a zbytku dlaně s horními prsty (Obr. 18). Měly by takto vzniknout 3 kontury. Dalším krokem je pouze správně identifikovat konturu předloktí a odstranit ji.

To provádím za oprávněného předpokladu, že pokud bude dlaň středem pomyslné jednotkové kružnice, tak předloktí bude ležet v úhlu cca 270 stupňů s odchylkou +/- 35 stupňů. Kontura předloktí je odstraněna a výsledná maska je vytvořena spojením kontur dlaně, palce a horních prstů.



*Obr. 18: **Segmentace paže**. Vstupní maska kontury ruky – detekce dlaně (modrý kruh) (a). Separace jednotlivých částí pomocí zvětšeného poloměru dlaně (b). Výsledná maska po segmentaci (c)*

4.4 Klasifikace ruky z množiny kandidátů

Velmi důležitou součástí algoritmu je klasifikace objektů. Tato část má za úkol určit, zda vstupní objekt reprezentuje ruku.

Rozhodl jsem se použít knihovní funkci OpenCV – *matchTemplate*(). Vstupem jsou dvě masky. Výhodou je, že je oproti vlastní implementaci optimalizovaná a poskytuje jednouchý interface. Tato funkce vypočte podobnost dvou masek na základě předvolené metody. Používám metodu *CV_TM_CCOEFF_NORMED*, vykazuje daleko lepší výsledky než ostatní, protože oproti nim je výsledná míra podobnosti u dvou naprosto rozdílných masek velmi malá. U masek podobných je naopak velká. Výsledek je míra podobnosti od 0 do 1.

Tato metoda není invariantní proti rotaci, ani proti změně měřítka. Invariantnost proti rotaci není potřebná, protože než započne tracking, ruka je identifikována v nenatočené pozici. Naopak jsem chtěl dosáhnout alespoň nějaké invarianci vůči změně měřítka, protože ruka nemusí být vzdálená optimálních 80 cm (viz. v 3.2), ale nelze předpokládat, že uživatel tuto vzdálenost přesně odhadne.

Databázi vzorových masek tvoří 18 vzorů pro pravou a 18 vzorů pro levou ruku. Vzorové masky byly vytvořeny v předpokládané vzdálenosti ruky od kamery s různými odchylkami (cca +/- 10cm) s drobnou variancí v natočení ruky. Tedy konkrétně ve třech vzdálenostech cca 70cm, 80cm a 90cm. V každé vzdálenosti byla ruka zachycena ve třech natočeních. Mírně doleva, pak nenatočená a nakonec mírně natočena doprava. Ještě byla vždy ruka zachycena s „plně roztaženými prsty“ a s prsty blízko u sebe. Dohromady tedy 3*3*2 = 18 vzorů.

Průměrná šířka vzorů byla 85px. Za účelem zvýšení robustnosti vůči změně měřítku jsem všechny vzory normalizoval podle šířky na tuto hodnotu při zachování jejich poměru výška/šířka. Při klasifikace je vstupní masce také změněna šířka na tuto velikost.

Dalším snahou o zpřesnění výsledku je provedeno tak, že se vezme v potaz podobnost poměru šířka/výška obou objektů. Implementováno je to následovně:

$$výsledná\ podobnost = míra\ podobnosti * \frac{MIN(r1, r2)}{MAX(r1, r2)}$$

$$r_i = \frac{výška\ masky}{šírka\ masky}$$

Někdy se totiž stává, že po provedení změny velikosti masky na normalizačních 85px vykazují některé kontury po klasifikaci přehnaně pozitivní výsledky a mohou být nesprávně identifikovány jako ruka.

Vzory jsou rozděleny na dvě skupiny – levou a pravou ruku. Jako výsledek skupiny je brán nejlepší výsledek z 18 vzorů. Na základě výsledku je rozhodnuto, zda porovnávaný objekt odpovídá první skupině (levá ruka) nebo druhé skupině (pravá ruka). Skoré musí přesáhnout hodnotu alespoň 0.70, aby bylo možné kandidáta považovat za ruku.

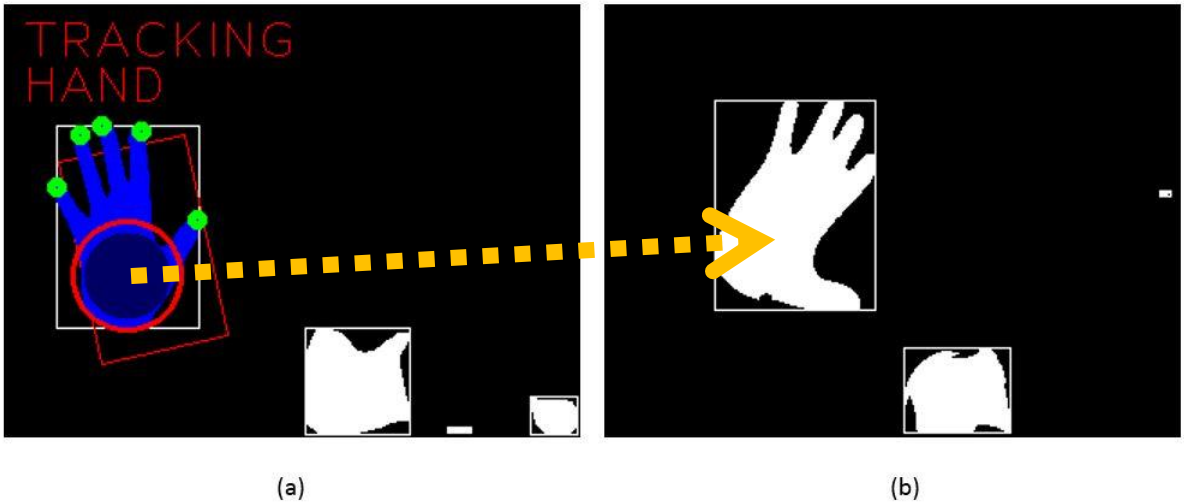
4.5 Sledování ruky

V tomto případě bude úloha trackingu docela jednoduchá a nebudou na ní kladeny velké požadavky. Účelem je sledovat pouze jeden objekt, kterým bude ruka. Většinou bude v každém novém snímku několik neidentifikovaných kandidátů a tracking bude mít za úkol identifikovat ruku na základě informací z předchozího snímku.

Úlohou trackingu je sledovat pouze jeden identifikovaný objekt. Pokud není sledován (trackován) žádný objekt, tak je v každém cyklu algoritmu hledána kontura ruky pomocí klasifikace (viz kap…). Jakmile je nějaká kontura klasifikována, tak je klasifikace přerušena a tento objekt je sledován, dokud nezmizí z obrazu.

Tracking je řízen pomocí jednoduchého vodítka vzdálenosti. V každém novém cyklu je vybrána kontura, která je nejblíž umístění sledované kontury v předchozím snímku. Nepředpokládám tak velkou změnu pozice, že by začal být sledován jiný objekt, protože by měl aktuálně menší rozdíl vzdálenosti.

Pokud objekt zmizí ze scény nebo pokud dojde k chybě v segmentaci a není možné dál sledovat daný objekt, tak se algoritmus snaží ho ještě po 3 následující snímky nalézt. Jeho umístění se nemění. Příklad je na
Obr. 19. Na (a) je sledována ruka. Úkolem trackingu je identifikovat ruku na obr (b). Z obrázku je zřejmé, že pozice ruky na (b) je nejblíže ze všech objektů pozici ruky na (a).

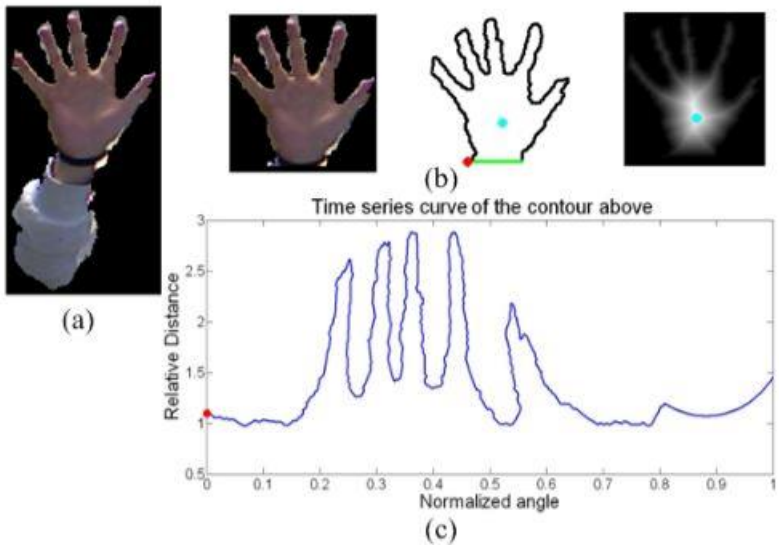


*Obr. 19: **Tracking**. Snímek v čase T (a). Snímek v čase T+1 (b)*

4.6 Analýza gest

4.6.1 Určení počtu prstů

Základní rozlišení mezi dvěma gesty pomocí určení toho, v jaké je ruka konfiguraci. Z návrhu vyplývá, že stačí určit, zda je ruka otevřená nebo v pěsti. Za tímto účelem jsem implementoval algoritmus na rozpoznání počtu prstů.



*Obr. 20: **Time-Series Curve**. Vstupní obrázek (a). Segmentovaná ruka (b). Průběh TSC (c). (zdroj:[18])*

Algoritmus je založen na výpočtu tzv. Time-Series Curve [18]. Což je průběh, který reprezentuje vzdálenost hranice od středu kontury v každém bodu této hranice. Vstup do algoritmu tvoří kontury ruky. Střed kontury určuji stejně jako v algoritmu pro segmentaci paže v 4.3.1. Ve zkratce

je na masku ruky aplikována DT (Obr. 20-b) a středem je bod, který je reprezentován největší vzdáleností od nejbližší hranice kontury. Pak je v cyklu procházen každý bod hranice a je pro něj vypočítána vzdálenost od středu dlaně. Tyto vzdálenosti jsou uloženy do vektoru. Ukázalo se, že pro správnou funkci algoritmu není zásadní, aby průběh obsahoval vzdálenosti všech bodů hranice. Takže za účelem urychlení algoritmu je do výsledného průběhu započten pouze každý pátý bod hranice kontury. Tato hodnota byla určena empiricky a vyhovuje všem podmínkám.

Dalším krokem je najít v průběhu lokální maxima, které má reprezentovat prsty. To by mohlo být realizováno pomocí analýzy derivace průběhu, ale nakonec jsem našel efektivnější řešení, které poskytuje spolehlivé výsledky. Základní předpoklad algoritmu, který vyhledává lokální maxima, je ten, že pro bod maxima $P_{\max(i)}$ platí:

$$P_{1(i-w)} < P_{\max(i)} \; a \; zároveň \; P_{2(i+w)} < P_{\max(i)}$$

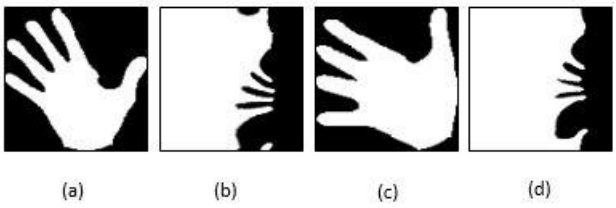
Kde *w* je předem zvolený krok vyhledávání.

Z předchozího vyplývá, že pro správnou funkci algoritmu je zásadní, aby se lokální maxima nacházela pouze tam, kde mají být prsty. To nemusí platit, protože hranice kontury ruky je po segmentaci většinou „křivá“. Proto je tak důležité vyhlazení kontury, které tyto „falešná“ maxima odstraní.

Všechna takto vyhledaná maxima ještě nemusí reprezentovat prsty. Protože jsou často nesprávně klasifikovány body, které tvoří „roh“ hranice ve spodní oblasti zápěstí, jako prsty, tak je důležité provést další filtraci. Zamítnuta jsou ta maxima, jejichž vzdálenost od středu dlaně je menší než 110% poloměru samotné dlaně.

Podle počtu prstů lze určit, zda je ruka v pěst nebo zda je otevřená.

4.6.2 Měření změny natočení ruky



*Obr. 21: **Výpočet změny natočení**. Maska 1 (a). Její transformace do pol. souřadnic (b). Maska 2 (c). Transformovaná maska 2 do polárních souřadnic. (d)*

Výpočet změny natočení ruky je prováděn pomocí vestavěné funkce *phaseCorrelate()*. Tato funkce je schopna určit vzájemné posunutí dvou snímků. Vstupem jsou dvě masky ruky v čase T a čase T-1. Před aplikací funkce *phaseCorrelate()* jsou snímky normalizovány na velikost 100x100 pixelů a poté převedeny do polárních souřadnic.

Výpočet vzájemného posunutí v tomto souřadném systému zajistí výpočet vzájemného pootočení ruky v čase.

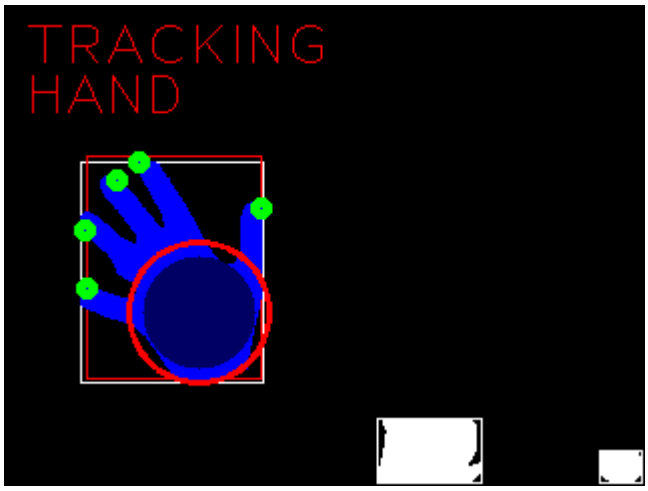
Funkce aplikuje na oba vstupní obrazy Fourierovu transformaci. Poté je vypočteno tzv. „cross-power spectrum“ a provedena jeho korelace. Na výsledný obraz je aplikována inverzní Fourierova transformace. Funkce vrátí bod maxima této transformace, který by měl odpovídat posunutí. Výsledný úhel pootočení je vypočítán jako:

$$\alpha = MaxPt.x * 360/50$$

Výsledný úhel je vypočítán z x-ové souřadnice, protože osa x v polárním obrázku odpovídá úhlu. Hodnota je následně z rozsahu 0 až 50 (šířka polární obrázku) normalizována do rozsahu 0 až 360.

Algoritmus funguje dobře, pokud obě masky reprezentují stejný tvar, akorát vzájemně pootočený. Pokud je podobnost malá vlivem špatné segmentace kontury, tak je výsledek nepřesný. Proto je výsledek zamítnut, je-li jeho absolutní hodnota větší než **8 stupňů**. Nepřesný výsledky často výrazně přesahuje tuto hodnotu.

Vzhledem k tomu, že mohu standartní rychlost otáčení definovat jako 90 stupňů za sekundu. Tak při 30 snímcích/s je vzájemné posunutí 3 stupně. Takže práh o velikost 5 poskytuje dostatečnou rezervu. Na Obr. 22 je ukázka tohoto gesta ve výsledném řešení.

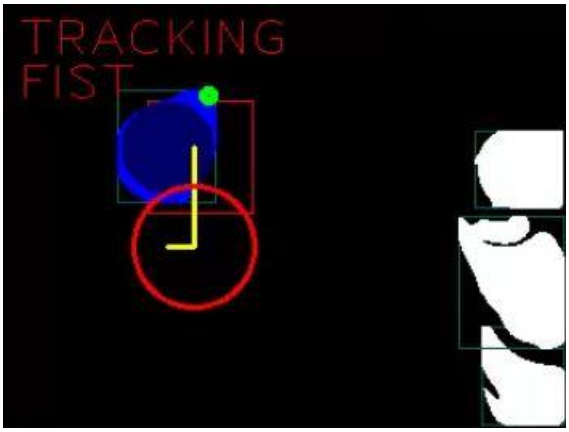


Obr. 22: ***Ovládání pomocí natočení ruky***

4.6.3 Gesto virtuální joystick

Implementace tohoto gesta nebyla složitá. Výchozím bodem je střed mrtvé zóny (odpovídá středu červeného kruhu na Obr. 23). Mrtvá zóna je v obraze umístěna tam, kde započne aktivace tohoto gesta. Tedy tam, kde uživatel dá ruku v pěst. Poloměr mrtvé zóny je o 25% větší než poloměr dlaně. Účel mrtvé zóny je takový, aby nedocházelo k nežádoucímu pohybu robota v důsledku drobných pohybů ruky, které nemají být interpretovány jako pokyn.

Pomocí gesta lze ovládat pohyb ve dvou osách (x a y). Intuitivně pohyb ruky nahoru z mrtvé zóny znamená pohyb v kladném směru osy y (pohyb dolů = záporný směr). Protože kamera vnímá strany zrcadlově otočené oproti uživateli, tak pohyb ruky doleva (z pohledu aplikace) znamená kladný směr v ose x (doprava = záporný směr). Mírou vychýlení ruky od mrtvé zóny lze určovat změnu rychlost nebo rychlost otáčení.



Obr. 23: ***Gesto virtuální joystick.*** *Vychýlení v kladném směru osy y.*

4.7 Vyhodnocení klasifikace

Testování klasifikace probíhalo na datové sadě, která obsahuje 6 uživatelů. Každý uživatel prováděl gesto, které probíhalo tak, že po spuštění nahrávání zvedl ruku do úrovně obličeje a rukou otočil o 90 stupňů. Uživatel gesto prováděl ve třech vzdálenostech (70cm, 120cm a 170cm). Třemi rychlostmi (pomalu, středně rychle a rychle). A ještě každou rukou a proti směru i po směru hodinových ručiček. Pro jednoho uživatele dohromady tedy 36 vzorků. Pro všech 6 to bylo 216 vzorků dohromady. Přesnost natočení jsem se nakonec rozhodl netestovat, protože uživatele gesto většinou neotočili o plných 90 stupňů. Často ruku při vraceni dotáčeli zpátky atd.

Klasifikaci, tedy rozpoznání ruky, jsem se rozhod testovat tak, že pro každé video zjistím, zda v něm byla klasifikována ruka, protože uživatel před započetím gesta drží ruku v neotočené pozici a přímo na kameru, tedy přesně podle specifikace (3.2).

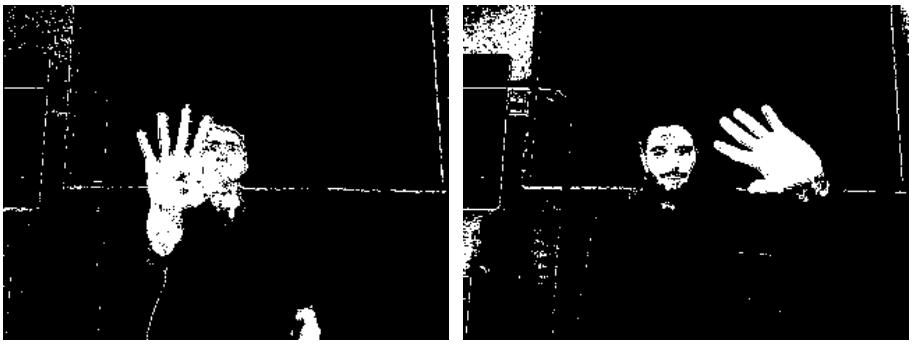
Bohužel z 6 uživatelů jsem mohl použít pouze jednoho, protože většina uživatelů gesto vykonávalo přímo před svým obličejem (Obr. 24 – levý obrázek). Což samozřejmě nebyla jejich chyba, ale při nahrávaní jsem si tento problém neuvědomil. Takže jsem nakonec testoval pouze na 36 vzorcích. 12 pro každou vzdálenost. Nejbližší vzdálenost byla 70cm, což s odchylkou 10cm odpovídá specifikované vzdálenosti mé aplikace.

Vzdálenost [cm]	**Počet správně klas.**	**Úspěšnost [%]**
70	10/12	83.3
120	5/12	41.6
170	0/12	0

Tab 1: ***Úspěšnost klasifikace***

V tomto vzorku nebyl ani jeden nesprávně klasifikovaný objekt jako ruka. Na blízkou vzdálenost byla úspěšnost vysoká. Ve dvou případech ke klasifikaci nedošlo. V jednom byl pohyb moc rychlý a ve druhém ruka splývala s konturou obličeje. Ve střední vzdálenosti, která je o 40cm větší než

specifikovaná pro toto řešení, byla ruka klasifikována jen v 5 případech, což se dalo očekávat vzhledem k použité metodě. A na největší vzdálenost, která je o 90cm větší než specifikována, nebyla ruka klasifikována ani jednou. Na tuto vzdálenost už vykazuje špatné výsledky i segmentace. Kontura ruky je hodně ovlivněna šumem.

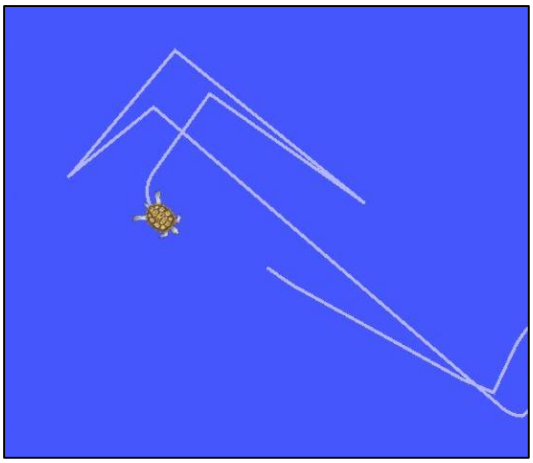


*Obr. 24: **Testování klasifikace**. Levý obrázek: Ruka splývá s obličejem. Pravý obrázek: Správná pozice ruk*

4.8 Testování na simulátoru

Výsledné řešení jsem testoval na aplikaci dostupné v ROS. Jde o *TurtleSim* (Obr. 25). Želva, která je po inicializaci umístěná uprostřed, lze ovládat pomocí zpráv (ROS message). TurtleSim naslouchá na topicu */turtle1/cmd_vel*. Typ zprávy, který topic očekává je *geometry_msgs/Twist*. Ovládání by šlo lehce rozšířit i na jiného robota. Záleží jenom na tom, jaký typ zpráv podporuje.

Lze ovládat v podstatě pouze pohyb dozadu, dopředu, jeho rychlost a natočení doprava a doleva. Pohyb dopředu a dozadu je ovládán osou x druhého gesta (virtuální joystick). Osa y ovládá natočení. První gesto (natočení ruky) také slouží k ovládání natočení želvy.



*Obr. 25: **Ovládání TurtleSim - pohyb rovně se zatáčením**: Nahoře vlevo: Vstupní obrázek. Nahoře vpravo: Aplikace. Dole: TurtleSim*

5 Závěr

Tato práce měla za cíl vytvořit aplikaci na ovládání pozemního robota gesty. Byla navrhnuta gesta a postup byl rozdělen do několika částí, které odpovídají standartnímu dělení v této oblasti. Řešení se skládá ze segmentace na základě barvy kůže, identifikace ruky, její sledování a vyhodnocení gest.

Pro základ segmentace na základě barvy kůže byl využit statický model pro barvu kůže a pro barvu pozadí. Výsledky se snaží vylepšit dynamický model, který by měl přesněji reprezentovat aktuální situaci. Identifikace probíhá pomocí klasifikace objektů na základě template matchingu. Byla vytvořena databáze pro levou a pravou ruku a navrhnut postup, který se snažil zvýšit nezávislost na velikosti ruky v obraze. Pro správné fungování klasifikace byl implementován algoritmus pro segmentaci předloktí. Klasifikovaná ruka je následně sledována pomocí jednoduchého vodítka pozice ruky v předchozím snímku. Gesta jsou poté analyzována a vyhodnocena. První gesto ovládá pohyb robota pomocí natočení ruky, druhé gesto funguje jako virtuální joystick. Ovládací pokyny pro robota jsou zasílány pomocí zpráv ROS. Testování aplikace probíhalo na TurtleSim v ROS.

Pro vylepšení segmentace by mohla být zavedena inicializace modelu barvy kůže pomocí rozpoznání tváře. Tento model by pak věrněji reprezentoval aktuální rozložení barvy kůže. Také by mohla být do budoucna zvýšena robustnost sledování, aby se dokázalo vypořádat s chvilkovým prolnutím dvou kontur. Dále by mohlo být zavedeno ještě jedno gesto. Uživatel by mohl ovládat robota pomocí přibližování a oddalování ruky od kamery. Vyhodnocení by probíhalo na základě změny velikosti detekované dlaně.

Literatura

[1] Antonis A. Argyros and Manolis I.A. Lourakis (2005). Tracking Skin-Colored Objects in Real-Time, Cutting Edge Robotics, Vedran Kordic, Aleksandar Lazinica and Munir Merdan (Ed.), ISBN: 3-86611-038-3, InTech

[2] Bradski G. R., Kaehler A. Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media, Inc. 2008.

[3] D. Reynolds, Gaussian Mixture Models, 1995, MIT Lincoln Laboratory

[4] Daeho Lee; SeungGwan Lee, Vision-Based Finger Action Recognition by Angle Detection and Contour Analysis, ETRI Journal . Jun2011, Vol. 33 Issue 3, p415-422. 8p

[5] I. Pavlovi'c, Rajeev Sharma, Thomas S. Huang, Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19 (1997), pp. 677-695

[6] Jones, M.J.; Rehg, J.M., "Statistical color models with application to skin detection," *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.* , vol.1, no., pp.,280 Vol. 1, 1999

[7] Juan Pablo Wachs, Mathias Kölsch, Helman Stern, and Yael Edan. 2011. Vision-based hand-gesture applications. *Commun. ACM* 54, 2 (February 2011), 60-71.

[8] Kovac, J.; Peer, P.; Solina, F., "Human skin color clustering for face detection," *EUROCON 2003. Computer as a Tool. The IEEE Region 8* , vol.2, no., pp.144,148 vol.2, 22-24 Sept.

[9] K. Papoutsakis, A.A. Argyros, Integrating tracking with fine object segmentation, Image and Vision Computing, Volume 31, Issue 10, pp. 771-785, Oct. 2013.

[10] Ming-Yu Liu; Tuzel, O.; Veeraraghavan, A.; Chellappa, R., "Fast directional chamfer matching," *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* , vol., no., pp.1696,1703, 13-18 June 2010

[11] M. Kolsch, Vision Based Hand Gesture Interfaces for Wereable Computing and Virtual Enviroment, 2004, Dissertation, University of California

[12] M. Sonka, V. Hlaváč, R. Boyle: Image Processing, Analysis, and Machine Vision, CL-Engineering, ISBN-13: 978-0495082521, 2007.

[13] Osman, G., Hitam, M. S. & Ismail, M. N. (2012). Enhanced skin colour classifier using RGB Ratio model. *CoRR*, abs/1212.2692.

[14] R. Laganière, OpenCV 2 Computer Vision Application Programming Cookbook, Packt Publishing Ltd, 2011

[15] V. Vezhnevets et al., A Survey on Pixel-Based Skin Color Detection Techniques, 2003, IN PROC. GRAPHICON-2003, pp 85-92

[16] Wei Wang; Jing Pan, "Hand segmentation using skin color and background information," Machine Learning and Cybernetics (ICMLC), 2012 International Conference on , vol.4, no., pp.1487,1492, 15-17 July 2012

[17] X.Zabulis, H. Baltzakis, A. A. Argyros, "Vision-based Hand Gesture Recognition for Human Computer Interaction", Chapter 34, in "The Universal Access Handbook", Lawrence Erlbaum Associates, Inc. (LEA), Series on "Human Factors and Ergonomics", ISBN: 978-0-8058-6280-5, pp 34.1 - 34.30, Jun 2009

[18] Zhou Ren, Junsong Yuan, and Zhengyou Zhang. 2011. Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. In *Proceedings of the 19th ACM international conference on Multimedia* (MM '11). ACM, New York, NY, USA, 1093-1096.

Příloha CD

Přibalené CD obsahuje:

- **plakatek.pdf**
- **videoprezentace.mp4**
- **tz.pdf**
  Technická zpráva ve formátu pdf.
- **tz.docx**
  Zdrojový soubor technické zprávy.
- **dema/**
  Tato složka obsahuje vstupní videa pro aplikaci.
- **handGesture/**
  Tato složka obsahuje samotné řešení. Představuje ROS package.
- **README.txt**
  Pokyny ke spuštění aplikace.