Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

Bakalářská práce

Detekce pohybu míčku pro mechatronický model stolního fotbalu

Plzeň, 2020 Matěj Sieber

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Matěj SIEBER**
Osobní číslo:	**A17B0574P**
Studijní program:	**B3918 Aplikované vědy a informatika**
Studijní obor:	**Kybernetika a řídicí technika**
Téma práce:	**Detekce pohybu míčku pro mechatronický model stolního fotbalu**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

1. Nastudujte problematiku počítačového vidění a zpracování dat na zařízení s malým výpočetním výkonem.
2. Navrhněte algoritmus detekce míčku a jeho pohybu pro zjednodušený mechatronický model stolního fotbalu. Zaměřte se na odezvu v reálném čase.
3. Navrhněte vhodný způsob snímání obrazu a umístění kamery.
4. Realizujte a otestujte navržený detektor a prediktor pohybu s využitím vhodných HW a SW prostředků.

Rozsah bakalářské práce: **30-40**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

Dodá vedoucí práce.

Vedoucí bakalářské práce: **Ing. Luboš Šmídl, Ph.D.**
Katedra kybernetiky

Datum zadání bakalářské práce: **15. října 2019**
Termín odevzdání bakalářské práce: **25. května 2020**

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
L.S.

Doc. Dr. Ing. Vlasta Radová
děkanka

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 15. října 2019

PROHLÁŠENÍ

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne .....

Matěj Sieber

Poděkování

Chtěl bych poděkovat svému vedoucímu bakalářské práce Ing. Luboši Šmídlovi Ph.D. za odborné vedení, za pomoc a rady při zpracování této práce. Mé poděkováni patři též Ing. Martinu Goubejovi Ph.D. za pomoc s částí predikce.

Abstrakt

Cílem práce je návrh detektoru a prediktoru pro mechatronický model stolního fotbálku. V teoretické části této práce jsou představeny metody pro zpracování obrazu, detekci míčku a nakonec predikci pohybu. V testovací části jsou jednotlivé metody vyzkoušeny a jsou uvedeny jejich výstupy. V závěru je systém realizován na mikropočítači Raspberry Pi 3 Model B+ v programovacím jazyku Python s pomocí knihoven OpenCv a NumPy.

Klíčová slova

Detekce objektů, Zpracování obrazu, Kalmanův filtr, Python, OpenCv, Raspberry Pi

Abstract

The aim of this work is to design a detector and predictor for a mechatronic model of table football. The theoretical part of this work presents methods for image processing, ball detection and finally motion prediction. In the testing part, the individual methods are tested and their outputs are shown. Finally, the system is implemented on a Raspberry Pi 3 Model B+ microcomputer, program is written in Python with the help of OpenCv and NumPy libraries.

Key words

Object detection, Image processing, Kalman filter, Python, OpenCv, Raspberry Pi

Úvod

Cílem práce je automatizace stolního fotbálku. Tato práce řeší dvě základní úlohy a to zpracování obrazu a predikci pohybu míčku. Kolega Martin Jandík se zabýval ve své práci řízením pohybu hráče. Pro úvodní experimenty byla úloha zjednodušena na detekci míče na zeleném pozadí a později otestovaná na modelu arény.

V teoretické části jsou nejprve v kapitole 2 představeny metody pro předzpracování obrazu, které jsou nezbytné pro zlepšení vlastností obrazu. Následně se zabýváme detekcí míčku v kapitole 3. Pro funkčního hráče musíme vědět, kde se bude v dalším časovém okamžiku nacházet míček, metody řešení predikce jsou uvedeny v kapitole 4.

Uvedené metody byly otestovány v kapitole 6 a jsou ukázány jejich jednotlivé výstupy. Realizace zařízení je popsána v kapitole 7, kde jsou uvedeny nezbytné hardwarové a softwarové prostředky k realizaci. V dané kapitole se také nachází zhodnocení jednotlivých vlastností vyzkoušených metod.

Obsah

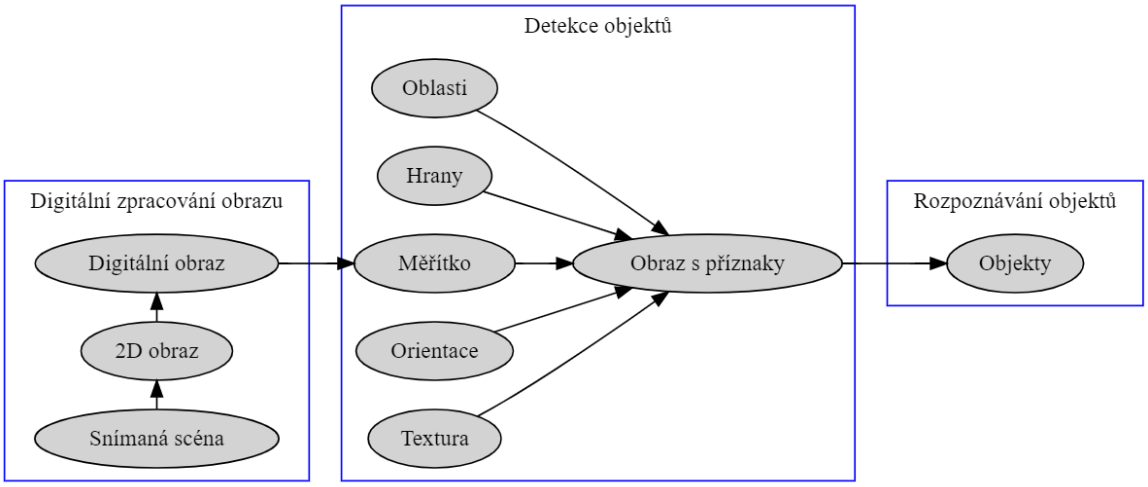
Seznam obrázků

Kapitola 1

Počítačové vidění

Počítačové vidění je technickým odvětvím, jež se zabývá zpracováním digitálního obrazu za účelem extrakce příznaků a informací, nebo též porozumění scény. Toto odvětví zažívá v posledních letech velký rozmach díky nárůstu výpočetního výkonu a tedy možnosti nasadit tyto systémy v reálném čase. Cílem těchto systémů je do jisté míry být srovnatelný s člověkem. V komerční praxi se počítačové vidění používá například v autonomních automobilech. V této práci jde o zpracování real-time videa a určení aktuální a budoucí polohy míčku.

Počítačové vidění dělíme na více fází, což ilustruje obrázek 1.1. Ve fázi zpracování digitálního obrazu pouze převádíme analogovou scénu do 2D digitální podoby. Až v detekci objektů dochází ke zpracování a hledání požadovaných příznaků. Ty máme ve fázi poslední a výstupem je obraz s příznaky nebo už pouze jeho části v podobě hledaných objektů.



Obrázek 1.1: Schéma zpracování digitálního obrazu [14]

1.1 Omezení zpracování obrazu

Na omezení narážíme v každé části, jak získávání tak zpracování digitálního obrazu, které ovlivňují kvalitu získávání požadovaných příznaků. Omezení je celá řada, a tak při aplikaci konkrétního řešení musíme zohlednit vnější vlivy, daná omezení a najít kompromis. Nejčastěji se vyskytující omezení jsou uvedena v následujících podkapitolách.

1.1.1 Ztráta informace ve 2D obrazu

Při pořízení obrazu standardním způsobem, tedy digitálním fotoaparátem nebo kamerou, zaniká informace o hloubce a tedy i měřítku. Z 2D obrazu nelze zjistit vzdálenost ani velikost fotografovaných objektů, ukázka je na obrázku 1.2. V mnoha případech je však zjednodušení do 2D ulehčením, jelikož je informace o hloubce nadbytečná, například u zpracování psaných textů [14].

Obrázek 1.2: Ilustrace ztráty informace ve 2D obrazu, nelze určit poloměr kruhu bez reference

1.1.2 Šum

Šum je všudypřítomný a nikdy se ho úplně nezbavíme. Lze jej však zmírnit výběrem vhodného snímače. Některá prostředí jsou náchylnější k zašumění obrazu, příkladem jsou fotografie pořízené za špatných světelných podmínek. Šum se nedá snadno matematicky popsat a většinou se tedy uvažuje bílý šum [14]. Metody k potlačení šumu zahrnují vyhlazování uvedené v kapitole 2.1.

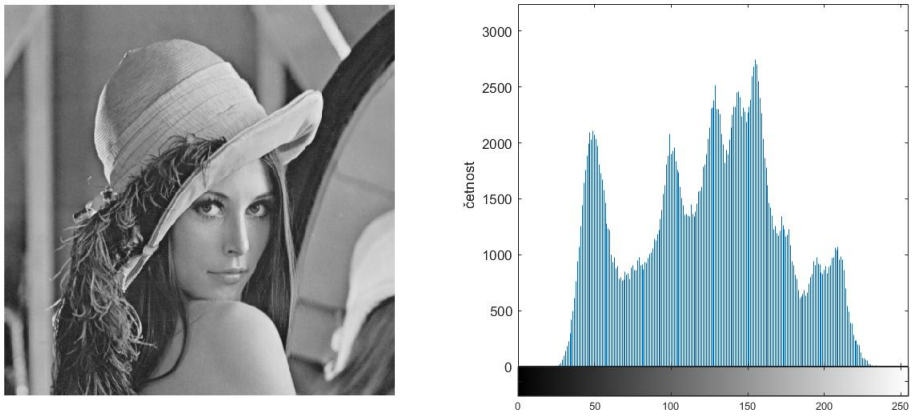
1.1.3 Velikost dat

Velikost dat je omezující nejčastěji při požadavku na zpracování v reálném čase, nebo při malém výpočetním výkonu. Velikost dat tedy souvisí s dostupným rozlišením. Pokud chceme hledat drobné objekty, nebo detaily v objektech větších, potřebujeme obraz s vysokým rozlišením. Musíme se smířit s nižším počtem zpracovaných obrazů za jednotku času. Pokud

nám jde o rychlost, můžeme snížit rozlišení a zvýšit tím rychlost detekčního algoritmu, avšak detekované předměty musí být dostatečně veliké a ztratíme informaci o jejich detailech.

1.1.4 Jas

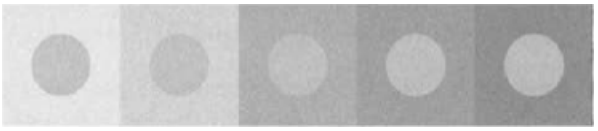
Jas obrazu je závislý na množství okolních vlivů, jimiž jsou například počet zdrojů světla, jejich typ,intenzita a úhel natočení. Také záleží na prostředí, konkrétně na rozložení scény a světelných vlastnostech materiálů ve scéně. Jas se nejčastěji vyjadřuje pomocí histogramu, ukázka snímku a jeho histogramu je na obrázku 1.3 .



Obrázek 1.3: Příklad histogramu daného obrázku [11]

1.1.5 Kontrast

Kontrast se definuje jako poměr mezi střední hodnotou jasu objektu a střední hodnotou pozadí. Pro úspěšnou detekci je lepší vysoký kontrast. Příklad je na obrázku 1.4, kde vidíme podmíněný efekt kontrastu mezi objektem a pozadím. Jedná se o iluzi, které podlehne lidské oko, jelikož vnímá jas logaritmicky. Lidské oko vnímá jas kruhů na různém pozadí odlišně, přestože je pořád stejný.



Obrázek 1.4: Ukázka podmíněného efektu kontrastu objektu s pozadím [14]

Kapitola 2

Obecné algoritmy předzpracování obrazu

Cílem předzpracování digitálního obrazu je zlepšení parametrů obrazu a ulehčení lokalizace užitečných informací. Parametry jsou například šum, zkreslení a rysy obrazu. Výběr parametrů, které chceme potlačit nebo zvýraznit určují potřeby aplikace.

2.1 Pixelové operace

Vyhlazování je jednou z nejpoužívanějších technik zpracování digitálního obrazu. Jedná se o pixelovou operaci, slouží k odstranění šumu a vyhlazení hran objektů.

$$g(x,y) = \omega * f(x,y) = \sum_{dx=0}^{a} \sum_{dy=0}^{b} \omega(dx,dy) f(x+dx, y+dy) \tag{2.1}$$

Kde $g(x,y)$ je nová hodnota pixelu po konvoluci, ω je jádro vyhlazení, $f(x,y)$ je vstupní obraz, a/b jsou velikosti jádra.

2.1.1 Průměrové vyhlazování

Průměrové vyhlazení funguje pomocí aritmetického průměru. Všechny prvky jádra mají hodnotu 1 a tudíž všechny prvky mají stejnou váhu. Výsledný obraz vzniká konvolucí jádra a obrazu. Operace odstraňuje bílý šum, tedy i drobné detaily obrazu. Příklad použití je na obrázku 2.1. Jádro průměrování:

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Obrázek 2.1: Příklad použití průměrového vyhlazování jádrem 21x21 [11]

2.1.2 Gaussovské vyhlazování

Vyhlazování používá místo konstantních prvků tzv. Gaussovo jádro [3]. Funkce je definována jako:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.2}$$

Je nutné specifikovat velikost jádra, nejčastěji se používá 5x5 [3]. Gaussovské vyhlazování se dá použít dvěma způsoby. Jednak jako rozostření a také jako doostření. Jak je vidět na obrázku 2.2, je toto rozostření vhodné pro potlačení Gaussovského šumu. Jelikož rozostření potlačuje vysoké frekvence, můžeme využít této skutečnosti, k originálnímu obrazu přičteme násobek rozdílu originálního obrazu a rozmazaného dle rovnice 2.3. Kde v rovnici parametr α určuje intenzitu doostření. Tím nám vznikne doostřený obrázek dle literatury [15].

$$g_{doostr} = f + \alpha(f - \omega * f) \tag{2.3}$$

Obrázek 2.2: Příklad použití Gaussovského rozostření jádrem o velikosti 5x5 [11]

2.1.3 Mediánové vyhlazování

Mediánové vyhlazování rozdělí vzestupně seřazené prvky v jádře na dvě stejně početné poloviny. Jelikož je počet prvků jader vždy lichý nemusíme řešit případ sudého počtu prvků. Mediánové vyhlazování je vhodné v momentě, kdy se v jádře vyskytují extrémní hodnoty, jelikož je na rozdíl od vyhlazování průměrováním ignoruje. Příklad použití je na obrázku 2.3. Nastínění funkčnosti pro konkrétní hodnoty jádra:

$$\begin{bmatrix} 2 & 2 & 2 \\ 1 & 5 & 1 \\ 4 & 150 & 20 \end{bmatrix} => [1, 1, 2, 2, \mathbf{2}, 4, 5, 20, 150] => \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$$

Obrázek 2.3: Příklad použití Mediánového vyhlazování [11]

2.1.4 Prahování

Prahováním dostaneme oblast zájmu u pixelů, které mají hodnotu nad, pod, nebo mezi pevně stanovenými hranicemi. Výstupem prahování je binární obraz. Prahování je definováno následovně

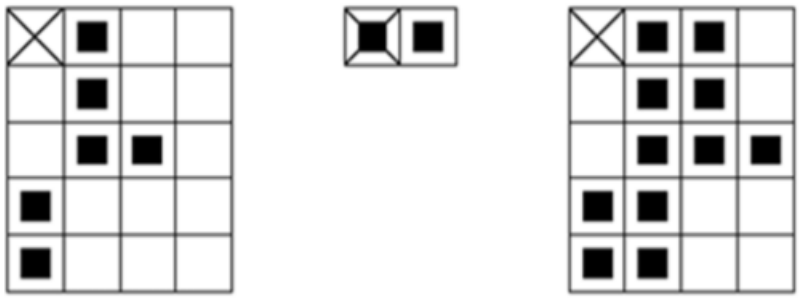
$$f(x) = \begin{cases} 1 & \text{pokud } c < \mathit{práh} \\ 0 & \text{pokud } c \geq \mathit{práh} \end{cases} \tag{2.4}$$

Cílem prahování je oddělit objekt zájmu od pozadí, v našem případě míč a připravit tak obraz na další zpracování. Nevýhodou prahování je pevně daná hranice, tedy objekt musí být v kontrastu s pozadím, aby byl správně určen. Prahování se používá v kombinaci s vyhlazováním pro odstranění šumu a zlepšení výsledků, ukázka na obrázku 2.4. Pro další zlepšení po převodu do binárního obrazu se používá morfologických operací uvedených v kapitole 2.2.

Obrázek 2.4: Ukázka prahování [11]

2.2 Morfologické operace

Morfologické operace jsou operace nad skupinou bodů a ,nejčastěji v binárním obrazu. Tyto operace upravují tvar skupin, zbavují je šumu ať už aditivního nebo subtraktivního. Nový obraz vzniká systematickým posunem jádra zleva doprava, po řádcích v originálním binárním obrazu, jak je vidět na ukázkovém příkladu dilatace 2.5.



Obrázek 2.5: Ukázka funkce jádra

2.2.1 Eroze

Eroze je subtraktivní operace nad skupinou bodů. Výsledkem je že objekt zeštíhluje a snižuje tak jeho celkovou plochu, ukázka na obrázku 2.6. Eroze odstraňuje šum a drobné objekty. Předpis eroze:

$$A \ominus B = \cup_{n \in B}^{b} X_b \tag{2.5}$$

Obrázek 2.6: Ukázka eroze jádrem 5x5 [11]

2.2.2 Dilatace

Dilatace je aditivní operace nad skupinou bodů. Výsledkem je zvětšení plochy všech objektů na úkor pozadí. Používá se k zaplnění děr, popřípadě zálivů ukázka na obrázku 2.7. Předpis dilatace:

$$A \oplus B = \cap_{n \in B}^{b} X_b \tag{2.6}$$

Obrázek 2.7: Ukázka dilatace jádrem 5x5 [11]

2.2.3 Otevření

Otevření je eroze následovaná dilatací. Většinou jedna iterace nestačí a tak se používá několikanásobné použití eroze následované stejným počtem dilatací. Otevření umožní odstranit šum a drobné objekty v pozadí.

2.2.4 Uzavření

Uzavření je dilatace následovaná erozí. Stejně jako u otevření se používá několik dilatací následovaných stejným počtem erozí. Uzavření umožňuje vyplnit malé díry v objektu v popředí.

2.3 Hledání hran

Hledání hran je velmi důležité předzpracování obrazu, využívá se ke sledování změn citlivostní funkce. Hranu tvoří pixely ve kterých se citlivostní funkce strmě mění. Jelikož jsou hrany do určité míry neměnné vůči osvětlení a úhlu pohledu, pokud bereme v potaz oblasti s největší změnou, pak nám takováto informace postačí pro základní porozumění obrazu.[1]

Jeden bod v hraně je vektor o dvou proměnných, těmi jsou velikost a úhel. Velikost je velikost gradientu a směr je rotován o -90°vůči původnímu směru gradientu. Velikost a směr gradientu se ve spojitém obrazu spočítají následovně:

$$|grad\, g(x,y)| = \sqrt{\left(\frac{\partial g}{\partial x}\right)^2 + \left(\frac{\partial g}{\partial y}\right)^2} \tag{2.7}$$

$$\phi = arg\left(\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}\right) \tag{2.8}$$

Kde arg(x,y) je úhel v radiánech od osy x k bodu (x,y). V některých případech nás zajímá pouze velikost výchylky bez jejich orientace. Uvedeme si základní výčet příkladů gradientních operátorů. Jelikož je digitální obraz z podstaty diskrétní, jedná se o konvoluci operátorů a obrazu.

Následující operátory nám vrátí všechny hrany a to i včetně těch nejmenších. Proto se musíme také zabývat kvalitou hrany. V jednoduchých případech lze všechny hrany prahovat a získat tak binární obraz obsahující významné hrany. V pokročilejších aplikacích se využívá prahování s hysterezí.

2.3.1 Robertsův operátor

Robertsův operátor patří mezi nejstarší. Má velmi malé výpočetní nároky díky jádru o velikosti 2x2.

$$k_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, k_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Velikost výchylky se tedy spočte jako:

$$|g(x,y) - g(x+1,y+1)| + |g(x+1,y) - g(x,y+1)| \tag{2.9}$$

Hlavní nevýhodou Robertsova operátoru je jeho náchylnost na šum, jelikož používá malé okolí pixelu na aproximaci odchylky.

2.3.2 Laplaceův operátor

Laplaceův operátor je velmi oblíbený, jelikož jsou jeho vlastnosti ve všech směrech stejné, je neměnný vůči natočení. Ve spojité formě je definován takto:

$$\nabla^2 g(x,y) = \frac{\partial^2 g(x,y)}{\partial x^2}, \frac{\partial^2 g(x,y)}{\partial y^2} \tag{2.10}$$

V diskrétní podobě je aproximován konvoluční sumou. Laplaceův operátor vrací pouze velikost výchylky. Pro jádro k 3x3 se nejčastěji používá čtyřokolí a osmiokolí definované jako

$$k_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, k_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

2.3.3 Sobelův operátor

Sobelův operátor se nejčastěji používá jako detektor horizontálních a vertikálních hran. Pro detekci těchto hran se používají jádra k_1 a k_2 . Berme odezvu na jádro k_1 jako x a odezvu na jádro k_2 jako y. V tomto případě spočteme výchylku následovně:

$$\sqrt{x^2 + y^2} \quad nebo \quad |x| + |y| \tag{2.11}$$

směr se spočte jako $arctan(y/x)$

$$k_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, k_2 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}, k_3 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & & 1 \end{bmatrix}, \cdots$$

2.3.4 Cannyho hranový detektor

Cannyho hranový detektor má tři hlavní kritéria, a těmi jsou

- Kritérium detekce, které říká, že důležité hrany by neměly být vynechány a dále by neměly existovat falešné hrany.
- Kritérium lokalizace, které říká, že vzdálenost mezi skutečnou a detekovanou hranou by měla být co nejmenší.
- Každá hrana by měla být detekována pouze jednou.

Nyní víme jaká máme kritéria a následuje výčet jednotlivých kroků.

1. Eliminace šumu
   K potlačení šumu se dají použít medoty z kapitoly o Pixelových operacích 2.1 nejčastěji se používá Gaussovo filtru, kapitola 2.1.2 .
2. Určení gradientu
   K určení gradientu se používají výše zmíněné operátory. Nejobvyklejší volbou je Sobelův operátor 2.3.3.

3. Nalezení maxim
   Po určení gradientu máme stále rozmazané hrany, a pro splnění posledního kritéria musíme hrany ztenčit. To se provede algoritmem nonmaximum suppression. Algorimus projde okolí zkoumaného pixelu a pokud v sousedních pixelech po a proti směru gradientu je vyšší hodnota, vynuluje zkoumaný pixel. V opačném případě hodnotu ponechá.

4. Prahování
   Používá se dvojité prahování k eliminaci nepotřebných hran, kde máme prahy T_H a T_L.

   - hodnota hrany$>T_H$ je hrana uznána a označena jako silná
   - hodnota hrany$<T_L$ je smazána
   - hodnota hrany v intervalu $<T_L;T_H>$ je hrana označena jako slabá a je uznána pouze pokud je v jejím okolí silná hrana

Cannyho detektor hran lze dále vylepšit, lepší filtrací šumu, nebo použitím různých hranových operátorů např. Prewittův, Scharrfův. Podrobnosti o zlepšení funkčnosti lze najít v literatuře [14]

Kapitola 3

Detekce

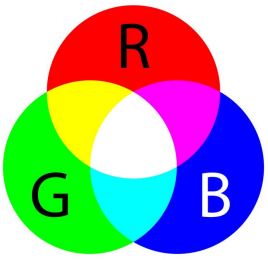
V rámci detekce představíme pár základních způsobů jak lze najít objekt se specifikovanými vlastnostmi v obraze. Mezi takovéto vlastnosti patří barva, velikost, obrazová předloha.

3.1 Detekce podle barvy

Pokud se rozhodneme detekovat předmět na základě barvy, je na naší volbě jakou reprezentaci barev si vybereme. Vstupem je tedy obraz a výstup je skupina bodů spadající do vymezeného barevného prostoru. Nejpopulárnější volby jsou následující.

3.1.1 RGB

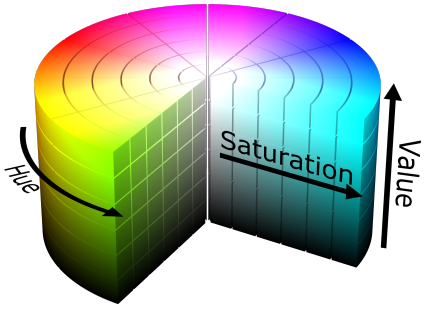
RGB je standardní barevný model pro ukládání a zobrazování obrázků v digitální podobě. Jeho předností je, že se dá transformovat do dalších barevných modelů. Jedná se o model jehož základními složkami jsou barvy červená, zelená a modrá. Všechny ostatní barvy se dají vytvořit jejich kombinací. RGB se nejčastěji používá v monitorech a dalších zobrazovacích zařízeních. Použitím pro detekci brání obtížná definovatelnost rozmezí barev. Obraz je tvořen aditivně jak je vidět na obrázku 3.1.



Obrázek 3.1: Zobrazení RGB barevného modelu [16]

3.1.2 HSV

HSV barevný model je ve zpracování obrazu častější. Zkratka značí Hue, Saturation, Value (odstín, sytost, jas) [9]. Mezi přednosti HSV modelu patří snadné definování rozsahu hledaných barev. Také je blíže lidskému vnímání barev a používají ho některé grafické editory[6]. Zobrazení spektra v HSV je na obrázku 3.2.



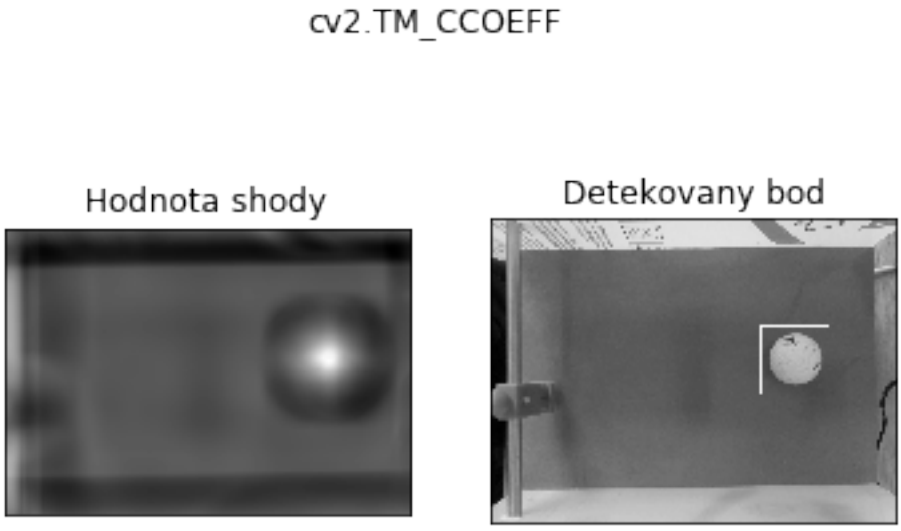
Obrázek 3.2: Zobrazení HSV barevného modelu [10]

3.2 Detekce podle plochy

Za předpokladu, že máme již vybrané oblasti zájmu (například prahováním), můžeme u těchto oblastí měřit jejich obsah, nebo pouze v našem případě poloměr a ten porovnat s referenčním měřením. Tato metoda je velmi neefektivní a náchylná k šumu. V obecném případě nastává problém, kdy může být v obrazu více velikostně shodných objektů. Metoda byla použita pouze v prvotních pokusech a dále se jí nebudeme zabývat.

3.3 Detekce podle pattern matchingu

Pattern matching je metoda hledání umístění menšího vzoru v obrazu. Tato metoda probíhá nad vstupním obrazem a požadovaným vzorem. Cílem metody je nalézt menší vzor ve větším. Hledání probíhá posunem vzoru po obrazu, tím je myšleno vždy o pixel z levého horního do pravého dolního rohu. V každém kroku se vypočte hodnota na základě zvoleného vyhodnocovacího kritéria. Výstupem je tedy matice s výsledky, ve které hledáme maximum, popřípadě minimum v závislosti na zvoleném kritériu. Pokud se ve vstupním obrazu nachází více vzorů, musíme nastavit práh. Ukázka funkčnosti a výstupu je na obrázku 3.3.



Obrázek 3.3: Ukázka funkce pattern matchingu

3.3.1 Vyhodnocovací kritéria

V literatuře [4] jsou uvedena následující kritéria s jejich předpisy. Kritéria určují způsob výpočtu shody vzoru (T) a daného výřezu (I) vstupního obrazu na souřadnicích x, y. U kritérií hledáme maximum ve výstupu kromě TM-SQDIFF, kde hledáme minimum. Ke všem uvedeným metodám existuje i normovaná verze, která omezuje hodnoty na interval 0-1. Normované verze jsou vhodnější pro použití, jelikož se dá snáze definovat míra shody [7].

1. TM-SQDIFF (Template Matching Square Difference)

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2 \tag{3.1}$$

2. TM-SQDIFF-NORMED (Template Matching Square Difference Normed)

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} (T(x',y'))^2 \sum_{x',y'} (I(x+x',y+y'))^2}} \tag{3.2}$$

3. TM-CCORR (Template Matching Cross Correlation)

$$R(x,y) = \sum_{x',y'} (T(x',y') * (x+x',y+y')) \tag{3.3}$$

4. TM-CCOEFF (Template Matching Correlation Coefficient)

$$R(x,y) = \sum_{x',y'} (T'(x',y') - I'(x+x',y+y')) \tag{3.4}$$

$$T'(x',y') = T(x',y') - 1/(w*h) * \sum_{x'',y''} T(x'',y'') \tag{3.5}$$

$$I'(x',y') = I(x+x',y+y') - 1/(w*h) * \sum_{x'',y''} I(x+x'',y+y'') \tag{3.6}$$

5. TM-CCOEF-NORMED (Template Matching Correlation Coefficient Normed)

$$R(x,y) = \frac{\sum_{x',y'}(T'(x',y') - I'(x+x',y+y'))}{\sqrt{\sum_{x',y'}(T'(x',y'))^2 \sum_{x',y'}(I'(x+x',y+y'))^2}} \tag{3.7}$$

Kapitola 4

Predikce

V této kapitole se budeme zabývat způsoby, jak lze zjistit stav systému v nastávajících časových okamžicích.

4.1 Tvorba modelu

Model je matematický popis systému, který simuluje chování reálné soustavy [5]. Náš systém se zabývá pohybem míčku a jeho výstupem je pozice. Při tvorbě modelu systému máme dvě hlavní možnosti, a to modelovat na základě znalosti dynamiky, nebo vytvořit generický model polynomiálního signálu. Vzhledem k tomu, že dynamika reálného systému je složitá a v této práci se zabýváme pouze zjednodušeným modelem, můžeme pro simulační účely vytvořit obecný systém s konstantním zrychlením nebo s konstantní rychlostí. Druhou možností a jejími variantami se zabýváme v kapitole 4.3. K reprezentaci modelu systému a vyzkoušení Kalmanova filtru jsem si vybral stavový popis systému. Schéma stavového popisu je na obrázku 4.1. Stavová rovnice obecného diskrétního v čase neměnného systému je následující.

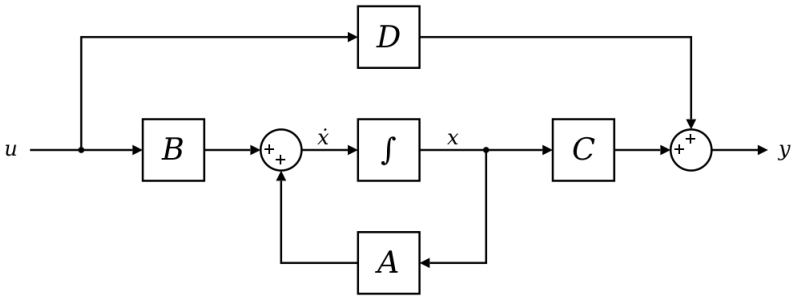
$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned} \tag{4.1}$$

Jelikož se míč pohybuje po parabole, volil jsem pro testování v Matlabu systém s konstantním zrychlením. Stavová reprezentace je následující: matice A je matice stavů, matice B je nulová, jelikož nepůsobíme žádným vstupem, C je matice výstupů a D je přímé působení vstupu na výstup také nulové.

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} B = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} D = 0$$

Po diskretizaci s periodou rovnou vzorkování kamery 0,03s vzniká diskrétní stavová reprezentace, kde se změny projeví pouze v matici A.

$$A = \begin{bmatrix} 1 & 0.03 & 4.5e^{-4} \\ 0 & 1 & 0.03 \\ 0 & 0 & 1 \end{bmatrix} B = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} D = 0$$



Obrázek 4.1: Schéma stavového popisu

4.2 Kalmanův filtr

4.2.1 Matematická formulace

Kalmanův filtr je pojmenován po Rudolfu E. Kalmanovi, který v roce 1960 vydal odborný článek [8] zabývající se rekurzivním řešením problému o odhadu diskrétních dat. Kalmanův filtr v literatuře [5] je sada matematických rovnic implementujících predikci a korekci, ta je optimální ve smyslu minimalizace odhadované chyby, pokud jsou splněny požadované podmínky. Kalmanův filtr je v posledních letech předmětem mnoha vědeckých prací. Nejčastější využití v praxi nachází v autonomní nebo asistované navigaci a to díky rekurzivní povaze výpočtu.
Kalmanův filtr odhaduje stav $x \in R^n$ diskrétního procesu, ten je určen rovnicí:

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \tag{4.2}$$

a měřením $z \in R^m$ které je definováno:

$$z_k = Hx_{k-} + v_k \tag{4.3}$$

Náhodné proměnné w a v reprezentují šumy, jejich rozptyly Q a R nám určují věrohodnost modelu a měření. Předpokládáme, že jsou na sobě vzájemně nezávislé a mají normální rozdělení a nulovou střední hodnotu.

$$p(w) \sim N(0, Q) \tag{4.4}$$

$$p(v) \sim N(0, R) \tag{4.5}$$

4.2.2 Algoritmus

V algoritmu se střídají 2 funkce ***predikce*** a ***korekce***. Predikce se skládá ze dvou rovnic:

$$\hat{x}_k^- = A\hat{x}_{k-1}^- + Bu_{k-1} \tag{4.6}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{4.7}$$

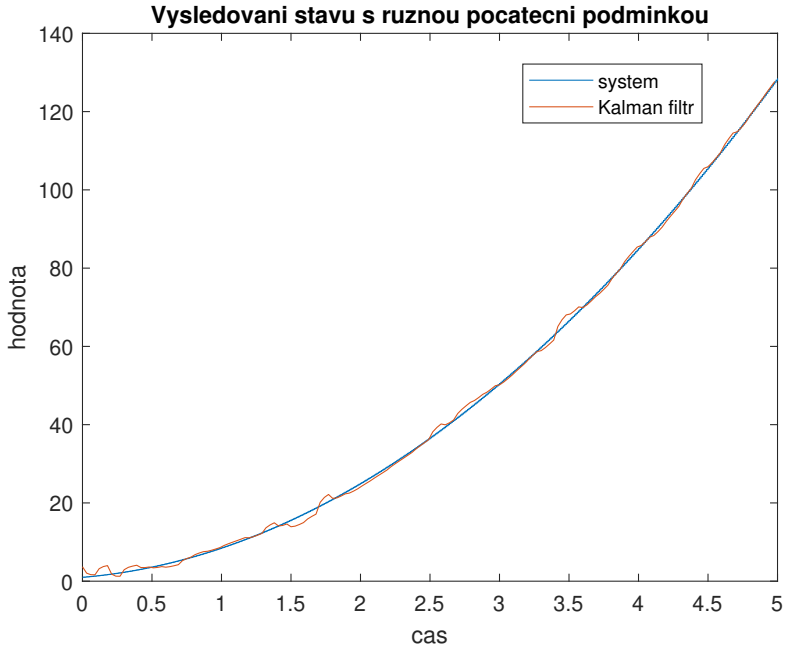
Spočteme odhad $\hat{x}_k^-$ z předchozího kroku. Matice A a B jsou ze stavového popisu systému. Q je rozptyl z normálního rozložení. Korekce má rovnice následující

$$K_k = P_k^- H^T (HP_k^- H^T + R) \tag{4.8}$$

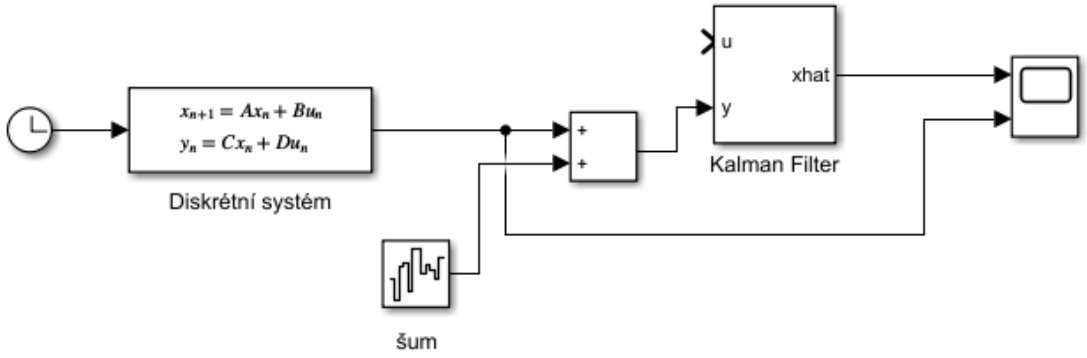
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{4.9}$$

$$P_k = (I - K_k H)P_k^- \tag{4.10}$$

Jako první vypočteme optimální zesílení/zeslabení K_k. Další krok je změření z_k a vypočtení aposteriorního odhadu stavu za pomocí rezidua měření a odhadovaného stavu. V poslední rovnici vypočteme aposteriorní odhad kovariance chyby. Ukázka z testování s různými počátečními podmínkami pro vysledování stavu na obrázku 4.2 z Matlabu při schématu 4.3.



Obrázek 4.2: Vysledování stavu Kalmanovým filtrem



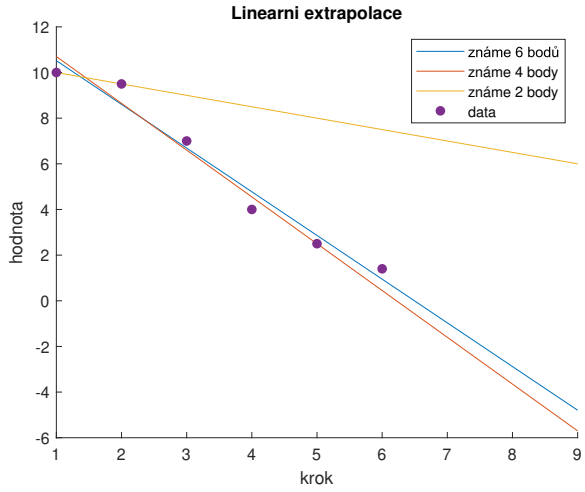
Obrázek 4.3: Schéma bloků při testování v Matlabu

4.3 Odhad budoucí polohy

Extrapolace je způsob, kterým lze získat přibližný budoucí stav proměnné na základě předchozího pozorování za předpokladu, že známe předpis dané funkce. Máme tedy k dispozici n-prvkový soubor měření. Požadujeme hodnotu stavu za k-kroků. Jelikož pro naší úlohu nemáme matematický předpis, aproximujeme tedy body polynomem. Obecně platí, že čím vyšší stupeň polynomu, tím přesnější odhad.

4.3.1 Lineární extrapolace

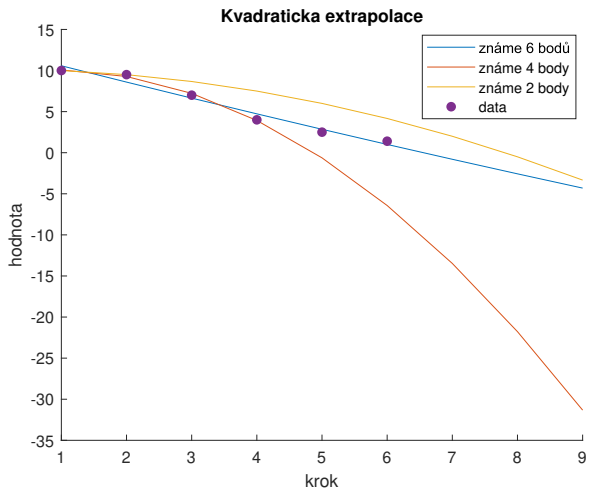
Nejjednodušší je lineární extrapolace, kde dané body prokládáme přímkou, ukázka s různým počtem bodů je na obrázku 4.4. Pokud jsou v souboru více než 2 body výsledná přímka je vypočtena metodou nejmenších čtverců. Na obrázku je vidět, že použitím lineární extrapolace se v odhadu dopouštíme velkých chyb, avšak je jednoduchá na výpočet.



Obrázek 4.4: Ukázka lineární extrapolace

4.3.2 Kvadratická extrapolace

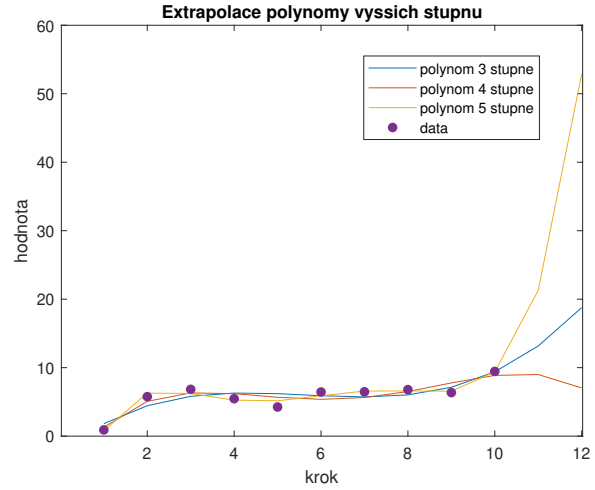
Kvadratickou extrapolací, jak je vidět na obrázku 4.5, jsme docílili mnohem lepších výsledků než v předchozím případě. Stejně jako u lineární extrapolace: nachází-li se v souboru více bodů, než je nezbytné pro definování paraboly, je cílová parabola vypočtena pomocí metody nejmenších čtverců.



Obrázek 4.5: Ukázka kvadratické extrapolace

4.3.3 Extrapolace n-tým polynomem

U polynomů vyšších řádů se postupuje obdobně jako u předchozích případů. Ukázka extrapolace vyššími polynomy viz. obrázku 4.6. U vyšších polynomů se chyba stále zmenšuje, ale narůstá výpočetní složitost. Otázkou je, jaký stupeň nám postačí pro věrnou predikci. U polynomů vysokých řádů je třeba věnovat pozornost případu, kdy máme polynom vyššího řádu než je počet dostupných měření. Polynom bude přesně procházet všemi body, ale nedojde k zobecnění a zachycení trendu.

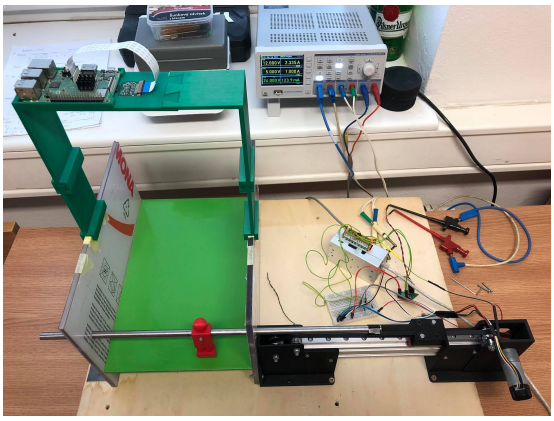


Obrázek 4.6: Ukázka extrapolace polynomy vyšších stupňů

Kapitola 5

Konstrukce

Aréna je zhotovena z plastových desek o tloušťce 8mm. Na konstrukci arény jsem spolupracoval s kolegou Martinem Jandíkem. Konstrukce arény je připevněna na základové desce ze dřeva, čímž je eliminováno možné prohýbání, či jiné deformace arény.



Obrázek 5.1: Kompletní konstrukce arény

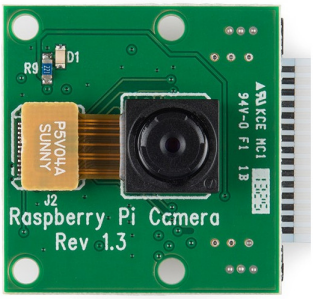
5.1 Snímání obrazu

Ke snímání obrazu jsem využil kameru pro Raspberry Pi obrázek 5.2. Parametry kamery jsou následující

- rozlišení snímku: 2592 x 1944
- rozlišení videa: 1080p při 30fps, 720p při 60fps
- velikost: 20 x 25 x 9mm
- váha: 3g

Nejdůležitějšími parametry jsou FPS a rozlišení. FPS určuje počet snímků za vteřinu (z anglického frames per second), což omezuje maximální možnou rychlost míčku pro funkční

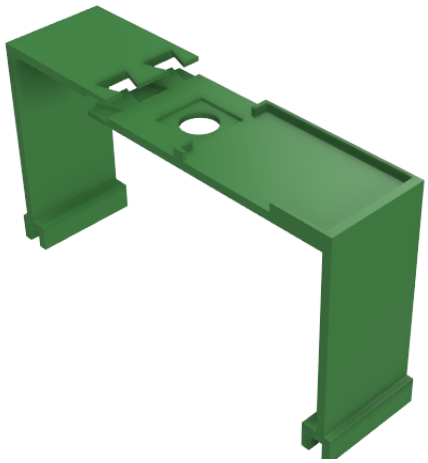
detekci a predikci. Dle Shannonova teorému [2] musí být vzorkovací frekvence 2x rychlejší než maximální frekvence změn v systému, aby nedocházelo k antialiasingu. Na rozlišení je závislý výpočetní čas. Většina algoritmů prochází celý obraz, je zde tedy výpočetní složitost N^2 nebo vyšší.



Obrázek 5.2: Kamera pro Raspberry Pi [12]

5.2 Umístění kamery

Optimální umístění kamery jsem získal experimentálním měřením. Nejlepší umístění vyšlo ve výšce 38cm. Pro její ukotvení jsem v programu AutoCAD vytvořil 3D model držáku. Je zhotovený ze 2 částí, které se v horní části spojují jako puzzle. Na držáku je vytvořeno místo pro ukotvení RaspberryPi a výřez s prohlubní pro kameru.



Obrázek 5.3: 3D model držáku kamery

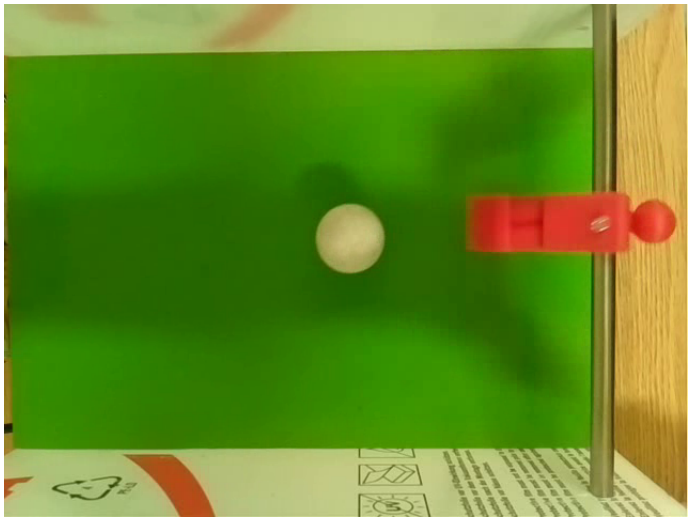
Kapitola 6

Testování

V této kapitole vyzkoušíme představené metody pro zpracování obrazu, detekci a predikci v praxi. Testování jsem nejprve provedl na statických snímcích. Poté jsem si natočil videa s pohybem míčku pro offline testování.

6.1 Detekce

Všechny provedené metody jsem testoval na vzorovém snímku 6.1.



Obrázek 6.1: Snímek pro testování metod

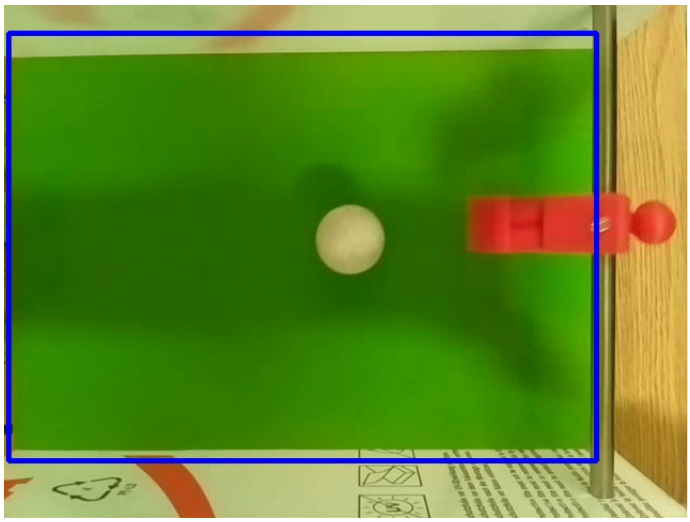
6.1.1 Výsledky prahování

Vstupní obraz převádíme z formátu RGB do formátu HSV pro snazší zadání mezí. Vstupní obraz prahujeme na základě námi zvolených mezních hodnot a dostáváme binární výstupní obraz viz. obrázek 6.2.

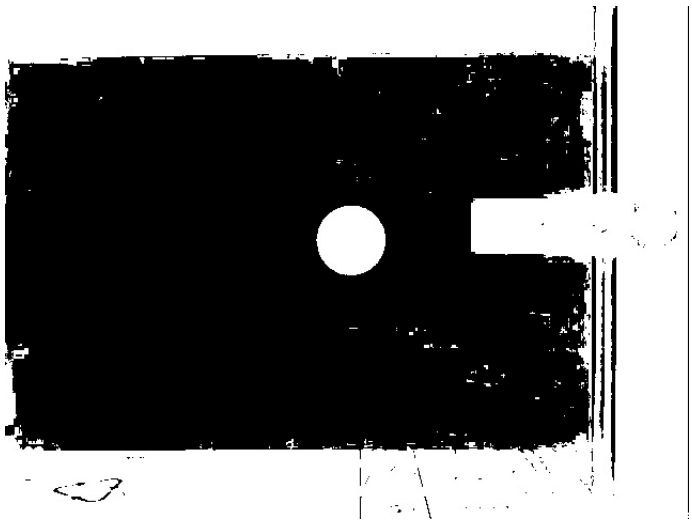
Obrázek 6.2: Ukázka prahování pro vzorový snímek

6.1.2 Filtrace podle barvy

Vstupní obraz ve formátu RGB převedeme do formátu HSV. Ve formátu HSV je snazší zadat rozpětí požadované barvy. Na obrázku 6.3 jsem vykreslil kontury ohraničující hřiště. Ty nám definují hrací plochu. Účelně jsem nepoužil žádné další metody ke zlepšení výsledků, je tedy vidět že ohraničení je větší než skutečná plocha, což je zapříčiněno zeleným odrazem stranách hřiště. Na obrázku 6.4 jsem nechal binární obraz, je zde dobře vidět filtrovaný míček a šum.



Obrázek 6.3: Ukázka filtrace pro zelenou plochu hřiště

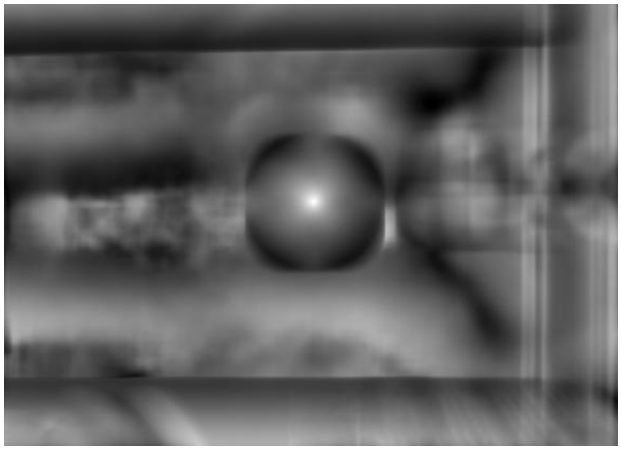


Obrázek 6.4: Ukázka filtrace míče

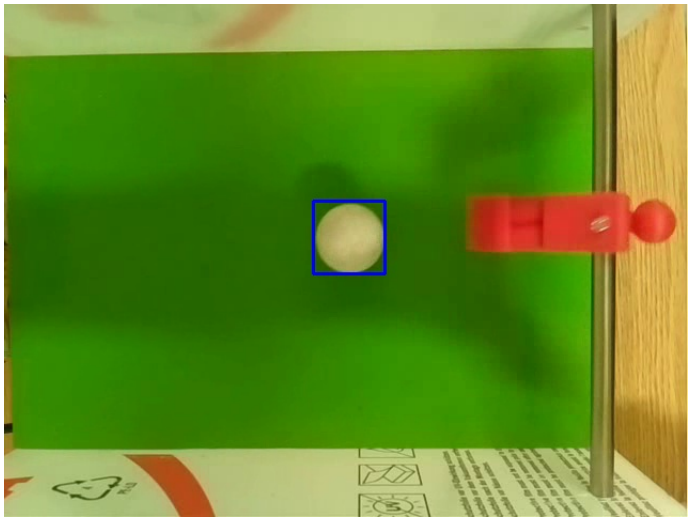
6.1.3 Detekce podle pattern matchingu

V pattern matchingu jsem používal pouze normované metody, jelikož mají pevně daný rozsah výstupní hodnoty 0-1. Výstupní hodnota je tedy rovna procentuální míře shody míčku. Na obrázku 6.5 vidíme vzor a na obrazu 6.6 výstupní hodnoty metody pro celý obraz, kde světlost odpovídá pravděpodobnosti výskytu míče. Po nalezení maxima máme souřadnice míčku. Interpretace výsledků je na obrázku 6.7. Kvalita detekce je přímo úměrná kvalitě vstupních dat. Pokud přijde silně zašuměný obraz může dojít k chybné identifikaci.

Obrázek 6.5: Vzor použitý pro pattern matching



Obrázek 6.6: Výstup pattern matchingu pro metodu TM-CCOEFF-NORMED



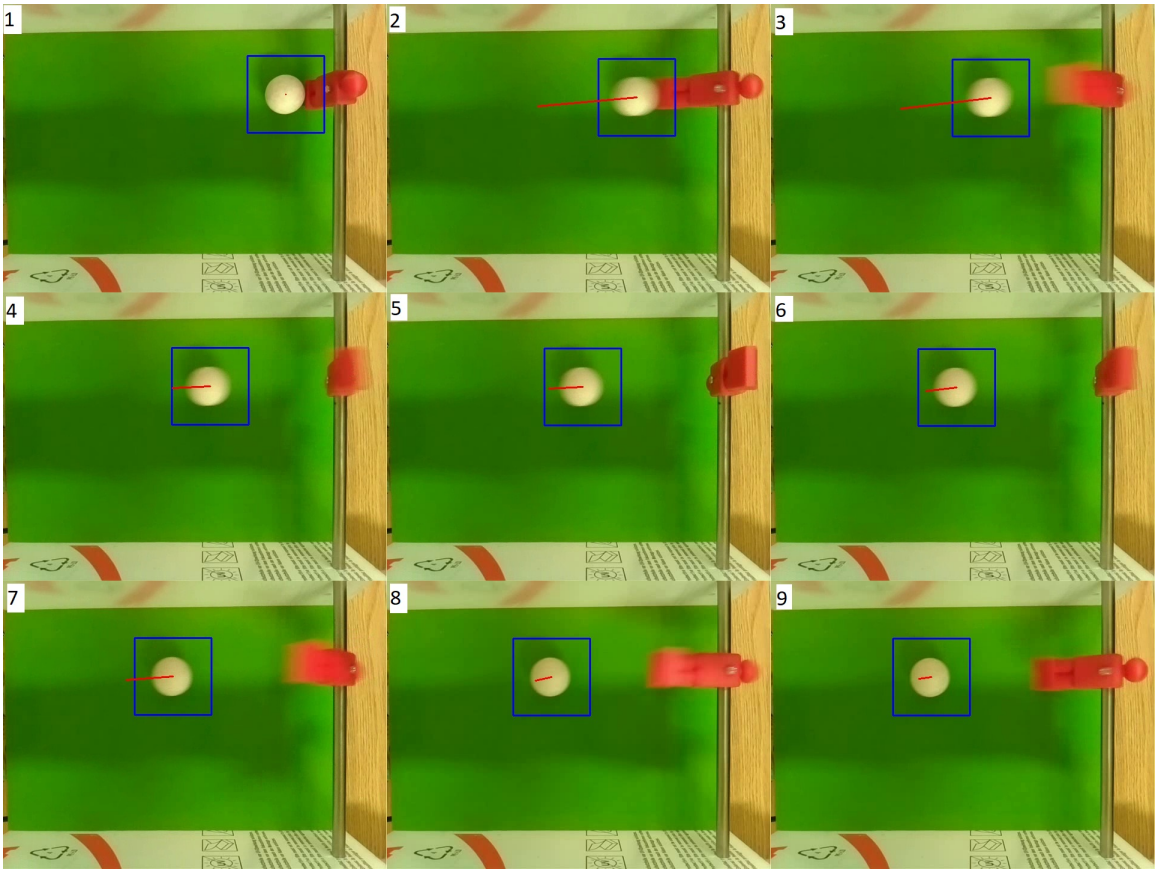
Obrázek 6.7: Interpretace výsledků

6.2 Predikce

Při testování predikce jsem se zabýval lineární predikcí a Kalmanovým filtrem. Funkčnost jsem demonstroval posloupností snímků.

6.2.1 Lineární predikce

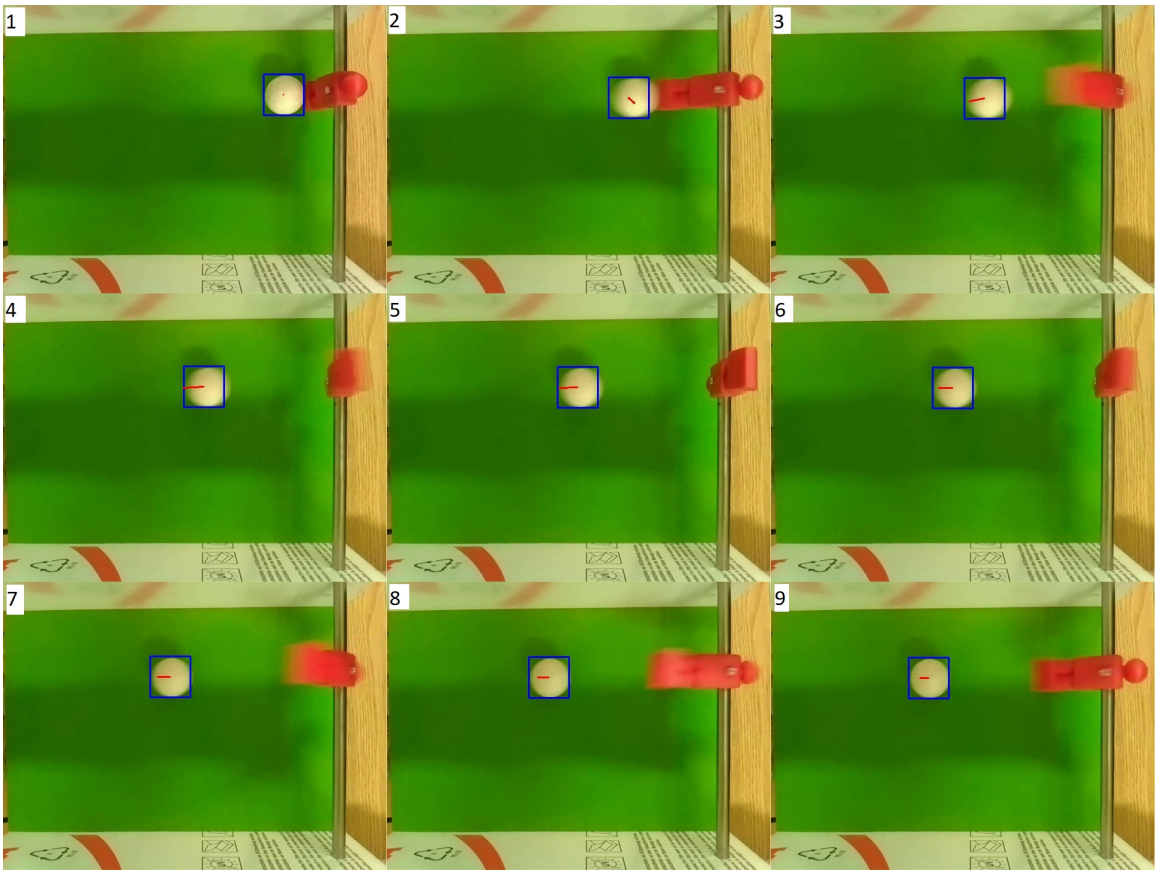
Lineární predikce je derivace dvou po sobě jdoucích snímků. Je tedy náchylná na prudké změny, jak je vidět u prvního vystřelení. Míček byl v sekvenci na obrázku 6.8 odpálen a následně ztrácel rychlost.



Obrázek 6.8: Sekvence snímků lineární predikce

6.2.2 Kalmanův filtr

Při implementaci Kalmanova filtru jsem volil model s konstantní rychlostí. Oproti lineární predikci je na obrázku 6.9 vidět pomalejší dynamika systému. Obě sekvence jsou shodné z nahraného videa.



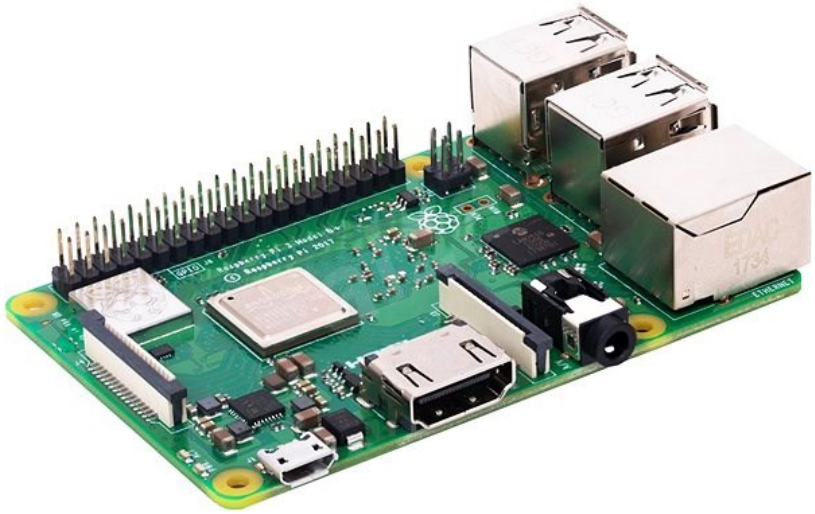
Obrázek 6.9: Sekvence snímků s použitím Kalmanova filtru

Kapitola 7

Praktická realizace

7.1 Hardware

K realizaci jsem si vybral Raspberry Pi 3 Model B+ na obrázku 7.1. Ke snímání obrazu byla použita kamera přímo určená pro Raspberry Pi parametry byly uvedeny v kapitole 5.1. Raspberry jsem si vybral z důvodu, že je to dobře dostupný mikropočítač s relativně vysokým výkonem, který je dostačující na zpracování obrazu.



Obrázek 7.1: Raspberry Pi 3 Model B+ [13]

Podstatné parametry Raspberry Pi 3 Model B+

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC 1.4GHz
- 1GB LPDDR2 SDRAM
- CSI port pro připojení Raspberry Pi camera
- Micro SD port pro načtení operačního systému a ukládání dat

- 5V/2.5A DC napájení

7.2 Software

Ze softwarové části jsem si vybral jazyk Python na verzi 3.6.5 a použil jsem knihovny OpenCv na verzi 3.4.2 a NumPy 1.14.3. Důvodem, proč jsem si vybral Python je to, že je dobře čitelný a objektově orientovaný a má mnoho knihoven podporujících zpracování obrazu.

7.2.1 Časová náročnost

Tabulka 7.1: Časová náročnost metod zpracování obrazu

Metoda	minimální čas	maximální čas	průměrný čas
šedotonové prahování	2.4ms	3.2ms	2.9ms
barevné prahování	2.7ms	4.5ms	3.1ms
pattern matching	10.5ms	15.5ms	14.1ms
detekce pomocí kontur	5.7ms	7.5ms	6.5ms

7.2.2 Zhodnocení metod detekce

Šedotonové prahování Šedotónové prahování je vhodným nástrojem pro zpracování obrazu, je-li mezi jednotlivými prvky velký kontrast. Podle pokusů je rychlejší než barevné prahování, ale jeho výsledky jsou mnohem více zašuměné. A výstup je náchylný na změny osvětlení v aréně.

Barevné prahování Barevné prahování je pomalejší než šedotónové, je kvalitnější a to je pro následující metody jako Hough circles a detekci pomocí kontur kritické. Náchylnost k chybám je u barevného prahování menší než u šedotónového. Pokud ovšem máme na ploše koncentrovaný bod světla promítne se do výstupu.

Pattern matching Pattern matching je nejrobustnější a nejpomalejší metoda, jelikož projede celý obraz a i za špatných světelných podmínek vykazuje stabilní výsledky. Jediným úskalím této metody je časová náročnost.

Detekce pomocí kontur Detekce pomocí kontur je metoda čistě závislá na vstupních datech. Pokud je ve vstupních datech více objektů, jsou nalezeny všechny a jediný faktor na rozpoznání míčku je obsah kontury. Tato metoda je rychlejší než pattern matching, ale její robustnost je nedostatečná pro praktické nasazení.

7.2.3 Zhodnocení metod predikce

Lineární predikce Lineární predikce je vhodná na jednoduché a pomalé pohyby míčku. Pokud dochází při pohybu míčku k prudkým změnám reaguje na ně predikce přehnaně, jak je vidět na obrázku 6.8.

Kalmanův filtr Kalmanův filtr podchycuje dynamiku systému, což bylo experimentálně ověřeno v kapitole 4.2, lépe a byl proto použit na cílové verzi.

7.2.4 Použité metody v cílové verzi

Cílovou verzi jsem směřoval do spolehlivosti na úkor výpočetního času. Příloha obsahuje všechny zmíněné metody a je možné je vyzkoušet. V rámci testování jsme zjistili, že metody jsou náchylné na vnější vlivy a fungují spolehlivě pokud nedojde k zásadním změnám v osvětlení. Jednotlivé kroky cílové verze jsou následující.

1. Vstupní obrázek je vyhlazen průměrovým vyhlazováním s jádrem 5x5
2. Dále je barevně vyprahován a je získán binární obraz
3. Na binárním obrazu se vyhledává vzor 6.5 pomocí pattern matchingu
4. Pokud je maximální hodnota větší jak 70% je nalezena pozice míčku
5. Fáze predikce a korekce (pokud byl nalezen míček) pro Kalmanův filtr, popsáno v kapitole 4.2
6. Kalmanův filtr vrací predikovanou pozici, ukázáno na obrázku 6.9

Kapitola 8

Závěr

Tato bakalářská práce se zabývala problematikou detekce míčku ve stolním fotbálku a predikcí jeho pohybu. Cílem tedy bylo vytvořit program, který zvládne tyto dvě úlohy.

V úvodu byla představena problematika počítačového vidění, dále metody pro zpracování obrazu. Ze zpracování obrazu se uplatňují metody vyhlazování a morfologické operace pro zlepšení vlastností vstupního snímku. V další části byly ukázány metody pro detekci, kde je nejvhodnější metoda pattern matchingu. Představeny byly také různé metody predikce, Kalmanův filtr se ukázal jako nejlepší varianta. Jednotlivé části byly otestovány před uvedením do praxe.

Největší překážkou v nasazení bylo nestatické osvětlení. V budoucnu by pro zmírnění tohoto efektu bylo vhodné nainstalovat LED pásky a případně změnit povrchovou úpravu hrací plochy na matnou. Dalším zlepšením by mohla být přidaná funkcionalita hráče na kopání, pro mojí část zpracování obrazu by to znamenalo naprogramovat indikátor pro moment, kdy má hráč vystřelit.

V praktické realizaci byl vyzkoušen detektor a prediktor na realizované aréně s mikropočítačem Raspberry Pi 3 Model B+, navrženým držákem a kamerou pro Raspberry Pi.

Bibliografie

[1] John Canny. *A computational approach to edge detection. Pattern Analysis and Machine Intelligence*. 1. vyd. IEEE Transactions on: PAMI-8(6):679–698, 1986.

[2] A Mathematical Theory of Communication. *OpenCV with Python By Example*. The Bell System Technical Journal 27, 1948.

[3] E.R. Davies. *Computer and Machine Vision. Theory, Algorithms, Practicalities*. 4th ed. Waltham, Mass: Academic Press, 2012. ISBN: 9780123869081. URL: http://search.ebscohost.com/login.aspx?direct=true%5C&db=nlebk%5C&an=453834%5C&scope=site.

[4] *Dokumentace Open CV*. 2019. URL: https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html (cit. 27. 04. 2020).

[5] A. Gelb. *Applied Optimal Estimation*. 1. vyd. Cambridge: MIT Press, 1974.

[6] Prateek. JOSHI. *OpenCV with Python By Example*. ISBN: 1785283936.

[7] Adrian Kaehler a Gary Bradski. *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*. First Edition. USA: O'Reilly, 2016. ISBN: 9781491937990. URL: https://books.google.com.au/books?id=SKy3DQAAQBAJ%5C&lpg=PT607%5C&ots=XGg5zrJXPp%5C&dq=TM_CCOEFF%5C&pg=PT606%5C#v=onepage%5C&q=ISSN%5C&f=false.

[8] R. E. Kalman. *A New Approach to Linear Filtering and Prediction Problems*. Transaction of the SME-Journal of Basic Engineering, 1960.

[9] S. Kolkur. "Human Skin Detection Using RGB, HSV and YCbCr Color Models". In: (2017), s. 9.

[10] wikpedia obrázek. URL: https://cs.wikipedia.org/wiki/HSV (cit. 31. 05. 2020).

[11] Lena Söderbergová Playboy 1972. URL: https://en.wikipedia.org/wiki/Lenna (cit. 30. 05. 2020).

[12] prodejce Raspberry Pi na ceskem trhu. URL: https://rpishop.cz/kamery/22-originalni-kamera-pro-raspberry-pi-camera-board-5mb-rev-13.html (cit. 31. 05. 2020).

[13] oficiální stránky Raspberry Pi. URL: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus (cit. 30. 05. 2020).

[14] Milan Sonka a Vaclav Hlavac. *Image processing, Analysis, and Machine Vision*. Third Edition. USA: Thomson, 2008. ISBN: 978-0-495-24428-7.

[15] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010. ISBN: 978-1-184882-934-3.

[16] LED neony a výroba. URL: https://blog.ledky.net/barevny-model-rgb/ (cit. 31. 05. 2020).