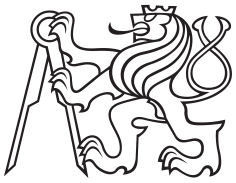
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Automatický vzlet a přistání robotické helikoptéry

Kateřina Kuglerová

Vedoucí: Ing. Jan Chudoba
Obor: Robotika
Studijní program: Kybernetika a robotika
Květen 2017

Poděkování

Děkuji vedoucímu práce Ing. Janu Chudobovi za vstřícnost a technickou podporu během celého projektu. Děkuji také své rodině a známým, kteří mě podpořili při mém studiu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně, a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 25. května 2017

podpis autora práce

Abstrakt

Cílem práce bylo navrhnout systém pro autonomní přistávání helikoptéry s využitím metod zpracování obrazu. Přehled známých metod zpracování obrazu na začátku práce je doplněn o vzory, které se používají jako přistávací plochy pro autonomní systémy, a jejich lokalizaci v obraze. Dále se práce zabývá lokalizačním systémem WhyCon. Jeho vzor je rozšířen o šachovnici 4x4 pole, která se nachází uprostřed původního vzoru a slouží k navádění stroje v blízkosti, kdy se celý vzor WhyCon nevejde do zorného pole kamery. V závěru práce se nachází popis modelu použité hexakoptéry a regulační algoritmus pro řízení přistání a vzletu.

Klíčová slova: zpracování obrazu, vizuální lokalizace, WhyCon, vzor šachovnice, automatické přistání

Vedoucí: Ing. Jan Chudoba

Abstract

The aim of the thesis was to design a system for an autonomous landing of a helicopter using image processing methods. An overview of well-known image processing methods at the beginning of the work is complemented by patterns that are used as landing surfaces for autonomous systems and their localization in the image. Further, the thesis deals with the WhyCon localization system. Its pattern is extended with a 4x4 field chessboard, which is placed in the middle of the original pattern, and is used to further navigate the drone near the landing pad where the whole Whycon model does not fit into the camera field. At the end of the thesis there is description of model of used hexacopter and control algorithm for landing and take-off.

Keywords: picture processing, visual-based localization, WhyCon, chessboard pattern, automated landing

Title translation: Automatic take-off and landing of robotic helicopter

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

V posledních letech došlo k velkému rozvoji malých bezpilotních letounů, jako jsou drony, helikoptéry nebo kvadrokoptéry. Dříve se v souvislosti s bezpilotními letouny mluvilo hlavně o jejich vojenském využití, čím dál častěji se ale malé drony používají i v civilním životě. Většinou se jedná o průzkum míst, která jsou člověku nedostupná nebo dokonce nebezpečná (poničené oblasti při přírodních katastrofách), hledání pohřešovaných osob při záchranných operacích [12], doručování různých věcí (zboží, balíčky s léky a jídlem) nebo sledování zvířat ve volné přírodě [13]. Další velkou oblastí s komerčním využitím se stalo natáčení záběrů z ptačí perspektivy, hojně využívané ve filmech [14], dokumentech a reportážích.

Spolu s rozvojem bezpilotních letounů došlo v oblasti počítačového vidění také k rozvoji metod zpracování obrazu. Mezi důležité okamžiky letu dronu patří jeho vzlet a přistání. GPS lokalizační systémy jsou sice schopné navigovat stroj nad přistávací plochu, nedokáží ale obvykle zacílit přímo na ni. Diferenciální GPS systém dosahuje díky specializovaným vysílačům centimetrové přesnosti s nevýhodou náročných výpočetních operací. Standardní GPS dosahuje přesnosti v řádu jednotek metrů, tým ale J.Farrela na konci roku 2015 publikoval kombinovaný systém GPS, který dosahuje podobné přesnosti jako D-GPS a má menší výpočetní náročnost. Také se může přistávací plocha nacházet v místě, např. v budovách, kde nelze zachytit GPS signál a navádění pomocí GPS systému tak není možné využít. Lokalizační systém založený na zpracování obrazu je naproti tomu schopný sledovat ve snímaném obraze typický vzor na přistávací ploše nebo určit vzdálenost mezi ní a kamerou (pokud jsou dopředu známé geometrické údaje vzoru) a navádět stroj bezpečně na přistávací plochu.

V kapitole 2 zmiňuji podobné projekty zabývající se automatickým přistáním dronu nebo helikoptéry. Tyto studentské projekty využívají v navádění jako hlavní zdroj informací o poloze dronu/helikoptéry systém s vizuální lokalizací vzoru na přistávací ploše.
Různým metodám zpracování obrazu, segmentaci a vyhledávání specifických rysů v obraze se věnuje kapitola 3.
V kapitole 4 se podrobněji věnuji lokalizačnímu systému WhyCon, který jsem si vybrala pro svoji práci.
Na tuto kapitolu navazuje v kapitole 5 úprava vzoru používaného systémem WhyCon a s ní spojená úprava lokalizačního algoritmu.
Předposlední kapitola se zabývá modelem dronu a regulačním algoritmem, který má za úkol řídit rychlost dronu při automatickém vzletu a přistání.
Na konci práce se nachází experimentální ověření upraveného algoritmu na fotkách z webkamery a kamery připevněné na dronu.

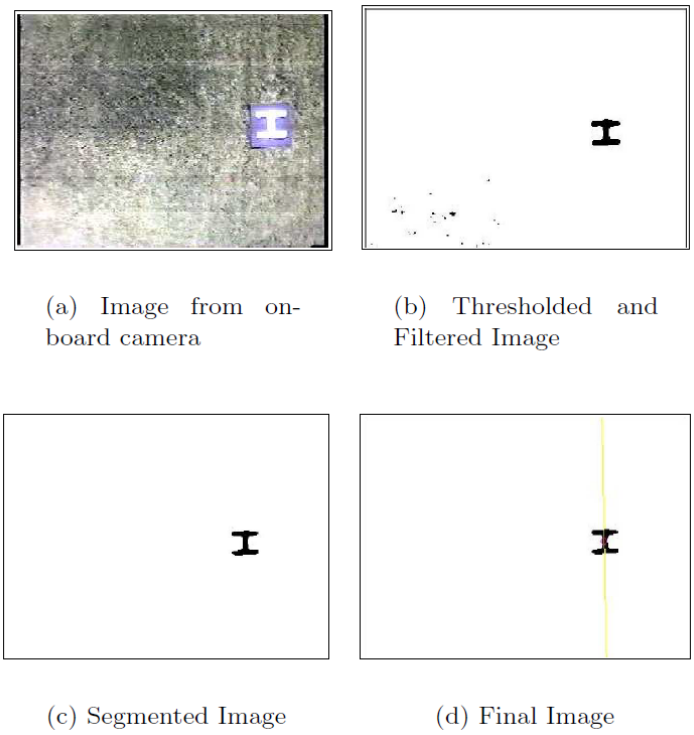
Kapitola 2

Související projekty

Jak už bylo zmíněno v úvodní kapitole, kvadrokoptéry a drony zažívají v posledních letech velký vzestup. Díky tomu existuje po celém světě řada projektů, které se zabývají automatickým řízením dronů včetně automatického vzletu a přistání.

Společný projekt Univerzity of Southern California a California Institute of Technology [1] využívá pro automatické přistání helikoptéry na přistávací ploše velkého písmene **H**. Písmeno je bílé na černém pozadí, proto se v algoritmu nejdříve na vstupní obraz aplikuje metoda prahování (podrobněji v následující kapitole) a použijí se filtry pro odstranění šumu (2.1). Po segmentaci se písmeno v obraze najde porovnáním specifických geometrických rozměrů písmena s parametry nalezených segmentů. Díky tvaru písmene je algoritmus schopen nalézt relativní natočení helikoptéry vůči přistávací ploše, čehož využívá při natočení při přistání a průměrná odchylka udávaná v dokumentu se pohybuje okolo 31 cm od určené pozice a 6°od svislé osy písmene.

Použití přistávací plochy s jedním velkým znakem předkládá problém, že když se znak nevejde celý do obrazu z kamery, dron či helikoptéra pak musí přistát "naslepo", tedy není nic, podle čeho by mohl stroj určit, v jaké vzdálenosti se nachází od přistávací plochy. Další projekty se zaměřily částečně i na tento problém a zkusily ho vyřešit přidáním dalších podobných vzorů do obrazce.



(a) Image from on-board camera

(b) Thresholded and Filtered Image

(c) Segmented Image

(d) Final Image

Obrázek 2.1: Zpracování vstupního obrazu v projektu [1]

Projekt Linkoping University [2] využívá pro navigaci helikoptéry přistávací plochu s celkem 15 kruhy (2.2). Vždycky 3 kruhy leží v rozích trojúhelníka.

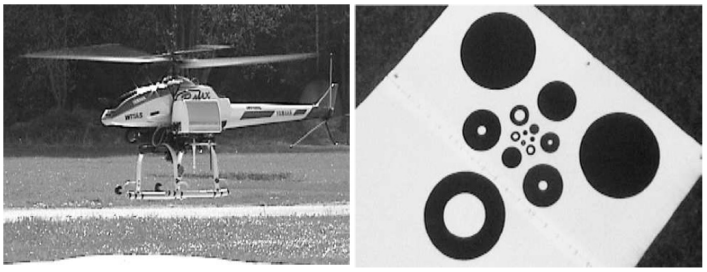


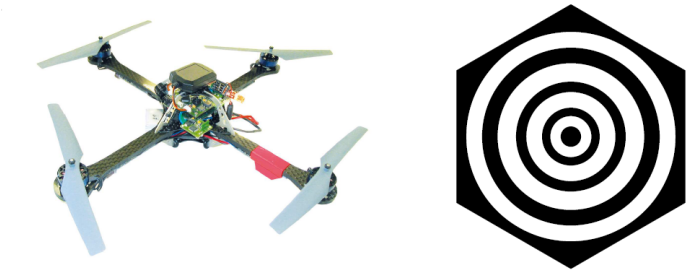
Fig. 1. The WITAS helicopter descending to the landing pad.

Fig. 2. Landing pad with reference pattern seen from the on-board camera.

Obrázek 2.2: Zpracování vstupního obrazu v projektu [2]

Každá skupina kruhů má vlastní specifické velikosti vnitřních kruhů, podle kterých je lze bezpečně odlišit a zároveň je každá skupina tvořící trojúhelník menší než ta předchozí (poloměr 2 - 32 cm, vzdálenost středů 8 - 128 cm). Tím se docílí toho, že když je přiblížení helikoptéry tak velké, že už se vnější 3 kolečka nevejdou do obrazu kamery, algoritmus pro výpočet polohy použije další největší 3 současně viditelná kolečka (zároveň díky odlišným středům ví, která to jsou). Při použití helikoptéry Yamaha RMAX s celkovou délkou 3,6 m je udávaná odchylka asi 42 cm (13% průměru vrtule helikoptéry).

Projekt Chemnitz University of Technology [3] se zaměřil přímo na problém s přesným naváděním dronu až do momentu dosednutí. Jejich přistávací



(a) UAV (b) Landing target

Obrázek 2.3: Zpracování vstupního obrazu v projektu [3]

plocha (2.3) se skládá ze 4 bílých kroužků se společným středem. Největší detekovaný kruh pak algoritmus určí podle toho, jaký je poměr jeho vnějšího a vnitřního poloměru. Autoři projektu zmiňují, že se můžou ke vzoru přidat další (větší nebo menší) kruhy podle potřeb identifikace vzoru z více vzdáleností. Dále udávají maximální odchylku do 2 cm při průměru vnějšího kruhu 45 cm v případě, že se dron nacházel přímo nad přistávací plochou. Během celého projektu předpokládají, že kamera směřuje kolmo na přistávací plochu a z toho odvozují i výpočet vzdálenosti.

Jako poslední bych zde chtěla zmínit komerční drony Phantom společnosti DJI, které také mají funkce pro automatické přistání. Drony Phantom se spoléhají při letu a přistání i na jiné senzory, než jsou ty optické (ty se využívají pro stabilizaci a jsou k dispozici ve výšce 30-300 cm nad povrchem). Uživatelé nicméně na diskuzních fórech velmi často píší, že při přistání chytají drony do rukou a nespoléhají se na automatické přistání drona na zem.

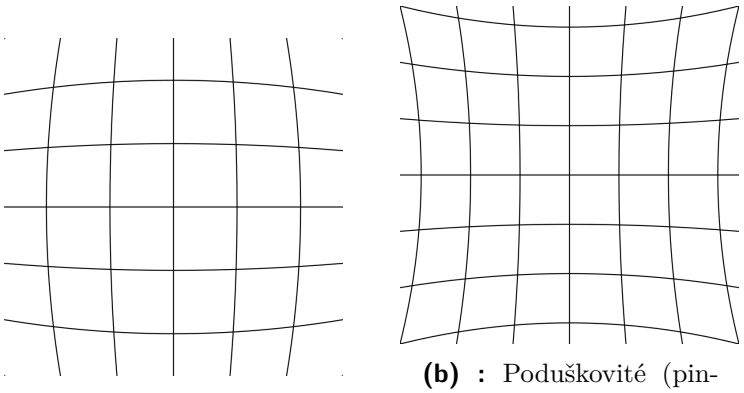
Kapitola 3

Zpracování obrazu

Pro vzlet i přistání helikoptéry je důležitá její relativní poloha vůči přistávací ploše. Helikoptéra snímá obraz z kamery, který po zpracování musí splňovat požadavek, že v obraze bude možné pomocí algoritmu vyhledat specifický obrazec na přistávací ploše. Použití konkrétních metod závisí vždy na kvalitě použité kamery, na šumu ve snímaném obrazu nebo na klíčových vlastnostech hledaného obrazce. Pokud se nachází v zachyceném obraze velké množství šumu nebo nerovnoměrně rozložený jas a kontrast, zpracování obrazu a vyhledání obrazce se můžou stát neproveditelnými.

3.1 Kalibrace kamery

Při pořizování snímků jednoduchými a levnými kamerami dochází v obraze k významnému zkreslení. Přímky v reálném světě se zobrazí jako křivky na snímku. I při použití kvalitních čoček se zkreslení projeví, i když je to pak míň zjevné než u nekvalitních kamer. Zkreslení se dá předejít při konstrukci objektivů, které jsou vyrobeny pro určitou vzdálenost, kde se zkreslení neprojevuje tolik jako při snímání objektů tímto objektivem na jiné vzdálenosti. Proto je kalibrace kamery základní operace, kterou je potřeba udělat před zpracováním obrazu. To platí hlavně v případech, kdy se ze vzoru v obraze počítá vzdálenost kamery od vzoru.
Jako model ideální kamery se používá dírková kamera, kde se 3D body zobrazují do roviny pomocí perspektivní transformace. Nejčastějšími typy zkreslení jsou pak radiální a tangenciální na 3.1, [15].



(a) : Soudkovité (barrel)

(b) : Poduškovité (pincushion)

Obrázek 3.1: Základní typy radiálního zkreslení [4]

Zkreslení je popsáno v knihovně OpenCV 4 rovnicemi pro neznámé parametry (k_x se týkají radiálního zkreslení a p_x tangenciálního)

$$Dist_{coef} = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix} \tag{3.1}$$

a pro převod mezi reálnými souřadnicemi a souřadnicemi v obraze platí

$$\begin{bmatrix} x \\ y \\ \mathrm{w} \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \tag{3.2}$$

kde f_x a f_y jsou ohniskové vzdálenosti kamery a c_x a c_y jsou optické středy vyjádřené v pixel souřadnicích. Matice s těmito 4 prvky se nazývá intrinsická matice. Cílem kalibrace je získat všechny parametry pomocí obrázků, na kterých se nachází vzor se známými parametry, kde každý obrázek udává vztah mezi 2D a 3D body daného vzoru. Jako takový vzor se velmi často používá šachovnice se stranami o různém počtu polí nebo vzor složený z malých (a)symetricky rozmístěných kruhů.
Při sbírání obrázků určených ke kalibraci je důležité, aby na každém z nich pokrýval vzor jinou část samotného obrázku, především hranice obrázku, kde se nejvíce projeví radiální a tangenciální zkreslení.

3.2 Metody zpracování obrazu

Cílem zpracování obrazu je zjednodušit ho tak, aby v něm bylo možné vyhledávat rychle a efektivně obrazce, hranice, křivky, čáry a další (z pohledu počítačového vidění) zajímavé objekty. Výsledkem metody je rozdělení vstupního obrazu na jednotlivé díly (segmenty), které mají společnou nějakou vlastnost (barvu, intenzitu, apod.). Segmentační metody se dají rozdělit podle přístupu na několik hlavních skupin [16]:

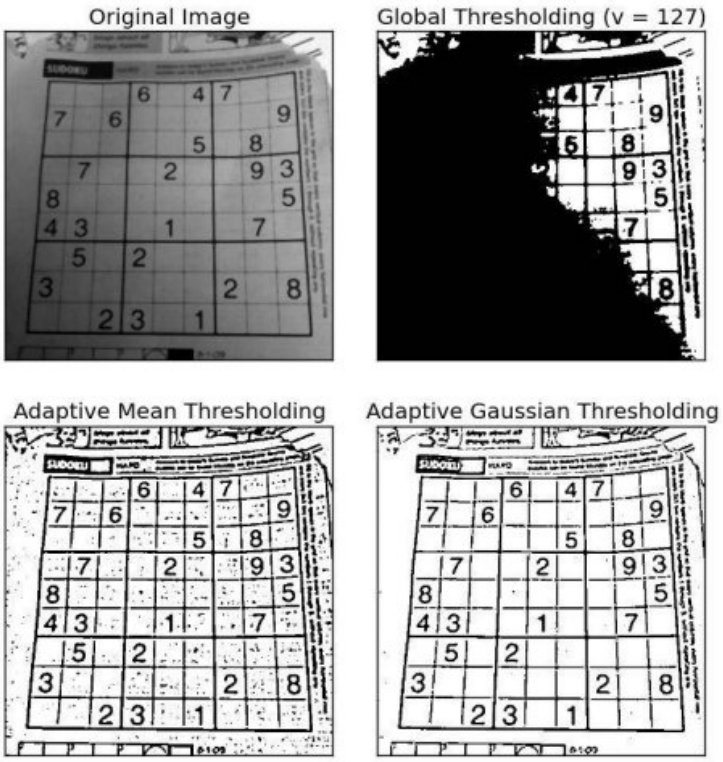
- Detekce hran (edge-based) - metody v obraze hledají významné hrany na základě rozdílných hodnot sousedních pixelů. Hlavní nevýhodou metod založených na detekci hran je obtížnost detekce vzoru na obrazech se šumem.
- Detekce regionů (region-based) - regiony jsou v obraze ohraničené skupiny pixelů, které mají stejné nebo podobné vlastnosti. V principu fungují tyto metody podobně jako ty založené na detekci hran, výsledky ale nemusí být (a většinou nebývají stejné).
- Statistické detekce - detekce probíhá na základě výsledků statistické analýzy obrázku (analýza proběhne ještě před samotnou detekcí segmentů v obrázku).
- Znalostní detekce (knowledge-based) - tyto metody mají předem k dispozici soubor trénovacích dat hledaného vzoru, ve kterých pak vyhledávají vzory nalezené v obraze.
- Hybridní detekce - metody v téhle skupině slučují více prvků ostatních metod, zároveň ale nepatří ani do jedné z nich.

3.3 Prahování

Nejjednodušší metoda segmentace obrazu se nazývá thresholding (česky prahování). Pracuje tak, že si na počátku zpracování obrazu určí hraniční hodnotu barvy pixelu, která bude rozdělovat pixely na tmavé a světlé (černé a bílé). Potom metoda projde všechny pixely v obraze a roztřídí je do skupin. Celkový počet skupin, do kterých jsou pixely děleny, může být v podstatě libovolný (nemusí být roven 2), ale při postupném zvyšování pak dochází k roztřídění do tolika skupin, že pak v obraze hledaný vzor zpravidla vůbec nejde najít a metoda ztrácí smysl. Proto je lepší nechat počet skupin co nejmenší.

3.4 Adaptivní prahování

Tato metoda funguje na podobném principu jako ta předchozí, pouze s tím rozdílem, že nejdříve obrázek rozdělí na několik menších oblastí. Pak se hraniční hodnota mezi černou a bílou (za předpokladu, že metoda rozděluje pixely v obrázku pouze na dvě barvy) spočítá pro každou malou oblast zvlášť (jednoduše jako průměrná hodnota pixelu v dané oblasti, nebo pomocí váženého součtu hodnot a Gaussovy funkce).
Různá hraniční hodnota pro různé části obrázku se hodí v případech, kdy je pozorovaný předmět osvětlený nerovnoměrně a rozdíl mezi metodami je vidět na 3.2, [5].
Hlavní nevýhoda oproti jednoduché metodě prahování je ta, že se musí vstupní obraz částečně zpracovat ještě před aplikováním metody. Potom je na tvůrcích aplikací a projektů, aby se rozhodli, zda upřednostní rychlou a nenáročnou metodu prahování, nebo o něco výpočetně náročnější, zato ale kvalitnější adaptivní prahování.



Obrázek 3.2: Rozdíl mezi metodami thresholding a adaptive thresholding funkcí OpenCV [5]

3.5 Metody zvětšujících se segmentů (flood fill, region growing)

Tyto metody spoléhají hlavně na to, že sousedící pixely, které patří do stejného segmentu, mají také stejné nebo podobné hodnoty. Metoda prochází seznam dosud neprozkoumaných pixelů-sousedů a pokud daný pixel vyhovuje vlastnostem sousedícího segmentu, je do něj zařazený a jeho sousedi se dostanou do seznamu k prozkoumání.

3.6 Vlastnosti vzoru na přistávací ploše

Vzhledem k výše popsaným metodám zpracování obrazu jsou na vlastnosti vzoru na přistávací ploše kladeny následující požadavky:

- Vzor by měl mít ostré hrany a přesně daný geometrický tvar umožňující jednoduché rozdělení obrazu na segmenty. Nejčastější vzory na přistávacích plochách jsou proto kruhy a mezikruží, inspirované přistávacími plochami pro helikoptéry.
- Barvy, které jsou použity k vykreslení vzoru, by měly být vůči sobě dost kontrastní, aby bylo možné vzor rozpoznat i za horších světelných podmínek (tj. málo, nebo hodně světla). Proto se většinou využívají dvoubarevné přistávací plochy, nejčastěji je to kombinace černé nebo bílé a některé další barvy, která se zmíněnými dobře kontrastuje (oranžová, červená, modrá), nebo přímo vzor v kombinaci černé a bílé. Při použití barev v přistávacím vzoru se nabízí možnost jejich využití, protože se však u většiny metod zpracování obrazu nejdříve převede vstupní obraz do odstínů šedi, s výhodou jde na přistávací plochu umístit již černobílý obraz (barvy se při převodu změní na šedou, černá s bílou tak mezi sebou budou mít největší kontrast). Barvy vzoru (a jejich celkový počet) a vybraná metoda spolu úzce souvisí a navzájem se ovlivňují.

Kapitola 4

WhyCon

WhyCon [6, 7, 17, 18] je lokalizační systém pracující s obrazy z kamer(y) schopný detekce specifických obrazců a zjištění pozice kamery v reálném čase. Byl vyvinut jako společný projekt ČVUT, the University of Buenos Aires a University of Lincoln. Existuje několik verzí detektoru WhyCon (každá použitelná v jiných situacích), pro bakalářskou práci jsem si vybrala verzi WhyCon-ROS.

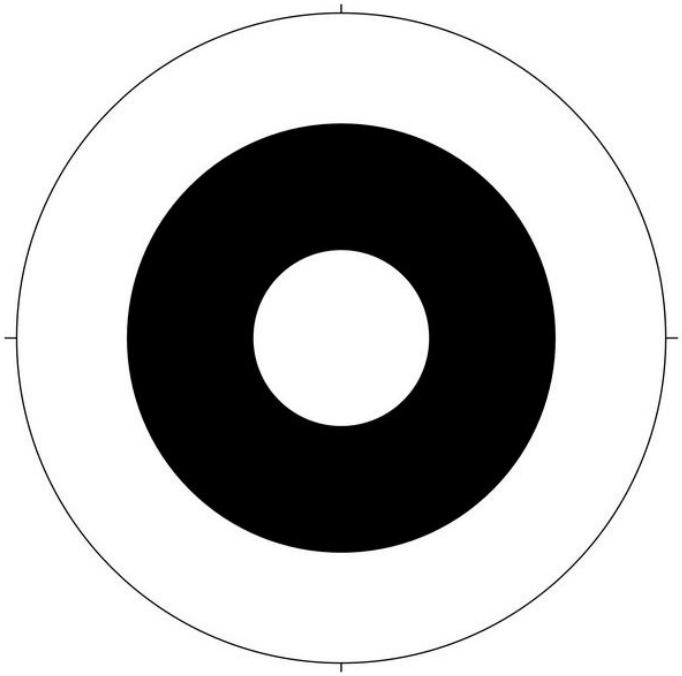
Systém byl vyvinut jako jednoduché, levné a nenáročné řešení lokalizačních úloh pro pozemní a létající roboty, včetně vzájemné lokalizace více robotů. Hlavní výhodou systému je vysoká výpočetní účinnost, která umožňuje systém WhyCon použít i u robotů, které mají omezeny výpočetní prostředky a levnější kamery s rozlišením (rozlišení u standardní webkamery se pohybuje okolo hodnot (1280-640)x(720-480) pixelů). Lokalizace robota vůči vzoru je zároveň rychlá a přesná, díky tomu se systém WhyCon používá v mnoha projektech (4.1).

4.1 Algoritmus

Lokalizační systém WhyCon používá k lokalizaci specifický obrazec, který se skládá z vnějšího černého kruhu a vnitřního bílého kruhu(4.2).
Výhoda kruhového vzoru spočívá v tom, že se u něj nerozlišuje natočení. Ať je kruh snímaný z jakékoli strany (zepředu, z boku), vypadá vždy stejně a to

usnadňuje jeho detekci v obraze.

Obrázek 4.1: Lokalizační systém WhyCon v projektech [6]



Obrázek 4.2: Obrazec používaný k detekci systémem WhyCon [6]

Vně černého kruhu je ještě bílá část, která přímo není součástí obrazce, ale pomáhá při lokalizaci oddělením černé části obrazce od okolí. Zamezí se tak možnosti, že by se v těsné blízkosti černého kruhu nacházel nějaký tmavý předmět, který by zasahoval do obrazce a znemožnil tak identifikaci vzoru detektoru WhyCon.

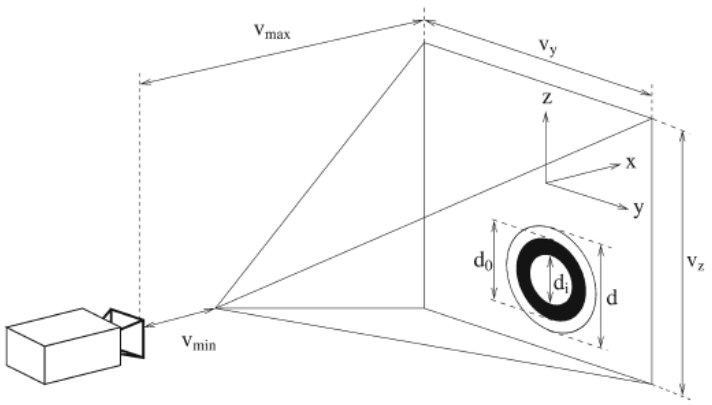
Pro práci s obrazem a kamerou se v programu využívá funkcí knihovny OpenCV [19].
Program se v obraze z kamery pokusí detekovat nejprve černý kruh tak, že prochází pixel po pixelu obraz z kamery a rozděluje jednotlivé pixely na černé, nebo bílé. Používá k tomu právě výše popsané metody zpracování obrazu, kdy si převede vstupní obraz na škálu černobílé. Bere si postupně každý nezařazený pixel, a přiřadí ho k sousedícímu segmentu, nebo ho použije jako základ nového segmentu. Pseudokód *Alg.* (2) on page 25 detailněji popisuje celý postup algoritmu detekce vzoru v obraze.

Údaje o obrazci jsou předány lokalizačnímu systému, který pomocí nich vypočte pozici obrazce vůči kameře (kamera je na pozici [0, 0, 0] = [x, y, z], vzdálenosti jsou udávány v metrech).

Součástí systému WhyCon je i jednoduchý program na kalibraci kamery. Ten používá ke kalibraci šachovnici (uživatel si zvolí počet políček podle svojí předlohy) a funkce pro kalibraci z knihovny OpenCV. Výsledkem programu je pak soubor `calibration.xml`, který obsahuje důležité kalibrační údaje použité kamery a který se v argumentu předá detektoru.

4.1.1 Omezení algoritmu

Předpokládá se, že při přistávání i vzletu bude helikoptéra přibližně nad určeným přistávacím místem. To znamená, že obraz kamery bude snímat rovinu, ve které bude obrazec určující přistávací plochu, pod kolmým (nebo jemu blízkým) úhlem. WhyCon je schopen detekovat obrazec a určit jeho pozici celkem přesně i při velkém úhlu mezi středem obrazce a kamerou (podle tabulky přesností, která je k dispozici v dokumentaci detektoru WhyCon [7], byla průměrná odchylka okolo 1 % při úhlu 40 °mezi obrazcem a kamerou, okolo 0,6 % při kolmém snímání roviny s obrazcem při použití standardní web-kamery), proto jsem se rozhodla toto omezení zanedbat.
Další problém nastane, když je kamera příliš blízko přistávací ploše a to tak, že nezachytí celý vnější (černý) kruh a nemůže určit svoji pozici při úplném začátku vzletu/konci přistání (na 4.3 je znázorněna nejmenší vzdálenost kamery od roviny obrazce, která je závislá na samotných parametrech kamery). Při úlohách typu sledování pohybujícího se objektu, kde si robot udržuje od objektu konstantní vzdálenost, není nutné tento problém řešit, pro přistání a vzlet je ale důležité znát pozici dronu co nejdelší dobu, aby se mohlo předejít slepým přistáním.



Obrázek 4.3: Označení os a důležitých parametrů při lokalizaci, [7]

Algorithm 1: Detekce WhyCon vzoru, [7]

Data: $(p_0, \tau$,Image): p_0 - position to start search; τ - threshold; Image being processed

Result: (c, p_0, τ): c - the pattern data; p_0 - position to start next search; τ - an updated threshold

$s_{id} \leftarrow 0; i \leftarrow p_0$ // initialize
// #1 search throughout the image
repeat
  if *pixel_class[i] = unknown* **then**
    if *classify(Image[i]) = black* **then**
      pixel_class[i] $\leftarrow$ black
  // initiate pattern search
  if *pixel_class[i] = black* **then**
    // search for outer ring
    $q_{end} \leftarrow q_{start} \leftarrow 0$
    $c_{outer} \leftarrow$ flood-fill_seg(i, ρ_{outer}, black)
    if *valid(c_{outer})* **then**
      // search for inner ellipse
      $j \leftarrow center(c_{outer})$
      $c_{inner} \leftarrow$ flood-fill_seg(j, ρ_{inner}, white)
      if *valid(c_{inner})* **then**
        // test area ratio (no. of pixels):
        if *ratio_ok* **then**
          check segments for concentricity
          compute ellipse semiaxes e_0, e_1
          if $q_{end} \approx \pi/e_0 e_1/$ **then**
            assign segment ID
            compensate illumination
            mark segment as valid
  $i \leftarrow (i + 1)$ mod sizeof(Image) // go to next pixel
while $i \neq p_0$
// #2 set the thresholding value
if *valid(c_{inner})* **then**
  $\tau \leftarrow (\mu_{outer} + \mu inner)/2$
  paint over all inner ellipse pixels as black
else
  $\tau \leftarrow$ binary search sequence
// #3 perform the cleanup
if *only two segments examined* **then**
  reset $pixel_class$[] inside bounding box of c_{outer}
else
  reset entire $pixel_class$[]

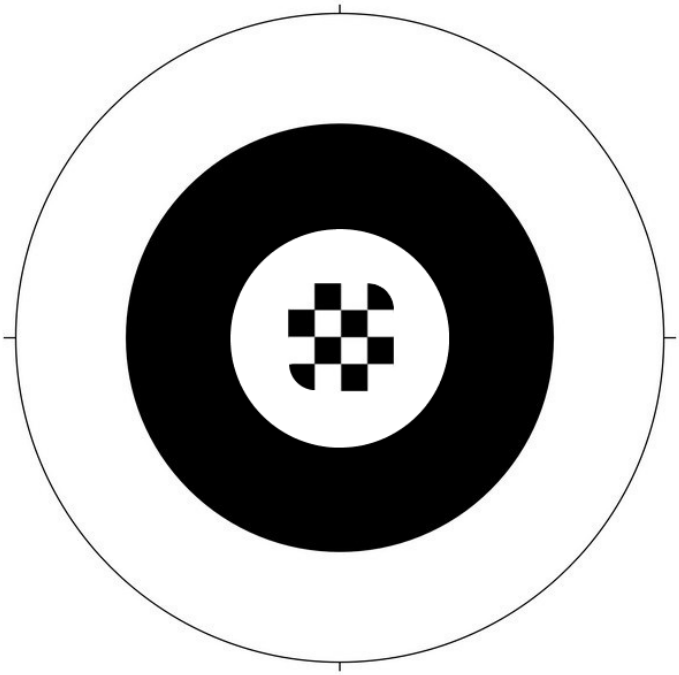
Kapitola 5

Úprava obrazce a detektoru WhyCon

Kvůli omezením lokalizačního systému jsem se rozhodla přidat další orientační bod doprostřed obrazce WhyCon. Zamýšlený cíl byl zachovat dobré vlastnosti lokalizačního systému ve středních vzdálenostech od kamery a umožnit lokalizaci v ještě větší blízkosti dronu od přistávací plochy.

5.1 Výběr obrazce

Doprostřed vnitřního bílého kruhu jsem vložila černobílou šachovnici o rozměrech pole 4x4, jejíž střed se shoduje se středem obrazce WhyCon (5.1). Šachovnice byla zvolena díky svým typickým rysům, které se hodí pro vyhledávání v obraze. Mezi tyto rysy patří hlavně kontrast mezi dvěma barvami (které jsou už v samotném obrazci detektoru použité) a ostré hranice mezi sousedícími poli šachovnice.
Okraje černých krajních polí šachovnice (pravý horní a levý spodní roh) jsou mírně zaoblené a zároveň je zvětšená hranice mezi černým kruhem a vnitřním bílým kruhem. Když by byly šachovnice a černý kruh příliš blízko sebe, hrozilo by, že budou při zpracování obrazu zařazeny pod jeden segment a nebude tak možné systémem identifikovat správně celý obrazec.



Obrázek 5.1: Upravený vzor detektoru WhyCon

5.2 Šachovnice v počítačovém vidění

Šachovnice se v oblasti počítačového vidění používají velmi často díky výše popsaným typickým rysům, které se hodí pro algoritmické vyhledávání v obrazech. Použití šachovnic v počítačovém vidění může být rozděleno do několika hlavních oblastí.

5.2.1 Kalibrace kamery

Podrobný popis kalibrace kamery se nachází v kapitole Zpracování obrazu. Lokalizační systém WhyCon kromě detektoru obsahuje i program na kalibraci kamery `camera-calibrator`, který využívá šachovnici pro určení kalibračních parametrů. Soubor obsahující výstup kalibračního programu se pak zadá lokalizačnímu systému jako vstupní argument a ten je využije k výpočtu při lokalizaci a pro odstranění zkreslení obrazu.

5.2.2 Hledání prvků šachovnice v obraze

Obecně při vyhledávání šachovnice v obraze převládají dva algoritmy:

- Harris [20] - detektor analyzuje vlastní hodnoty tensoru struktury u každého pixelu a pokud je vlastní hodnota tensoru významně větší než okolní hodnoty, označí pixel za roh pole. Proto se mu také říká Harrisův detektor rohů. Existují další algoritmy pro hledání rohů v obraze, Harrisův je z nich pravděpodobně nejznámější.

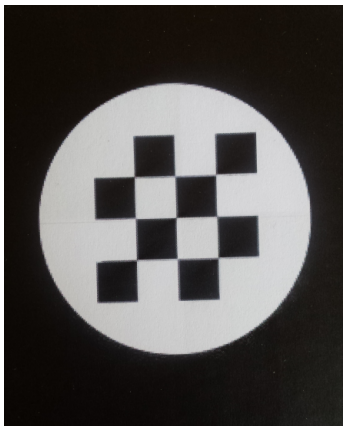
- Hough [21] - detektor vychází z toho, že přímku lze popsat pomocí dvou parametrů (ρ, θ), které udávají vzdálenost od počátku k nejbližšímu bodu přímky a úhel mezi osou x a přímkou. Obrázek, ve kterém se nejdříve detekují hrany, se tak převede Houghovou transformací do matice, jejíž prvky obsahují počet pixelů, které leží na přímce popsané dvěma parametry. Za předpokladu, že na obrázku jsou nejdelší přímky zároveň přímky na šachovnici, vybere se z matice 18 největších prvků (přímky tvořící síť šachovnice 8x8 polí). Ty jsou rozděleny do 2 skupin na základě úhlu, který svírá dotyčná přímka s osou x. Rozdíl úhlů mezi těmito dvěma skupinami je ideálně 90 °, jako je úhel mezi přímkami, jejichž průsečík tvoří roh pole.

V projektu jsem využila open-source knihovny OpenCV (knihovny využívá i program WhyCon při práci s obrazem z kamery), které obsahují funkci na principu Harrisova detektoru rohů pro vyhledávání šachovnice v daném obrázku
`bool findChessboardCorners(image, patternSize, corners, flags)`,
která vrací souřadnice průsečíků polí v pixelech. Funkce umožňuje vyhledávat libovolnou šachovnici, která má počet polí v řádku i ve sloupci roven aspoň 4, pro menší šachovnice funkce vrací chybu. Toto omezení je ve funkci zavedeno z důvodu robustnosti, při menším počtu polí by mohla být jako šachovnice chybně detekována oblast se střídajícími se světlými a tmavými plochami nebo naopak by mohl zaniknout kontrast mezi některými poli a šachovnice by detekována nebyla.

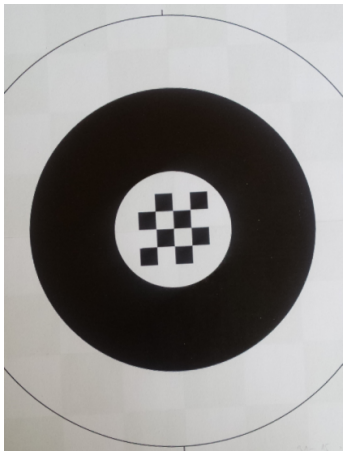
5.3 Úprava kódu algoritmu WhyCon

Po přidání šachovnice doprostřed bílého kruhu se lokalizace rozdělí na několik možných scénářů (předpokládá se, že při přistání bude kvadrokoptéra mít v dohledu vzor WhyCon):

- Velká vzdálenost od vzoru - detekuje se úspěšně vnější černý kruh, ale šachovnice ne (pravděpodobně kvůli nízkému rozlišení)
- Střední vzdálenost od vzoru - detekuje se úspěšně černý kruh i šachovnice uvnitř něj (5.2b)
- Malá vzdálenost od vzoru - černý kruh se nevejde do zorného pole kamery a WhyCon detekuje pouze šachovnici (5.2a)



(a) : Pouze šachovnice detekovatelná



(b) : Kruh i šachovnice detekovatelné

Obrázek 5.2: Detekce vzoru v různé vzdálenosti

V prvním případě se postupuje podobně jako v původním detektoru, jen s tím rozdílem, že vnitřní kruh není čistě bílý. Nelze ho tedy najít flood-fill metodou, která pixely do segmentu (vnitřního kruhu) zařazuje na základě jejich (bílé) barvy. Na základě toho, že je pak u nalezeného segmentu testováno několik jeho vlastností (např. velikost vzhledem k vnějšímu černému kruhu nebo kulatý tvar), rozhodla jsem se použít pro nalezení vnitřního segmentu také flood-fill metodu. Ta nepřiřazuje pixely do vnitřního segmentu na základě jejich barvy, ale vyplňuje vnitřek černého kruhu tak dlouho, dokud nenarazí na jeho vnitřní hranice. Hranici je možné rozpoznat podle toho, jestli je pixel na případné hranici už zařazen do černého vnějšího kruhu, nebo ne.

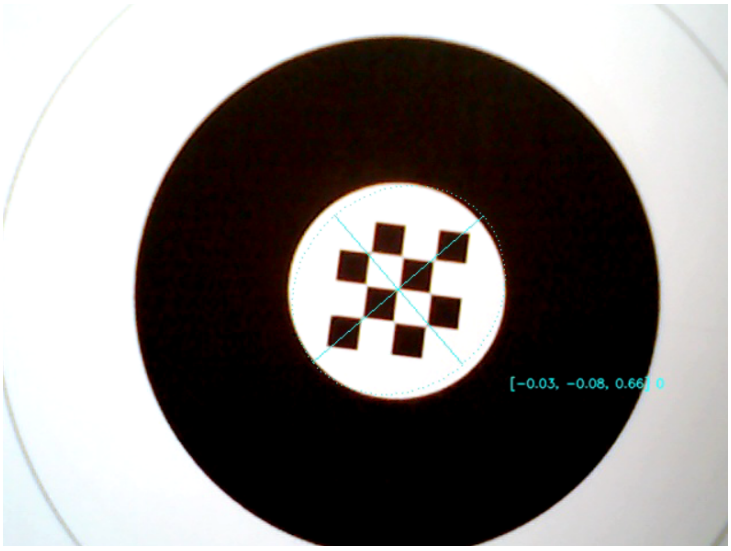
V druhém případě je na začátku vyhledávání nalezena šachovnice v obraze. Znalost pozice šachovnice v obraze umožňuje systému WhyCon po detekci vnějšího černého kruhu pouze porovnat jeho střed se středem šachovnice:

$$\begin{aligned} |chess_x - circle_x| < kcircle_r \\ |chess_y - circle_y| < kcircle_r \end{aligned}'$$

kde $chess_x$ a $chess_y$ jsou souřadnice středu šachovnice v pixelech, $circle_x$ a $circle_y$ jsou souřadnice předpokládaného středu kruhu v pixelech, k je toleranční konstanta a $circle_r$ je přibližný poloměr černého kruhu:

$$circle_r = \frac{(circle_x - circle_{minx}) + (circle_y - circle_{miny})}{2} \tag{5.1}$$

Hodnota $circle_{minx}$ ($circle_{miny}$) je bod na vnějším černém kruhu, který má ze všech bodů kruhu nejmenší souřadnici x (y).
Výše popsaná podmínka potvrdí shodu středu šachovnice a kruhu, ale nezabrání špatné detekci v případě, že WhyCon detekuje jako černý vnější kruh buď nějaké černé políčko šachovnice nebo v případě nevhodné hodnoty thresholdu i vnitřní bílý kruh obsahující šachovnici (5.3).
Proto je zavedena v postupu vyhledávání vzoru ještě jedna podmínka na kon-

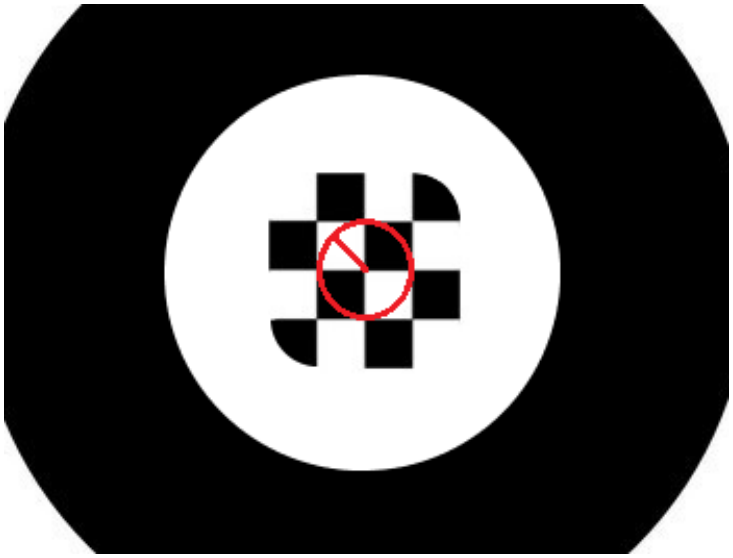


Obrázek 5.3: Nesprávně detekovaný kruh

trolu poměru velikosti šachovnice a vnějšího kruhu. Už vypočtený poloměr černého kruhu je porovnáván s průměrnou vzdáleností $chess_r$ 4 rohů šachovnice od jejího středu (4 body leží na červené kružnici vyznačené na 5.4). Poté v podmínce musí platit podobná nerovnice jako v předchozím případě:

$$\frac{chess_r}{circle_r} < l \tag{5.2}$$

, kde l je toleranční konstanta. Algortimus detektoru po obou úspěšných podmínkách pokračuje dál jako v původním detektoru, tedy zpřesní získané



Obrázek 5.4: Červeně: kružnice opsaná 4 rohům šachovnice

informace o černém kruhu na základě jeho známých geometrických rozměrů a předá získané souřadnice kruhu lokalizačnímu systému.

V případě, kdy detektor v obraze najde pouze šachovnici, předá lokalizačnímu systému informace o ní. Za tímto účelem byl do struktury Circle přidán jeden atribut `std::vector<cv::Point2f> chess_coords`, který v sobě nese souřadnice šachovnice, i když je jinak kruh nenalezený.

5.4 Relativní lokalizace helikoptéry vůči vzoru

Vzhledem k tomu, že je samotný vzor rozdělený na 2 části, je i lokalizace a výpočet vzdálenosti rozdělený na 2 funkce. Proto jsou do třídy Circle přidané ještě dva atributy `bool chessInside` a `bool circleValid`, podle kterých program pozná, jestli má k lokalizaci použít detekovaný kruh, nebo šachovnici.
Lokalizační funkce původního systému WhyCon počítá vzdálenost kamery od vzoru pomocí matice kuželosečky (elipsy/kruhu), jejích vlastních čísel a vektorů. Konkrétní vzorce jsou dohledatelné v dokumentaci systému WhyCon [7].
U lokalizace pomocí šachovnice se využívá faktu, že je k dispozici 9 bodů v prostoru, jejichž souřadnice jsou známé. Všechny body se nachází v rovině $z = 0$ a tvoří v ní síť 3x3, kde vzdálenost mezi body sítě je strana jednoho políčka a. K těmto bodům jsou přiřazeny jejich průměty do roviny, kterou

Algorithm 2: Detekce upraveného WhyCon vzoru

Data: $(p_0, \tau$,Image): p_0 - position to start search; τ - threshold; Image being processed

Result: (c, p_0): c - the pattern data; p_0 - position to start next search

$s_{id} \leftarrow 0; i \leftarrow p_0$ // initialize
// #0 search for chessboard in the image
findChessboard(Image)
// #1 search throughout the image
repeat
  if *pixel_class[i] = unknown* **then**
    if *classify(Image[i]) = black* **then**
      pixel_class[i] $\leftarrow$ black
  // initiate pattern search
  if *pixel_class[i] = black* **then**
    // search for outer ring
    $q_{end} \leftarrow q_{start} \leftarrow 0$
    $c_{outer} \leftarrow$ flood-fill_seg(i, ρ_{outer}, black)
    if *valid(c_{outer})* **then**
      // test chessboard and circle ratio
      if *chessboardFound && ratio_ok* **then**
        check concentricity
        compute ellipse semiaxes e_0, e_1
        if $q_{end} \approx \pi / e_0 e_1 /$ **then**
          assign segment ID
          compensate illumination
          mark segment as valid
    else
      // test inner segment and circle ratio
  $i \leftarrow (i + 1)$ mod sizeof(Image) // go to next pixel
while $i \neq p_0$
if *!valid(c_{outer}) && chessboardFound* **then**
  pass chessboard coordinates

představuje obraz kamery. Zároveň je k dispozici soubor *calibration.xml* obsahující kalibrační parametry kamery. Problém, kdy jsou známé výše popsané parametry a hledá se pozice kamery (translace a rotace) vůči reálným bodům, se nazývá $Perspective - n - Point$. Hojně se vyskytuje v kalibraci kamer a aplikacích zabývajících se počítačovým viděním.
Vychází se z následujícího modelu:

$$sp_c = K \left[R|T\right] p_w \tag{5.3}$$

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{5.4}$$

kde $p_c = [uv1]^T$ je bod v obraze, $p_w = [xyz1]^T$ je příslušný reálný bod, K je matice parametrů kamery a R a T jsou hledané rotace a translace kamery. Základním předpokladem pro řešení problémů je zkalibrovaná kamera, tedy známá její kalibrační matice.
Pokud je počet bodů v prostoru a jejich obrazů roven 3, je to zároveň minimální forma PnP problému, označovaná jako P3P. Vyřešením P3P problému vzniknou 4 možná řešení rotace a translace, proto se přidává ještě čtvrtý bod, který z množiny řešení najde to nejvhodnější.
Pro řešení problému PnP s n $>$ ($\geq$) 3 body existuje více metod. Mezi nejznámější patří následující metody:

- Iterativní metoda založená na Levenberg-Marquadtově optimalizaci [22, 23] - funkce hledá takovou pozici, která minimalizuje chybu reprojekce, tj. sumu mocnic vzdáleností mezi body a jejich projekcemi.

- EPnP (Efficient) metoda představená trojicí Moreno-Noguer, Lepetit a Fua [24] říká, že každý bod (reálný, i jeho projekce) může být vyjádřen váženým součtem čtyř virtuálních kontrolních bodů. Souřadnice těchto kontrolních bodů jsou neznámé celého problému a pozice kamery se vypočte z nich. Metoda není iterativní a podle autorů je rychlejší a stabilnější než klasické iterativní metody, ve spojení s Gaussovou Newtonovou metodou optimalizace dokonce dosahuje velmi dobré přesnosti.

- RANSAC [25] metoda se v mnohém podobá iterativní metodě, ale předpokládá, že mezi daty jsou i nevyhovující data (např. špatně pospojované body a jejich projekce), která by neměla být brána v potaz.

Všechny tyto metody jsou implementované v knihovně OpenCV, v práci jsem využila funkci
`solvePnP(objectPoints, imagePoints, cameraMatrix, distCoeffs, rvec, tvec)`
, kde `rvec, tvec` jsou vektory obsahující rotaci a translaci, tj. pozici kamery. Funkce defaultně využívá iterativní metodu, což lze přenastavit dalším parametrem.

5.5 Grafické znázornění nalezeného vzoru

Detektor WhyCon v režimu GUI zobrazí na monitoru video z kamery, kde modrým kruhem vyznačí obrys nalezeného černého kruhu, hlavní a vedlejší poloosu nalezené elipsy-kruhu a také vedle vypíše vypočtené souřadnice kruhu vzhledem ke kameře.
V případě, že program detekuje pouze šachovnici a kruh ne, šachovnice je pak ve výsledném videu označena dvěma modrými úsečkami. Ty tvoří kříž se středem ve středu šachovnice a také se vedle nachází vypočtené souřadnice (5.5).

Obrázek 5.5: Grafické zvýraznění nalezené šachovnice

Kapitola 6

Regulační algoritmus

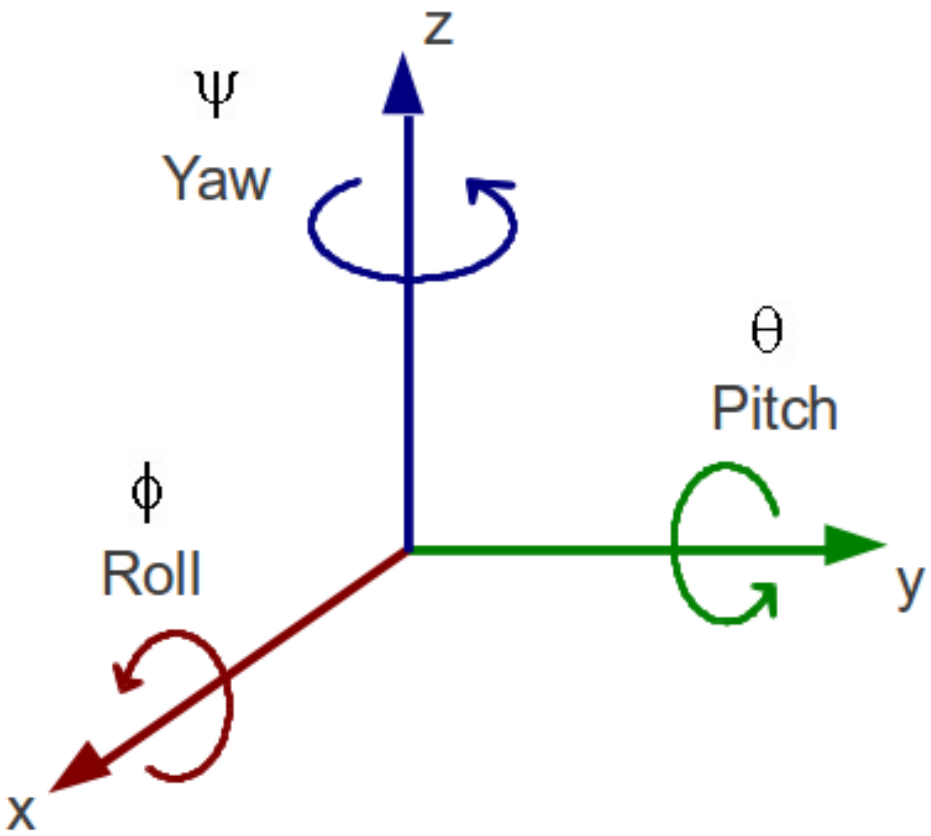
Pro experimentální účely se předpokládá použití hexakoptéry, tj. dronu se šesti vrtulemi na 6.1. Tato kapitola se zabývá jednoduchým modelem takového dronu a návrhem proporcionálního regulátoru určeného k přistávání dronu. Přestože se další text bude týkat zvolené hexakoptéry, řízení je prakticky stejné pro všechny podobné stroje (kvadrokoptéry, oktokoptéry).

Obrázek 6.1: Hexakoptéra a její souřadný systém

6.1 Model dronu

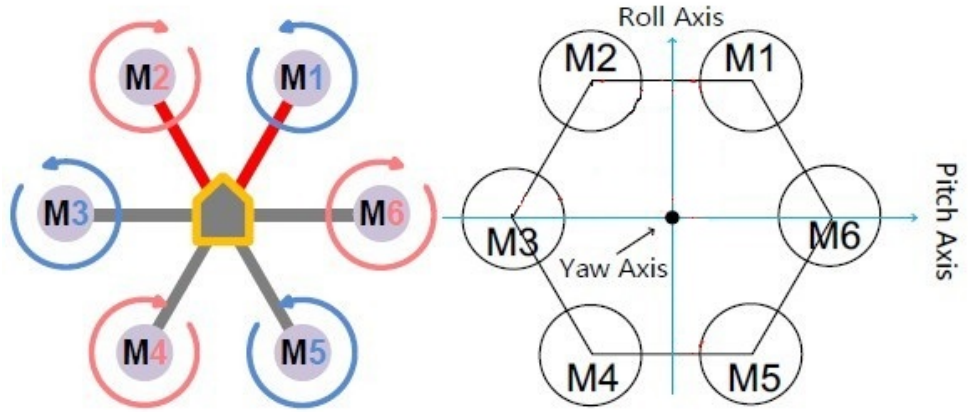
Jako každé volné pevné těleso v prostoru má dron 6 stupňů volnosti, 3 určují souřadnice x_R, y_R a z_R vzhledem k počátku referenční souřadné soustavy (střed vzoru na přistávací ploše), další 3 určují natočení os dronu (6.1, x červeně, y zeleně a z modře) vzhledem k osám referenční souřadné soustavy ($yaw : \psi =$ úhel otočení kolem osy z, $pitch : \theta =$ úhel otočení kolem osy y, $roll : \phi =$ úhel otočení kolem osy x, 6.2).

U dronů se sudým počtem vrtulí se většinou každé dvě sousední vrtule otáčí



Obrázek 6.2: Rotace kolem os x, y, z s příslušnými úhly [8]

v jiném směru. Tím je docíleno toho, že se při stejné rychlosti všech motorů navzájem vykompenzují síly otáčející dron okolo osy z (u klasické helikoptéry se otáčení okolo této osy řídí pomocí ocasní vrtule). Zvyšováním a snižováním rychlosti otáčení jednotlivých vrtulí se pak docílí naklonění dronu:



Obrázek 6.3: Označení motorů a os dronu [9]

- Pohyb vpřed/vzad vzniká, když se přední dva motory (M1 a M2 na 6.3) točí s menší/větší rychlostí než zadní motory (M4 a M5).
- Pohyb do strany vzniká, když se jeden boční motor (M3) točí pomaleji/rychleji než druhý na opačné straně dronu (M6).
- Pokud se všechny motory točí stejně rychle, dron klesá, stoupá, nebo zůstává ve stejné výšce v závislosti na celkovém tahu všech motorů a působení zejména gravitační síly.

Model dronu lze popsat Newton-Eulerovými rovnicemi [IRJET]:

$$\begin{bmatrix} mI_{3x3} & 0_{3x3} \\ 0_{3x3} & J \end{bmatrix} \begin{bmatrix} \dot{V} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} \omega & \wedge mV \\ \omega & \wedge J\omega \end{bmatrix} = \begin{bmatrix} \sum F \\ \sum M \end{bmatrix}, \tag{6.1}$$

kde m je hmotnost dronu, I_{3x3} jednotková matice, 0_{3x3} nulová matice, J matice setrvačnosti. $\sum F$ značí součet všech uvažovaných působících sil a $\sum M$ pak součet všech uvažovaných působících momentů.

Vektory V a ω jsou definovány následovně:

$$V = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T \tag{6.2}$$

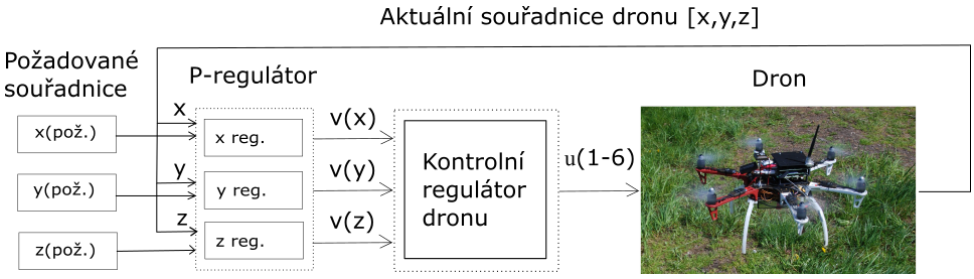
$$\omega = \begin{bmatrix} 1 & 0 & -sin\theta \\ 0 & cos\phi & sin\phi cos\theta \\ 0 & -sin\phi & cos\phi cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{6.3}$$

6.2 Řízení

Dron má 6 aktuátorů (= motorů), kterými lze řídit úhly natočení ψ, θ a ϕ a celkový tah. Tím se řídí vertikální pohyb dronu a zbylé dva stupně volnosti, určující translační pohyb v horizontální rovině, musí být řízeny regulátorem. Takové systémy, u kterých nemůže být kontrolním vstupem dosaženo libovolného stavu systému, nejsou plně řiditelné. V důsledku toho nemůže takový systém následovat libovolnou trajektorii [26]. Řízení těchto systémů je tak obtížnější než u plně řiditelných lineárních systémů. Při vhodné linearizaci nelineárního systému (kterým dron je) v okolí určeného pracovního bodu (například při vznášení v prostoru) lze zjednodušit model dronu až na systém prvního řádu.
Vzhledem k velkému množství dronů byly navrženy různé regulátory, které mají za cíl řídit trasu letu a zároveň stabilizovat dron ve vzduchu, tj. požadovaná hodnota úhlů θ a ϕ je rovna 0. Použitý model hexakoptéry obsahuje takový regulátor. Při manuálním řízení přijímá dron požadovaný směr a rychlost, které se vyhodnotí v regulátoru a ten pak řídí napětí jednotlivých motorů dronu ($u(1-6)$ na 6.4).
Při automatickém přistání se budou odlišovat 3 různé rychlosti. Zaprvé, vertikální rychlost sestupu v_z, tedy jak rychle bude dron klesat k přistávací ploše. Zadruhé, horizontální rychlosti v_x a v_y, s nimiž se budou korigovat x-ová a y-ová pozice dronu vůči přistávací ploše. Výsledné rychlosti předané systému z P-regulátoru (6.4) pak budou:

$$v_x = k_{xy}(x_d - x) \tag{6.4}$$
$$v_y = k_{xy}(y_d - y) \tag{6.5}$$
$$v_z = k_z(z_d - z), \tag{6.6}$$

kde k_{xy} a k_z jsou proporcionální konstanty určující rychlost regulace x-ové a y-ové souřadnice dronu a rychlost sestupu. Proměnné x_d,y_d a z_d vyjadřují požadovanou pozici dronu, při automatickém přistání se všechny rovnají 0 za předpokladu, že střed přistávací plochy má souřadnice $[0,0,0]^T$. V případě automatického vzletu dron vystoupá do předem určené výšky z_d a bude se držet nad přistávací plochou, proto jsou hodnoty $x_d = 0$ a $y_d = 0$ stejné jako při automatickém přistání (opět za předpokladu, že střed přistávací plochy má všechny souřadnice rovné 0).
V ideální situaci by při přistání dron klesal nejdříve rychle a postupně by zpomaloval až do momentu vypnutí motorů. V praxi by to ale mohlo znamenat, že dron rychlým manévrem ve velké vzdálenosti od přistávací plochy ztratí vzor ze zorného pole nebo že bude zachycený rozmazaně. Proto je vhodné omezit rychlosti v_x, v_y i v_z maximálními horizontálními rychlostmi v_{xmax}, v_{ymax} i rychlostí sestupu v_{zmax}.
Během automatického přistávání může nastat situace, že dron ztratí vzor na přistávací ploše ze zorného pole kamery nebo ho v obraze nedetekuje. To může být způsobeno například náhlým poryvem větru, který dron odnese



Obrázek 6.4: Schéma regulátorů

od přistávací plochy, nebo nevyhovujícím obrázkem z kamery. V takovém případě dron přeruší přistávání a vystoupá do takové výšky, ve které zase detekuje přistávací plochu nebo do předem nastavené maximální výšky z_{max}, a znovu zkusí přistát. Rychlost stoupání se v takovém případě zvyšuje postupně až do maximální vertikální rychlosti, která je rovna rychlosti v_{zmax} s opačným znaménkem.
Postup při přistávání je znázorněn v *Alg.* (3). Samotná implementace řídícího algoritmu se nachází v souboru `main.cpp`. V hlavní funkci se vytvoří nový proces, ve kterém se spustí program WhyCon. Oba procesy jsou propojené přes rouru (pipe), kam posílá lokalizační systém vypočtené souřadnice kamery. Rodičovský proces v opakujícím se cyklu načítá obdržené souřadnice a v p-regulátoru je přepočítá na požadované rychlosti. Cyklus skončí ve chvíli, kdy je dron při přistávání v minimální výšce nad přistávací plochou (nebo v předem určené výšce nad přistávací plochou, pokud jde o vzlet).

Algorithm 3: Přistávací algoritmus

Data: $([x_d, y_d, z_d])$: $[x_d, y_d, z_d]$ - desired dron position

$z_{min} \leftarrow$ minimum distance between the camera and the landing pad for the pattern possible to be detected
$z_{max} \leftarrow$ starting and maximum z-position of the drone
drone hovering above the landing pad
while *not at* z_{min} **do**
    get drone position $[x, y, z]$
    if *known(position)* **then**
        compute desired velocities $[v_x, v_y, v_z]$
        adjust x, y position
        fly down
    else
        compute desired velocities $[v_x, v_y, v_z]$
        fly up to z_{max}
turn off motors

Kapitola 7

Experimentální ověření implementovaných funkcí

S upraveným lokalizačním systémem byly provedeny příslušné experimenty, aby mohla být ověřena funkčnost implementací.
Nejprve byl proměřen výpočet vzdálenosti na obrazech z kamery, kde byl v obraze detekován celý vnější kruh i šachovnice. Vzor s průměrem černého kruhu 11,9 cm byl od kamery umístěn ve vzdálenosti od 25 cm do 50 cm. Absolutní rozdíly vypočtených hodnot pomocí detektoru kruhu i šachovnice jsou zaznamenány v tabulce 7.1. Z výsledků je patrné, že lokalizace pomocí detekce kruhu i šachovnice jsou natolik přesné, že přepnutí z detekce kruhu na detekci šachovnice a naopak nezpůsobí žádný problém související s významným rozdílem po sobě zjištěných hodnot souřadnic.

	x	y	z
25 cm	0.1071	0.0372	0.3710
30 cm	0.0302	0.0216	0.8229
35 cm	0.0587	0.0226	0.4577
40 cm	0.0943	0.0142	0.2519
45 cm	0.0620	0.0217	0.4946
50 cm	0.0955	0.0149	0.0101

Tabulka 7.1: Absolutní hodnoty rozdílu naměřených souřadnic x,y,z [cm]

Poté jsem vyzkoušela, jak program reaguje na přechod mezi detekováním kruhu a detekováním šachovnice v reálném čase. Ve vzdálenosti kamery od vzoru, kde byl ještě detekovatelný celý kruh, jsem spustila program a

	T [ms]	fps [-]	medián fps [-]
Detekovatelný vzor	38.4928	25.9789	24.5307
Chybějící vzor v obrázku	143.5132	6.9680	5.2963

Tabulka 7.2: Průměrná doba zpracování jednoho obrázku z webkamery WSD01 [10]

	T [ms]	fps [-]	medián fps [-]
Detekovatelný vzor	11.5964	86.2334	64.8378
Chybějící vzor v obrázku	377.8601	2.6465	2.6006

Tabulka 7.3: Průměrná doba zpracování jednoho obrázku z kamery mvBlueFOX-MLC [11]

postupně se s kamerou přibližovala, abych simulovala přibližování kamery při přistávání helikoptéry. Ve chvíli, kdy už kruh částečně přesahoval hranice kamery a nebyl detekovatelný, program plynule přešel na lokalizaci pomocí šachovnice.
Při simulaci vzletu (postupném oddalování kamery od vzoru) přešel program stejně plynule z detekce šachovnice na detekci kruhu.

Při experimentech byl upravený lokalizační systém WhyCon spouštěn na stolním počítači s procesorem Intel Core i7-3770 ([27]) a s kamerou IT WORKS WSD01 ([10], rozlišení 640x480 pixelů). Byla sledována průměrná doba jednoho *while* cyklu, ve kterém se načte obraz z kamery, provede lokalizace vzoru a výpočet vzdálenosti a následně se pozice kamery zapíše do výstupního souboru (output.log). Průměrná doba (perioda) tohoto cyklu T a počet zpracovaných obrázků za sekundu fps (frame-per-second) byly vypočteny pro situace, kdy vzor v obraze byl i nebyl detekovaný. V tabulce (7.2) jsou zaznamenány vypočtené průměrné hodnoty T a fps získané při experimentech s webkamerou WSD01. Použité obrázky jsou k dispozici na přiloženém CD ve složce `webcam`.

Ověření funkčnosti implementace bylo provedeno také pomocí fotografií z dronu. Ten byl ovládán manuálně a během letu nad travnatou plochou, kde byla umístěna přistávací plocha se vzorem, ukládal obrázky pořízené kamerou mvBlueFOX-MLC ([11], rozlišení 752x480 pixelů). Tyto obrázky jsou na přiloženém CD umístěné ve složce `bluefoxcam`.
Z fotografií zachycených kamerou mvBlueFOX-MLC se stejně jako v případě obrázků z webkamery WSD01 spočítaly průměrné hodnoty T a fps. Je ale nutné podotknout, že načítání obrázků bylo v obou případech různé (reálný přenos u webkamery, načítání uložených obrázků u kamery mvBlueFOX). Proto byla u fotografií z kamery mvBlueFOX, kde vzor nebyl detekovatelný, zvlášť změřená i doba lokalizace vzoru bez zpoždění řídící smyčky (7.4).
Naměřené průměrné hodnoty fps se od mediánu liší mnohem víc u obrázků

	T [ms]	fps [-]	medián fps [-]
Chybějící vzor v obrázku	360.4383	2.7744	2.6643

Tabulka 7.4: Průměrná doba detekčního algoritmu u fotografií z kamery mvBlueFOX bez detekovatelného vzoru

z kamery mvBlueFOX než v případě obrázků z webkamery. To je pravděpodobně způsobeno tím, že bylo oproti webkameře použito méně fotografií, které byly navíc pořízeny za různých světelných podmínek.
Největším problémem při detekci vzoru v obrazech z dronu se ukázal být sluneční svit, který kvůli velkému jasu znehodnotil obrázek (7.1) tak, že v něm vzor bez další úpravy obrázku nebyl detekovatelný. Na podobných obrázcích s vysokým jasem lze navíc vidět, že šachovnice ve středu vzoru vůbec není a místo ní je vnitřní kruh celý bílý, stejně jako u původního vzoru detektoru WhyCon (4.2). Tyto problémy se vyskytly pouze na obrázcích, které byly pořízeny při rychlém přeletu nad přistávací plochou. Obrázky, které byly pořízeny při manuálním přistání dronu na přistávací plochu a tedy při menší rychlosti dronu, obsahují detekovatelný vzor (7.2).
Pro automatické řízení je důležité zajistit, aby byly řídící signály vysílány v co nejpravidelnějších intervalech za sebou. Z experimentů s fotkami z dronu vyplývá, že za vyhovujících světelných podmínek a úspěšné detekce vzoru v obrázku je možné posílat signály pravidelně v krátkých intervalec . Jestliže ale selže detekce vzoru, dron nebude spolehlivě řiditelný kvůli velkým prodlevám mezi signály. Minimální frekvence signálů, které potřebuje vnitřní regulátor dronu k jeho řízení a stabilizaci, se pohybuje okolo 2 až 3 Hz, což je interval, kde se nachází i průměrné hodnoty fps (7.4).

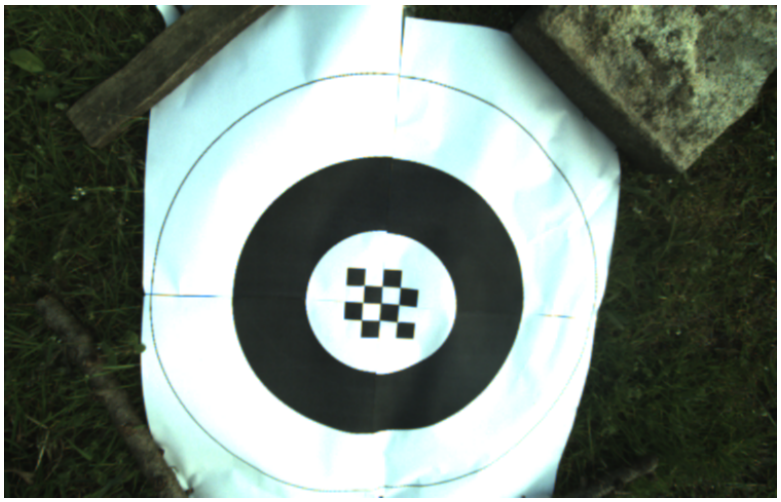
Obrázek 7.1: Obrázek pořízený z dronu při manuálně řízeném letu

Obrázek 7.2: Obrázek pořízený z dronu při manuálně řízeném letu

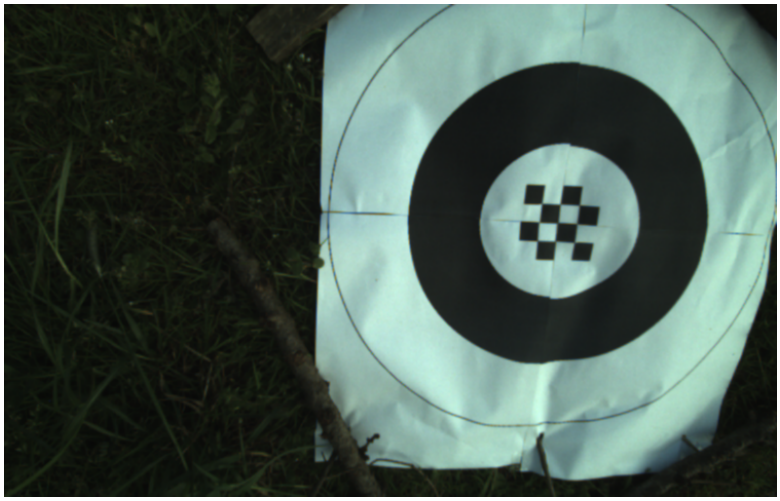
	T [ms] (šachovnice)	T [ms] (kruh)
Obrázky z webkamery WSD01	4.065	11.247
Obrázky z kamery mvBlueFOX	0.829	3.437

Tabulka 7.5: Průměrná doba detekce šachovnice a kruhu

Jako další bylo v lokalizačním algoritmu proměřeno, kolik času zabírá detekce šachovnice a samotného kruhu (7.5). Vzhledem k tomu, že se v detekci kruhu používá k jeho ověření nalezené šachovnice, byly pro měření vybrány obrázky s detekovatelnou šachovnicí i kruhem. Zároveň kvůli této závislosti nelze jednoznačně změřit čas detekce kruhu. Při tomto experimentu bylo u fotografií z kamer mvBlueFOX zjištěno, že doba detekce šachovnice je malá (řádově desetiny ms) na obrázcích s nižším stupněm jasu (7.4), zatímco kruh se rychleji detekuje na obrázcích s vyšším stupněm jasu (7.3).

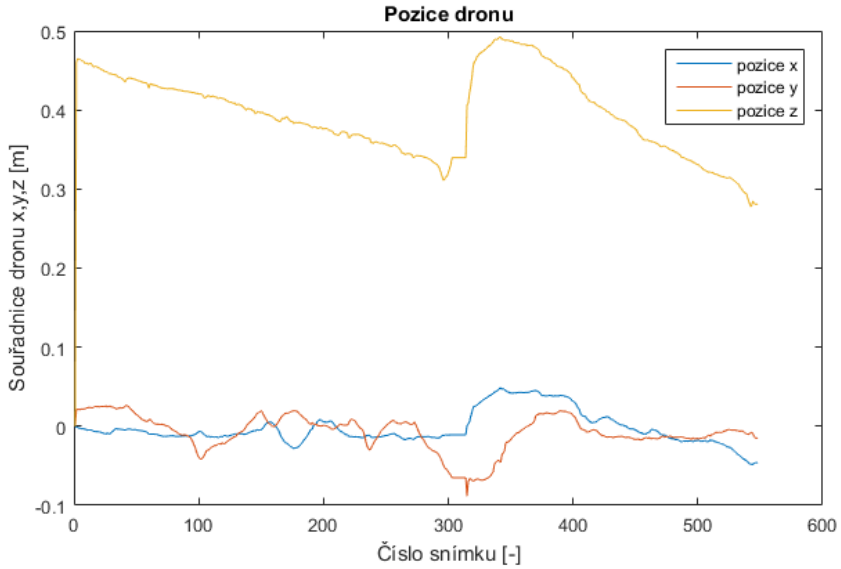


Obrázek 7.3: Obrázek s vyšším stupněm jasu pořízený z dronu

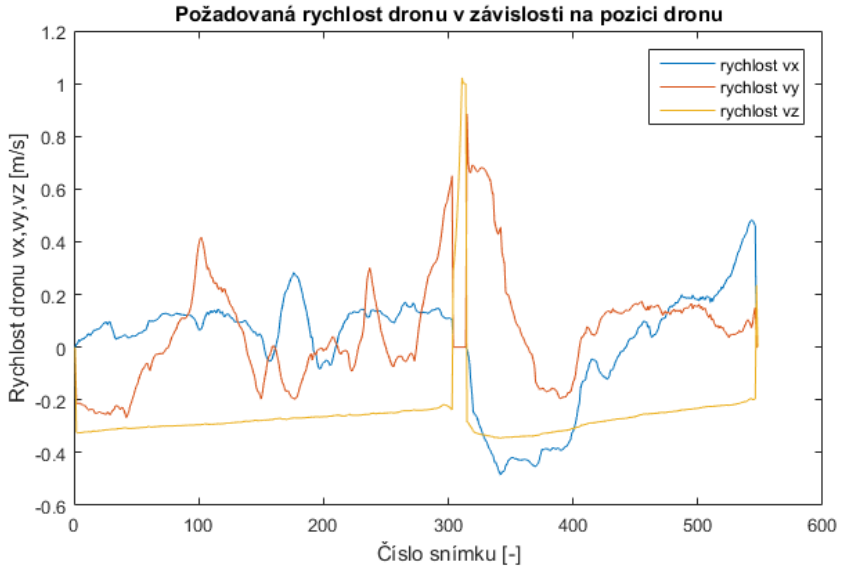


Obrázek 7.4: Obrázek s nižším stupněm jasu pořízený z dronu

Regulační algoritmus nebylo možné ověřit reálným experimentem, kde by se uskutečnilo automatické přistání dronu. Proto se pro ověření regulačního algoritmu simulovala situace přistávání jako postupné přibližování kamery k přistávací ploše. Regulátor reagoval na aktuální pozici dronu a generoval požadované rychlosti. Ve chvíli, kdy přestal dostávat souřadnice z detektoru WhyCon (tj. vzor nebyl nalezen), přepnul regulátor do režimu stoupání, a po další úspěšné detekci opět do režimu klesání.
Pozice kamery při simulaci i požadované rychlosti jsou vidět na 7.5 a 7.6. Náhlý výkyv rychlosti v_z v okolí snímku 300 označuje situaci, kdy nebyl vzor detekován a dron by měl stoupat zvyšující se rychlostí (až 1 ms^{-1}), než znovu detekuje vzor.



Obrázek 7.5: Průběh pozice dronu



Obrázek 7.6: Požadované rychlosti dronu ve směru os x,y a z v závislosti na aktuální poloze dronu

Kapitola 8

Závěr

Cílem této práce bylo navrhnout systém pro autonomní přistávání robotické helikoptéry na přistávací plochu s využitím obrazu z kamery umístěné na helikoptéře. Prostudovala jsem proto metody zpracování obrazu i metody pro vyhledávání specifických vzorů v obraze.
V projektu jsem vycházela z lokalizačního systému WhyCon ([6, 7, 17, 18]), který byl navržen jako jednoduché a nenáročné řešení lokalizačních úloh, což umožňuje jeho využití u robotů s omezenými výpočetními prostředky. Systém k lokalizaci využívá vlastní speciálně navržený vzor (4.2). Při automatickém přistávání dojde k okamžiku, kdy se část lokalizačního vzoru již nevejde do zorného pole kamery a stroj musí přistát naslepo. Proto jsem se přidala do středu vzoru ještě šachovnici o rozměrech 4x4 pole. Když se pak vzor ztratí ze zorného pole kamery, přejde lokalizační systém na výpočet polohy z detekované šachovnice a přistání naslepo proběhne z menší výšky. Řídící program pro automatické přistání byl napsán do zvláštního programu, který načítá detektorem vypočtenou pozici kamery.
Provedla jsem příslušné experimenty s fotkami z webkamery i z kamery umístěné na dronu, abych ověřila přidané funkce programu a přechod mezi detekcí šachovnice a detekcí vzoru WhyCon. Detektor byl schopen za stabilních světelných podmínek najít hledaný vzor v obraze a vypočítat relativní pozici kamery vůči přistávací ploše. Výpočty přitom proběhly v dostatečně krátkém čase na to, aby mohl být dron řízen pravidelnými signály. Použité fotografie i videa jsou k dispozici na přiloženém CD.

Příloha A

Literatura

[1] S. Saripalli, J.-F. Montgomery, and S. Sukhatme, "Visually-guided landing of an unmanned aerial vehicle," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 3, pp. 371–380, 2003.

[2] T. Merz, S. Duranti, and G. Conte, "Autonomous landing of unmanned helicopter based on vision and inertial sensing," tech. rep., Linköping University, Sweden, 2006.

[3] S. Lange, N. Sunderhauf, and P. Protzel, "Autonomous landing for multirotor uav using vision," tech. rep., Chemnitz University of Technology, Germany, 2008.

[4] Q. Wang, W.-C. Cheng, N. Suresh, and H. Hua, "Development of the local magnification method for quantitative evaluation of endoscope geometric distortion." `http://biomedicaloptics.spiedigitallibrary.org/article.aspx?articleid=2522079`, 2016.

[5] "Opencv tutorial: Image thresholding." `http://docs.opencv.org/trunk/d7/d4d/tutorial_py_thresholding.html`.

[6] M. Nitsche and T. Krajník, "WhyCon: Stable release, GitHub," May 2014.

[7] T. Krajník, M. Nitsche, J. Faigl, P. Vaněk, M. Saska, L. Přeučil, T. Duckett, and M. Mejail, "A practical multirobot localization system," *Journal of Intelligent & Robotic Systems*, 2014.

[8] "Joints." `http://doc.aldebaran.com/2-1/family/robots/joints_robot.html`.

[9] "Customize motor mixer." http://wiki.dji.com/en/index.php/Customize_Motor_Mixer.

[10] "It works wsd01 web camera." http://www.darty.com/nav/achat/image_son/photo_numerique/webcam/it_works_wsd01.html.

[11] "Usb 2.0 board-level camera - mvbluefox-mlc." https://www.matrix-vision.com/USB2.0-single-board-camera-mvbluefox-mlc.html.

[12] "Lockheed's new drone will help rescuers find missing people." https://www.wired.com/2015/04/lockheeds-new-drone-will-help-rescuers-find-missing-people/, 2015.

[13] M.-A. Olivares-Mendez, C. Fu, P. Ludivig, T.-F. Bissyandé, S. Kannan, M. Zurad, A. Annaiyan, H. Voos, and P. Campoy, "Towards an autonomous vision-based unmanned aearial system against wildlife poachers," *Sensors*, vol. 15, 2015.

[14] "Lights, camera, drones: Hollywood's lens gets a little larger." http://www.npr.org/sections/thetwo-way/2014/09/26/351731611/lights-camera-drones-hollywoods-lens-gets-a-little-larger, 2014.

[15] "What is camera calibration?." https://uk.mathworks.com/help/vision/ug/camera-calibration.html.

[16] M. Španěl, "Obrazové segmentační techniky." http://www.fit.vutbr.cz/~spanel/segmentace/, 2005.

[17] T. Krajník, M. Nitsche, J. Faigl, T. Duckett, M. Mejail, and L. Přeučil, "External localization system for mobile robotics," in *16th International Conference on Advanced Robotics (ICAR)*, Nov 2013.

[18] M. Nitsche, T. Krajník, P. Čížek, M. Mejail, and T. Duckett, "Whycon: An efficent, marker-based localization system," in *IROS Workshop on Open Source Aerial Robotics*, 2015.

[19] "Open source compute vision." http://docs.opencv.org/2.4.13/.

[20] C. Harris and M. Stephens, "Proceedings of the 4th alvey vision conference," in *A Combined Corner and Edge Detector*, pp. 147–151, August 1988.

[21] C. Danner and M. Kafafy, "Visual chess recognition," tech. rep., Stanford University, California, 2015.

[22] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Mathematics*, pp. 164–168, 1944.

[23] D. Marquadt, "An algorithm for least-squares estimation of nonlinear parameters," *SIAM Journal on Applied Mathematics*, pp. 431–441, 1963.

[24] V. Lepetit, F.Moreno-Noguer, and P.Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal Computer Vision*, vol. 81, no. 2, 2009.

[25] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Comm. of the ACM*, vol. 24, pp. 381–395, 1981.

[26] R. Tedrake, "Underactuated robotics: Learning, planning and control for efficient and agile machines," tech. rep., MIT, Massachusetts, 2009.

[27] "Intel core i7-3770 processor." `http://ark.intel.com/products/65719/Intel-Core-i7-3770-Processor-8M-Cache-up-to-3_90-GHz`.

Příloha B

Obsah CD

Přiložené CD obsahuje elektronickou verzi tohoto dokumentu a následující složky:

- webcam - materiály týkající se experimentů s webkamerou IT WORKS WSD01
- bluefoxcam - materiály týkající se experimentů s webkamerou mvBlueFOX-MLC
- whycon-chess - hlavičkové (include) a zdrojové (src) kódy upraveného lokalizačního systému WhyCon
- regulator - zdrojový kód řídícího programu (regulátoru) v souboru main.cpp

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Kateřina K u g l e r o v á

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Automatický vzlet a přistání robotické helikoptéry

Pokyny pro vypracování:

Navrhněte systém pro autonomní přistávání helikoptéry na přistávací plochu s využitím metod zpracování obrazu z kamery umístěné na helikoptéře. Prostudujte problematiku zpracování obrazu a zaměřte se na metody pro hledání a lokalizaci požadovaného vzoru. Vyberte nebo navrhněte vhodnou metodu a vizuální/technické provedení přistávací plochy pro helikoptéru. Implementujte algoritmy relativní lokalizace helikoptéry vůči heliportu a regulační algoritmus realizující automatické přistání.

Seznam odborné literatury:

[1] Tomáš Krajník, Matias Nitsche et al.: A Practical Multirobot Localization System. Journal of Intelligent and Robotic Systems (JINT), 2014.
[2] Milan Šonka a Václav Hlaváč: Počítačové vidění. Praha: Grada, 1992.

Vedoucí bakalářské práce: Ing. Jan Chudoba

Platnost zadání: do konce letního semestru 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 9. 1. 2017