České vysoké učení technické v Praze
Fakulta strojní
Ústav přístrojové a řídicí techniky

Diagnostika chyb s využitím strojového vidění

Diplomová práce

Bc. Jiří Kubica

Magisterský program: Průmysl 4.0
Magisterský obor: Bezoborový
Vedoucí práce: Mgr. Ing. Jakub Jura, Ph.D.

Praha, červenec 2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kubica** Jméno: **Jiří** Osobní číslo: **438349**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav přístrojové a řídící techniky**

Studijní program: **Průmysl 4.0**

Studijní obor: **bez oboru**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Diagnostika chyb s využitím strojového vidění

Název diplomové práce anglicky:

Machine Vision based Fault Diagnosis system

Pokyny pro vypracování:

1) Seznámit se s knihovnou pro strojové vidění OpenCV
2) Vytvořit jednoduchý program pro automatickou identifikaci rozsvícené kontrolky pomocí analýzy obrazu
3) Navrhnout jednoduchý diagnostický program pro systém logického řízení
4) Do diagnostického programu začlenit informace z analýzy obrazu

Seznam doporučené literatury:

[1] M. Hollender, Collaborative process automation systems. ResearchTriangle Park, NC: ISA, 2010, oCLC: ocn430843245.
[2] G. Van Rossum and F. L. Drake Jr., Python tutorial. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.
[3] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.
[4] O. Foundation, "OpenCV," 2019. [Online]. Available: https://opencv.org/
[5] "Object detection," in The OpenCV Reference Manual, release2.4.13.7 ed. OpenCV Team, 2019, pp. 338–340.
[6] T. Krajnık, M. Nitsche, J. Faigl, P. Vanek, M. Saska, L. Preucil,T. Duckett, and M. Mejail, "A practical multirobot localization system", Journal of Intelligent & Robotic Systems, 2014. [Online]. Available:http://dx.doi.org/10.1007/s10846-014-0041-x
[7] "Getperspectivetransform," inThe OpenCV Reference Manual, release 2.4.13.7 ed. OpenCV Team, 2019, pp. 270–271.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Mgr. Jakub Jura, Ph.D., U12110.3

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Ing. Matouš Cejnek, U12110.3

Datum zadání diplomové práce: **30.04.2020** Termín odevzdání diplomové práce: **07.08.2020**

Platnost zadání diplomové práce: _____

Ing. Mgr. Jakub Jura, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Michael Valášek, DrSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

4. 8. 2020 Datum převzetí zadání	Podpis studenta

Vedoucí práce:
Mgr. Ing. Jakub Jura, Ph.D.
Ústav přístrojové a řídicí techniky
Fakulta strojní
České vysoké učení technické v Praze
Technická 2
160 00 Praha 6
Česká republika

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně s tím, že její výsledky mohou být dále použity podle uvážení vedoucího diplomové práce jako jejího spoluautora. Souhlasím také s případnou publikací výsledků diplomové práce nebo její podstatné části, pokud budu uveden jako její spoluautor.

V Praze 2020

.....
Bc. Jiří Kubica

Abstrakt

Tato práce se zabývá návrhem systému pro diagnostiku a identifikaci chyb ve výrobních systémech s využitím nástrojů strojového vidění. Součástí práce je rešerše vybraných metod strojového vidění vhodných pro analýzu obrazu zkoumaného výrobního procesu. Navržený diagnostický systém používá k vyhodnocení chybových stavů zkoumaného procesu proměnné určené analýzou obrazu a proměnné získané z PLC, které daný proces řídí. Tato data jsou použita v diagnostickém modulu, který je díky většímu množství vstupů schopen přesněji identifikovat příčinu vzniklé chyby. Práce popisuje jednotlivé moduly a funkce navrženého systému nejen z pohledu vnitřní struktury použitých algoritmů, ale poskytuje i návodný přehled možností a ovládání systému pro koncového uživatele se zaměřením na praktickou aplikaci. V závěru práce je systém otestován na jednoduchém laboratorním modelu.

Klíčová slova: diagnostika chyb, identifikace chyb, strojové vidění, automatické řízení, OpenCV

Abstract

This thesis deals with the design of a fault diagnostic and identification system in production systems using machine vision tools. The work includes a research of machine vision methods suitable for the analysis of the image of the investigated production process. The proposed system uses variables from image analysis and variables obtained from PLC for the fault state evaluation of the investigated process. This data is processed in the diagnostic module, which is now able to accurately identify the cause of errors due to the increased amount of inputs. The thesis describes the modules and functions of the proposed system from the point of view of structures of these algorithms. It also provides the user with instructions on the various options and settings of the system with a focus on practical use. At the end of the work, the system is tested on a simple laboratory model.

Keywords: Fault diagnosis, Fault identification, Machine vision, automatic control, openCV

Poděkování

Děkuji Mgr. Ing. Jakubovi Jurovi, Ph.D., za jeho konzultace, rady a podněty, které mi během vypracovávání této práce poskytl. Dále děkuji Ing. Matouši Cejnkovi za jeho ochotu, plodné diskuse a odborné rady.

Děkuji své rodině, přátelům a přítelkyni za velkou trpělivost a stále trvající podporu během celého studia.

Seznam tabulek

Seznam obrázků

Seznam zkratek

BF Brute-force. 15

BRIEF Binary Robust Independent Elementary Features. xii, 14, 15

CAFDIS Camera Aided Fault Diagnosis and Isolation System. viii, xiii, 2, 3, 27, 28, 30, 32–34, 36–39, 42, 43, 46–48, 52, 55–57, 60–64, 66–69, 74, 75

CLAHE Contrast Limited Adaptive Histogram Equalization. 37

CNN Konvoluční neuronové sítě. 19–22, 26, 28

CRC Cyklický redundantní součet. 42

CV Strojové vidění. 1, 3, 4, 27, 74

DFT Diskrétní Fourierova transformace. 39

FAST Features from Accelerated Segment Test. viii, xii, 13, 15, 24

FLANN Fast Library for Approximate Nearest Neighbors. 15

FMS Systémy referenčních tagů. viii, xii, 4, 5, 7, 39, 74

GUI Grafické uživatelské rozhraní. xiii, 3, 34, 36–38, 40, 42, 43, 45, 46, 48, 49, 55, 60–64, 67, 69, 74, 75

HSV Hue, Saturation, Value. 18, 19, 59

ID Identifikátor. 56

LDA Linear Discriminant Analysis. 14

LED Světlo emitující dioda. ix, 2–4, 28, 30, 32, 60, 68–73, 75

LoG Laplacian of Gaussian. 11, 12

ORB Oriented FAST and Rotated BRIEF. xii, 14, 15, 25

OS Operační systém. 61

PCA Principal Component Analysis. 14

PLC Programovatelný logický kontroler. viii, ix, xiii, 1–4, 34, 47, 48, 60, 62, 63, 68–70, 72–75

RGB Červená, zelená, modrá. 18, 60

RPN Region Proposal Network. 22

SIFT Scale-Invariant Feature Transform. viii, 11, 12, 14

SSD Single Shot Detection. 22

SURF Speeded Up Robust Features. xii, 12, 14

TM Template Matching. 16, 26, 39, 43–45, 66, 67, 74

YOLO You Only Look Once. 22

Obsah

1 Úvod

Diagnostika chyb ve výrobních procesech a systémech je důležitou součástí průmyslového inženýrství. Včasná detekce chyby a její příčiny může zabránit vysokým ekonomickým ztrátám způsobeným pozastavenou výrobou. Současné trendy fenoménu průmysl 4.0 udávají jasný směr automatizace nejen ve výrobě, ale i v dalších systémech spjatých s podnikem. Celý proces detekce, identifikace a odstranění chyb však v současnosti ve výrobních systémech nebývá zcela automatizován.

Diagnostické systémy pracující na úrovni PLC a dalších kontrolerů jsou schopny detekovat většinu předvídatelných chyb. Identifikaci příčiny chyby poté provádí odborný pracovník. Cílem této práce je navrhnout diagnostický systém, který díky automatické identifikaci chyby pomůže zkrátit čas diagnostické práce.

Pro úspěšnou identifikaci chyby je nutné rozšířit množinu vstupních proměnných do diagnostického modulu. Prosté zdvojení senzorů použitých ve zkoumaném procesu je jednou z možností. Je zřejmé, že tento přístup není kvůli možným fyzickým omezením procesu (zástavbový prostor, integrované senzory) vždy možný. Pokud jeden z dvojce senzorů přestane poskytovat validní data, systém nemusí být bez dalších informací schopen identifikovat vadný senzor z dané dvojce.

S rostoucí popularitou a dostupností nástrojů strojového vidění (CV) se nabízí možnost jejich využití k rozšíření množiny vstupů do diagnostického modulu. Díky povaze většiny výrobních systémů lze pomocí kamery kontrolovat funkci akčních členů a některých senzorů. Toto řešení navíc umožňuje přístup ke zcela novým informacím získaným analýzou obrazu. Je možné například ověřovat přítomnost výrobku a jeho pozici ve výrobním procesu.

Pořízení kamer pro monitorování výroby není v porovnání s náklady na jiná výrobní zařízení nebo senzoriku drahou záležitostí. Zvláště uvážíme-li ekonomické úspory, které tento systém může přinést.

I přes již dlouho známé kamerové senzory a systémy používané na detekci jednotlivých chyb [1][2] zatím nevznikl ucelený systém s využitím CV, který by sloužil jako hlavní diagnostický nástroj celého výrobního procesu.

Cílem této práce je návrh diagnostického systému využívajícího informace z obrazů z kamer a jeho následná demonstrace na jednoduché laboratorní úloze.

2 Návrhová specifikace

2.1 Deklarace záměrů

Camera Aided Fault Diagnosis and Isolation System (CAFDIS) je systém pro podporu diagnostiky chyb v průmyslových výrobních zařízeních. Systém je určen pro úzká místa provozu a má za úkol odhalit přesnou příčinu vzniklé chyby, čímž pomáhá ušetřit čas následné opravy. Systém získává potřebná data mimo jiné z několika kamer, které vyšetřované místo monitorují. Uživatelské rozhraní systému umožní pověřenému uživateli konfigurovat systém pro konkrétní průmyslový proces a nastavit diagnostiku tak, aby dokázala identifikovat jednotlivé chyby procesu. Pokud v monitorovaném procesu chyba nastane, CAFDIS ji identifikuje a zobrazí její popis v uživatelském rozhraní. Systém je do značné míry imunní vůči neočekávanému posunutí kamery a vibracím.

Požadavky kladené na systém:

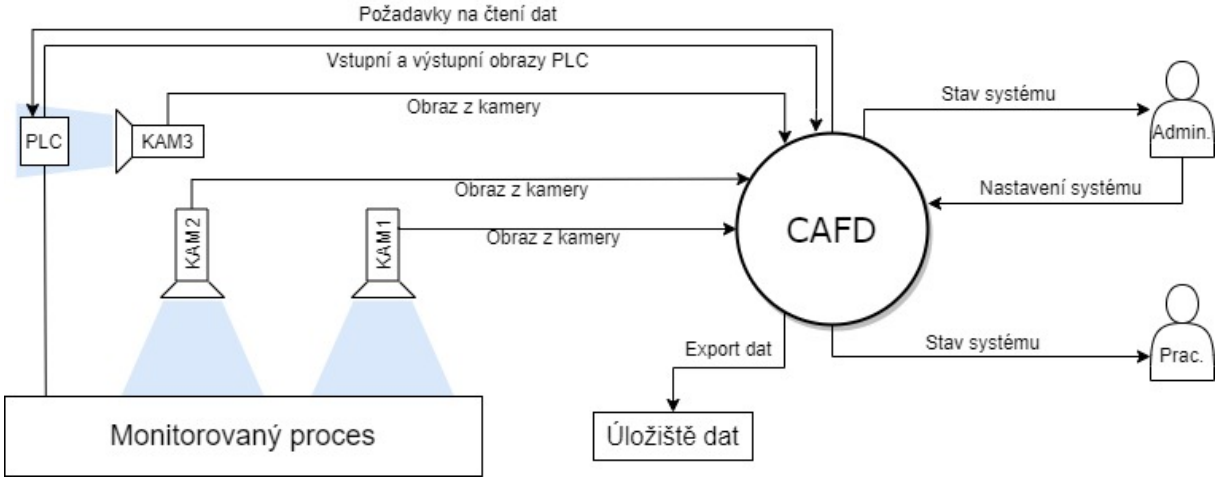
- spolehlivost,
- uživatelská přívětivost a přehledné nastavení a ovládání systému,
- necitlivost na přesnou polohu kamer, imunita vůči posunutí kamer a částečná odolnost proti vibracím,
- vyhodnocování událostí v reálném čase,
- přenositelnost na jiná zařízení a platformy,
- omezená potřeba výpočetního výkonu, nutnost plynulého fungování i na běžně dostupných zařízeních.

2.2 Základní návrh

Základní koncepce systému je popsána pomocí kontextového diagramu na obrázku 2.1.

K systému je možné připojit sérii kamer, které monitorují zkoumaný proces tím způsobem, aby bylo ze snímků možné extrahovat dodatečné informace potřebné k diagnostice a identifikaci příčiny chyby. Každý relevantní PLC je také monitorován kamerou: díky tomu lze kontrolovat stavy fyzických vstupů a výstupů PLC skrze stavy indikačních LED na těle PLC.

Konfiguraci a nastavení systému provádí pověřený pracovník s přidělenou rolí Administrátor. Stav diagnostiky procesu je možné zobrazit širším okruhem pracovníků.



Obrázek 2.1: Kontextový diagram systému CAFDIS

Systém CAFDIS ke svému správnému fungování potřebuje přístup k informacím ze zkoumaného procesu. Mezi tyto informace patří:

- Snímky z kamer monitorujících zkoumaný proces. Je především potřeba zachytit polohy aktuátorů a stavy indikačních LED na senzorech a PLC.
- Proměnné z PLC řídící monitorovaný proces, a to především vstupní a výstupní obrazy PLC.
- Konstanty procesu a informace o struktuře procesu poskytnuté prostřednictvím pověřeného pracovníka. Zejména informace o propojení jednotlivých komponent procesu mezi sebou.

Snímky z kamer je potřeba dále zpracovat nástroji CV, čímž je možné extrahovat data o stavu monitorovaného procesu. Možností, jak zpracování snímků provést, je celá řada. Nejvýznamnější z nich budou popsány v kapitole 3. Způsob získání dat z PLC značně závisí na typu PLC a pro různé výrobce bude řešení odlišné – podrobněji se komunikací s PLC zabývá kapitola 6.3. Poslední kategorií dat jsou data neměnného charakteru, která popisují zapojení a strukturu procesu, stejně tak jako informace o přípustných a chybových stavech, včetně jejich pravděpodobné příčiny. Tyto informace CAFDIS získá od pověřeného pracovníka prostřednictvím grafického uživatelského rozhraní (GUI).

3 Metody detekce objektu

S ohledem na účelné navýšení počtu vstupních proměnných do diagnostického modulu je na snímcích z kamer potřeba především detekovat:

- pozice aktuátorů a jiných pohyblivých částí,
- stavy indikačních LED na senzorech,
- stavy indikačních LED na PLC.

Aktuátory používané v praxi nabývají rozličných tvarů a velikostí, identifikovat nečinný aktuátor může proto být často obtížné i pro zkušeného člověka. Pokud se aktuátor začne pohybovat, lze pro jeho identifikaci pomocí CV s výhodou použít některý z nástrojů určený pro detekci pohyblivých objektů. Pokud bude snímek z jedné kamery zachycovat větší množství aktuátorů, které se budou ještě navíc pohybovat současně, může být separace jednotlivých prvků komplikovaná natolik, že bude výhodnější použít jinou metodu CV nebo kombinaci metod.

Detekce stavu indikačních LED na senzorech a PLC se může zdát jako prostá úloha na detekci barvy. Avšak jelikož je imunita systému vůči posunutí kamer jedním z požadavků z deklarace záměrů, je nutné polohu LED určovat jinak než v absolutních souřadnicích snímku.

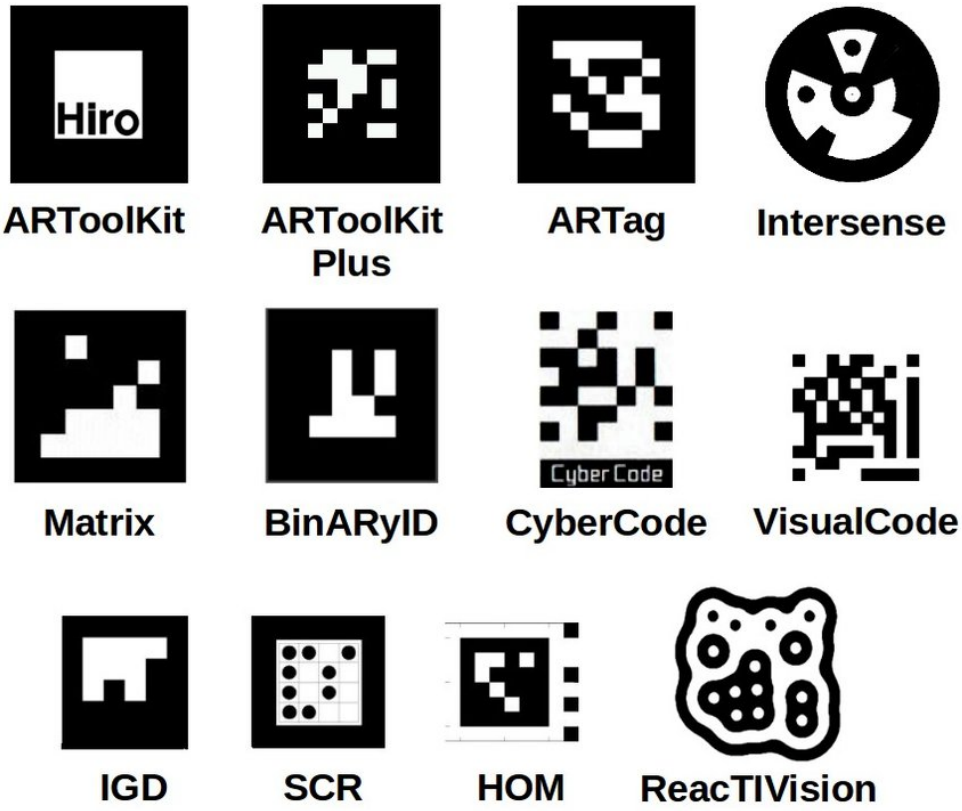
Tato kapitola obsahuje výčet a stručný popis významných metod CV, které by bylo možné na některý z daných problémů aplikovat.

3.1 Systémy referenčních tagů

FMS *(Fudical Marker Systems)* jsou soubory algoritmů používané na detekci speciálních referenčních tagů pomocí nástrojů strojového vidění. Referenční tagy používané těmito systémy mají mnoho společných rysů.

Každý tag se skládá ze dvou základních částí. První je obrys tagu, který je většinou pro všechny tagy v dané FMS skupině stejný. Nejčastěji představuje některý ze základních geometrických tvarů. Nejběžnější jsou tagy čtvercové nebo kruhové. Druhá část tagu je výplň obrysového tvaru. Typ výplně je dán použitým FMS. Výplň slouží k ověření, zda nalezený obrysový tvar je opravdu jedním z hledaných tagů, a pomáhá předcházet falešně pozitivním nálezům. Výplň může pomoci rozlišit jednotlivé tagy ve skupině tím, že bude představovat unikátní druh identifikace pro každý použitý tag. Používané tagy

jsou černobílé. Omezení složitosti tagu na dvě barvy umožňuje detekčním algoritmům používat binární prahové filtry barev, tedy každému pixelu zkoumaného obrazu přiřadit jen černou nebo bílou barvu. Tím se nejen sníží výpočetní náročnost celého algoritmu (jelikož se pracuje s menším množstvím dat), ale do značné míry se také daří eliminovat vliv světelných podmínek na výsledek celé detekce. Prahové filtry používané FMS nepracují s fixní prahovou hodnotou, nýbrž jsou adaptivní. Lze předpokládat, že takovýto filtr bude fungovat správně, i pokud se světelné podmínky budou v rámci jednoho vyšetřovaného snímku měnit, a to jak v čase, tak i v souřadnicích na obraze. [3][4]



Obrázek 3.1: Ukázka používaných tagů pro FMS [5]

3.1.1 ARToolKit [4]

Nejstarším systémem pro detekci referenčních tagů je systém ARToolKit. Přestože tento systém už byl překonám, dodnes se používá v některých méně náročných aplikacích. Jeho přímými potomky jsou systémy ARTag a později i ARToolKit Plus, které přinesly nejen spolehlivější detekci, ale i rychlejší běh celého algoritmu.

ARToolKit tagy používají černý čtverec na bílém pozadí. Výplň čtverce musí být černobílá a tvoří ji libovolná grafika, kterou si volí sám uživatel. FMS jako první detekuje všechny kontury tvořící černý čtyřúhelník. Všichni kandidáti na ARToolKit tag jsou vyhodnoceni pomocí korelace referenční výplně uložené v systému a výplně čtyřúhelníku

na vyšetřovaném snímku. Orientace tagu se zjistí jednoduchým výpočtem korelačního koeficientu ve všech čtyřech myslitelných směrech. Nejvyšší hodnota korelačního koeficientu je potom porovnána s nastavenou nejmenší přípustnou hodnotou, kandidát na tag je následně zamítnut nebo schválen.

Výhodou tohoto přístupu je teoreticky libovolné množství možných tagů, které se mohou jednotlivým algoritmem detekovat. Výpočetní náročnost metody ale roste s počtem hledaných tagů. ARToolKit tagy jsou díky svému statistickému přístupu k vyhodnocení také odolnější vůči okluzi výplně tagu, obrys tagu ovšem musí zůstat viditelný.

Obrázek 3.2: Ukázka ARToolKit tagů [4]

3.1.2 ARTag [4]

Oproti svému předchůdci využívá systém ARTag pro detekci základního obrysu metodu založenou na detekci hrany. Původní ARToolKit hledal obrysy tagů pomocí prohledávání morfologie celého černobílého obrazu. Výplň tagu je zastoupena digitálním binárním kódem, který slouží k odlišení správných nálezů od chybných kandidátů na ARTag, stejně jako k identifikaci štítků mezi sebou.

Binární kód systému ARTag obsahuje několik redundantních bitů pro detekci a následnou korekci některých chyb v rozpoznaném tvaru výplně. Díky tomu je algoritmus robustnější a umožňuje správnou detekci i při částečné okluzi, horší kvalitě obrazu nebo špatných světelných podmínkách. Správná orientace tagu je zjištěna prostým pokusem o dekódování binárního kódu ve všech čtyřech myslitelných směrech. Díky redundantním bitům je čtení úspěšné jen v jednom směru.

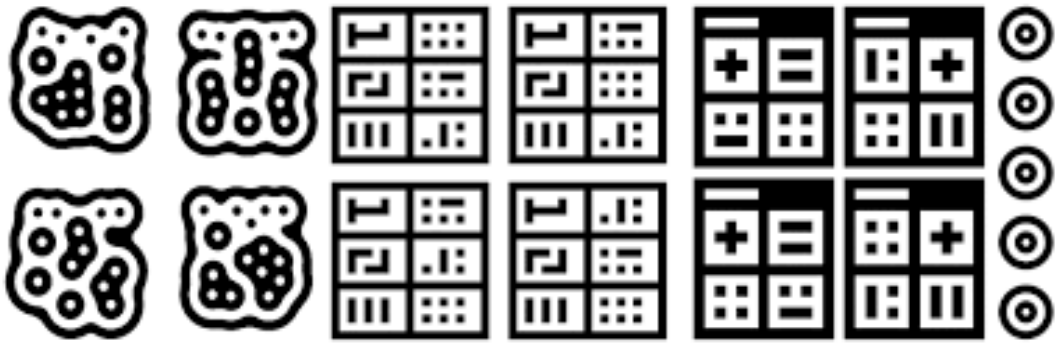
Výplň tagu byla inspirována tehdy novým 2D kódem *Datamatrix* [6], jehož principy si ARTag v nepříliš pozměněné míře osvojuje.

3.1.3 ARToolKit Plus

Přestože ARToolKit Plus vznikl až po vychvalované řadě ARTag, vrací se svým fungováním k původnímu systému ARToolKit. Zlepšením bylo použití spolehlivějších a rychlejších algoritmů na detekci morfologie tagu a jeho výplně. Oproti staré řadě, která používala tagy v odstínech šedi, pracuje pouze s černou a bílou. [4]

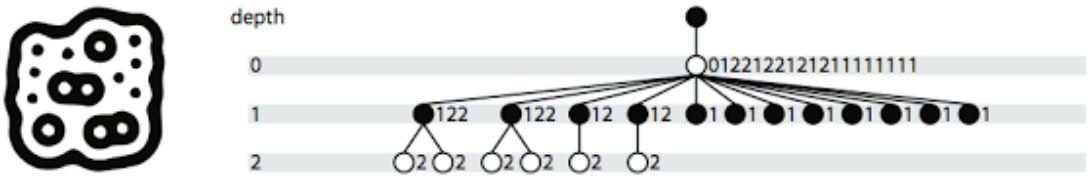
3.1.4 ReacTIVision

Skupina FMS, do které systém ReacTIVision patří, používá k detekci tagu jiných principů než jakýkoli z jeho předchůdců. ReacTIVision tagy nemusí a většinou nemají pevně daný geometrický tvar. Detekce totiž neprobíhá na základě detekce hran či tvaru. Organicky vypadající tagy se označují jako améby. [7]



Obrázek 3.3: Ukázka ReacTIVision tagů [8]

Vyšetřovaný obraz je nejprve filtrován pomocí adaptivního prahového filtru. Získaný černobílý obraz je podroben topologické analýze a následně je sestaven graf hierarchie kontur. Graf hierarchie kontur je orientovaný graf udávající souřadicí a podřadicí vztahy černých, resp. bílých regionů v obrázku mezi sebou. Jednotlivé uzly grafu představují souvislé černé, resp. bílé oblasti ve vyšetřovaném obrázku. Hrany mezi uzly značí, že nadřazená oblast obsahuje oblast podřazenou. [7]



Obrázek 3.4: Graf hierarchie kontur pro ReacTIVision tag [9]

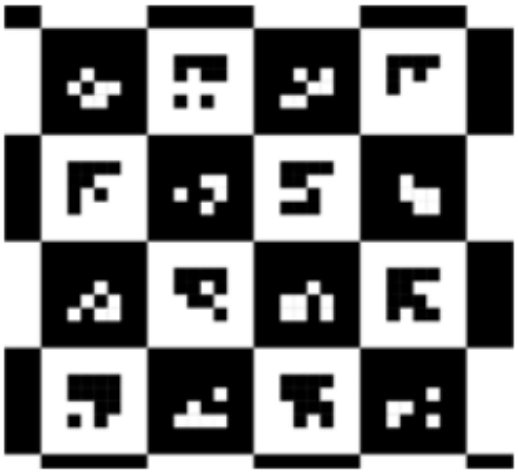
Jednotlivé tagy systému ReacTIVision mají unikátní hierarchii kontur, která je navíc dostatečně komplexní, aby nedocházelo k falešným nálezům ve vyšetřovaném snímku. Při samotné detekci se porovnávají grafy tagů, které má systém najít, s grafem vyšetřovaného snímku. Poloha tagu je potom nejčastěji určena jako těžiště obdélníku ohraničujícího vnější konturu tagu. Orientace se určuje podstatně hůře. Jednou z možností je vyhodnocení těžišť obdélníků opsaných i vnitřním konturám nalezeného tagu, samozřejmě za předpokladu, že se vnitřní kontury tagu svými topologickými grafy liší, a lze je tedy jednoznačně identifikovat. Na základě vzájemné polohy jednotlivých těžišť je vyhodnocena orientace celého tagu. [9]

Kvůli použitým principům je systém částečně chráněný proti okluzi. Jedinou nutnou podmínkou je, že všechny uzavřené kontury se musí opět detekovat jako uzavřené. Pokud tato podmínka nebude z důvodu podmínek měření splněna nebo dojde ke „slití“ stejně barevných oblastí tagu dohromady, detekce nemůže proběhnout správně. Systém ve svém základu neobsahuje žádný typ detekce nebo opravy chyb, jako tomu je například u ARTagu.

Díky své jednoduchosti je systém nenáročný na výpočetní výkon. Jelikož není k detekci potřeba hledat hrany, lze tagy úspěšně detekovat i na značně rozmazaných snímcích. [8] [7]

3.1.5 CALTag [10]

Systém tagů CALTag vznikl původně jako nástroj na spolehlivější kalibraci kamer. CALTagy mají šachovnicový charakter. Poloha tagu je popsána souřadnicemi všech rohů mezi díly šachovnice, tedy více parametry, než je pro přesné určení polohy potřeba. Jelikož je možné detekovat hrany a rohy s podpixelovou přesností, používají se souřadnice těchto rohů právě pro účely kalibrace. Aby byla detekce spolehlivá, je každý díl šachovnice vybaven binárním 2D kódem.



Obrázek 3.5: Ukázka tagu systému CALTag [9]

Systém byl navržen tak, aby byl co nejodolnější vůči okluzi a změně světelných podmínek. Toho je dosaženo pomocí metody zpětné rekonstrukce celého tagu. Každé pole šachovnice funguje jako samostatný tag detekovaný podobným způsobem jako například systém ARTag. Jelikož je architektura celé šachovnice detekčnímu systému známá, lze z polohy jen pár šachovnicových polí určit polohu tagu jako celku. V okolí míst, kde je očekáván roh šachovnicového pole, které nebylo úspěšně detekováno, je posléze spuštěna detekce znovu. Případný úspěšný nález rohu je uložen jako roh daného pole i bez úspěšné detekce onoho konkrétního pole.

Jedinou významnější nevýhodou tohoto systému je větší složitost a velikost používaných tagů. Pro detekci menších tagů je proto potřeba kamera s lepším rozlišením, aby byla zajištěna dostatečná kvalita obrazu hledaného tagu.

CALTagy se používají nejen k účelu, pro který byl tento systém vytvořen: v dnešní době udávají referenční pozice pro nejrůznější systémy virtuální nebo rozšířené reality.

3.2 Detekce charakteristických rysů

Charakteristické rysy (z angl. *Features*) jsou informace používané k vyřešení výpočetní úlohy v kontextu strojového vidění. Charakteristické rysy mohou být reprezentovány specifickými strukturami ve vyšetřovaném obrázku. Struktury vhodné pro tento účel jsou především hrany, rohy a další oblasti, kde se intenzita pixelů při posunu v jednotlivých směrech náhle mění. Na základě jedinečnosti těchto struktur je pak například možné srovnávat různé obrazy mezi sebou a hledat podobné objekty, které se na nich vyskytují. Charakteristické rysy mohou být také reprezentovány výsledkem algoritmů a matematických operací aplikovaných na danou oblast zkoumaného obrazu. [11]

Detekční algoritmy pracující s charakteristickými rysy mají 3 základní části:

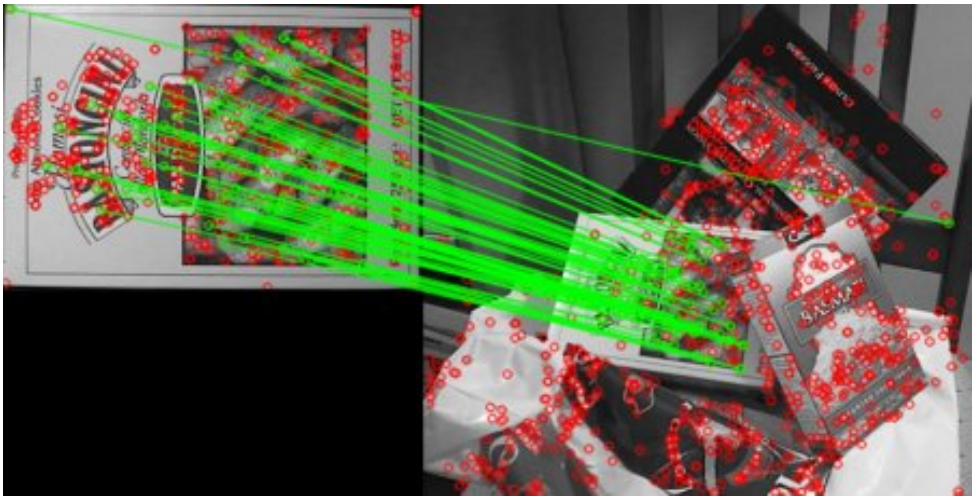
1. Identifikace bodů a oblastí vhodných pro sestavení charakteristických rysů.

2. Sestavení deskriptoru charakteristických rysů. Deskriptor popisuje charakteristický rys v řeči vhodné pro další zpracování. Nejčastěji v podobě matic, vektorů nebo například binárních řetězců.

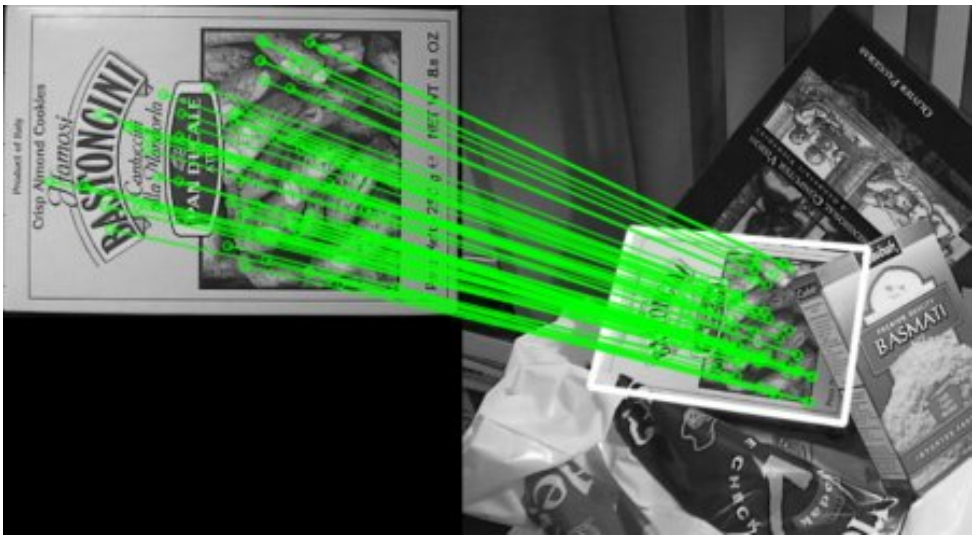
3. Nalezení shody mezi deskriptory hledaných objektů a deskriptory ze zkoumaného obrazu. Pokud se podaří najít shodu většího počtu deskriptorů, algoritmus se pokusí hledaný objekt připasovat na zkoumaný obraz.

Na obrázku 3.6 je znázorněna ukázka detekce objektu pomocí charakteristických rysů. V levé části obrázku je fotografie objektu. Algoritmus určí na objektu charakteristické rysy a jejich deskriptory. Poté je algoritmu předán snímek z pravé části obrázku 3.6, na kterém jsou provedeny stejné operace. Následně jsou hledány shody deskriptorů hledaného objektu a obrázku, které jsou znázorněny zelenou úsečkou. Z dostatečného množství dvojic shodných deskriptorů je poté vyhodnocena poloha a velikost objektu na zkoumaném snímku, tak jak to znázorňuje obrázek 3.7.

Algoritmů, které řeší detekci charakteristických rysů, vznikla celá řada. Liší se spolehlivostí, robustností a zejména rychlostí. Následuje výčet některých z nich.



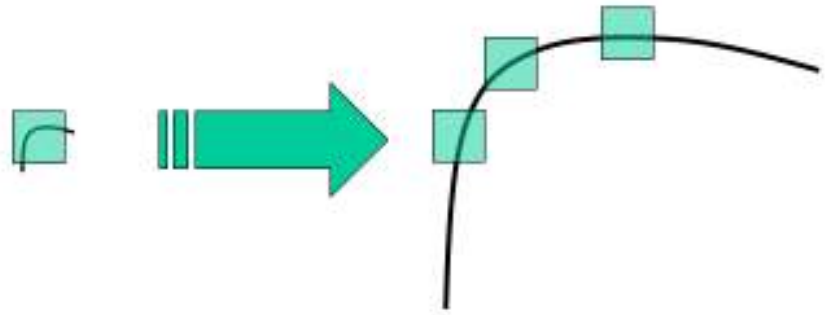
Obrázek 3.6: Ukázka detekce charakteristických rysů [12]



Obrázek 3.7: Nalezení polohy objektu pomocí charakteristických rysů [12]

3.2.1 SIFT [13]

Měřítkově invariantní algoritmus na popis charakteristik obrazu *Scale-Invariant Feature Transform* (SIFT) vychází z Shi-Tomasiho algoritmu na detekci hran a rohů [14]. Detektory rohů používají detekční masku, kterou umístí na vyšetřovaný obraz a sledují intenzitu barev na dané oblasti. Pokud se při posunu masky v různých směrech intenzita zásadním způsobem mění, algoritmus vyhodnotí danou oblast jako hranu nebo roh. Algoritmy tohoto typu jsou citlivé na dostatečnou „ostrost“ rohu – pokud je roh více oblý, je detekován jako hrana. Na obrázku 3.8 představuje zelený čtverec detekční masku. Levá část detekuje roh správně, v případě pravé části dojde jen k detekci hrany.

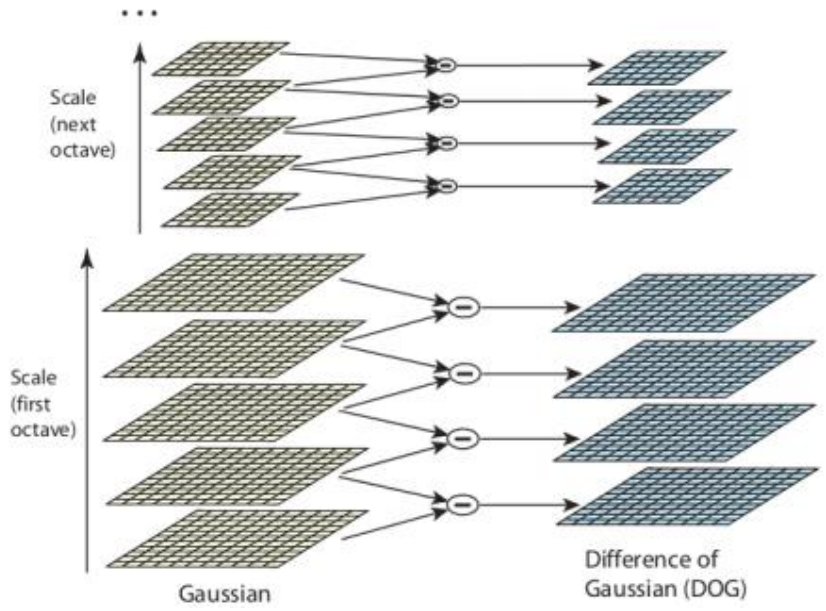


Obrázek 3.8: Citlivost detektoru na zaoblení rohu [15]

Jelikož před samotnou aplikací masky je obraz filtrován pro odstranění šumu, jsou pro urychlení celého procesu filtrování a výpočet intenzity spojeny do jednoho kroku pomocí LoG (*Laplacian of Gaussian*). Filtrování šumu probíhá pomocí gaussovského filtru s normálním rozdělením. Velikost masky pro filtr se shoduje s velikostí masky pro detekci rohů. Pro detekci oblejších rohů je potřeba větší detekční maska, ostré rohy jsou lépe detekovány maskou menší. Proto je potřeba velikost masky operativně měnit.

SIFT tento problém řeší pomocí změny parametru rozptylu σ gaussovského filtru. Větším hodnotám rozptylu přísluší větší maska a naopak. Jelikož je vyhodnocení pomocí LoG výpočetně náročné, pracuje SIFT s jeho aproximací pomocí rozdílu gaussiánů (gaussovských filtrů) pro dvě různé hodnoty σ. Tento proces je opakován pro několik oktáv v Gaussově pyramidě. V každé oktávě je vyhodnoceno 5 filtrů pro různé hodnoty σ, ze kterých jsou následně získány 4 obrazy rozdílů těchto filtrů.

Výsledné obrazy jsou následně prohledávány. Program najde polohu všech lokálních maxim, které mají větší hodnotu, než je nastavená mez. Nalezené body jsou podrobněji vyhodnocovány a analyzovány pomocí LoG v měřítku, ve kterém leželo maximum rozdílu gaussovských filtrů pro tento bod. Pro každý bod je sestaven deskriptor, který popisuje směry a velikosti gradientů intenzity v blízkém okolí tohoto bodu. Blízké okolí takovéhoto bodu se nazývá dobrý charakteristický rys obrazu (*Good Feature*). Deskriptory jsou



Obrázek 3.9: Princip detekce charakteristických rysů pomocí SIFT [13]

uloženy a mohou složit pro porovnání s deskriptory získanými na jiném snímku. Pokud je zkoumaný objekt, pro který bylo nalezeno dostatečné množství charakteristických rysů, hledán na novém snímku, je nový snímek vyhodnocen stejným způsobem. Pokud se deskriptory nového snímku shodují s deskriptory hledaného objektu, je poloha objektu na novém snímku získána pomocí připasování odpovídajících si deskriptorů na sebe.

3.2.2 SURF [16]

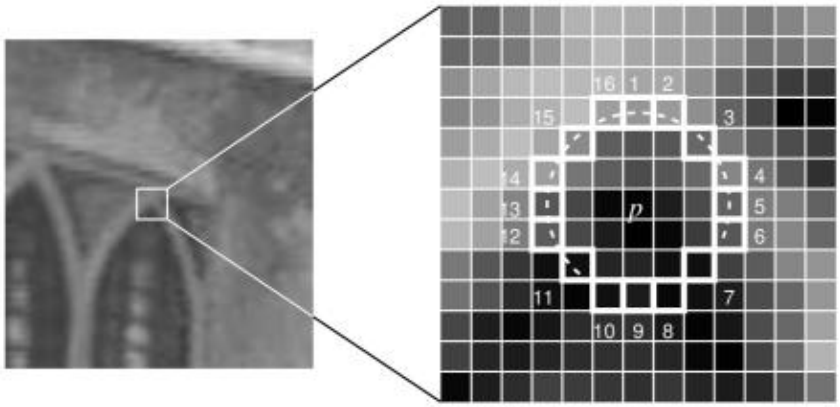
Algoritmus Sift je sice velice robustní a spolehlivá metoda detekce charakteristických rysů objektu, ale kvůli své náročnosti není vhodná pro aplikace s detekcí v reálném čase. Proto v roce 2006 pánové H. Bay, T. Tuytelaars a L. Van Gool publikovali článek *SURF: Speeded Up Robust Features*, ve kterém představili nový algoritmus nazvaný SURF – zrychlené robustní charakteristické rysy.

Zrychlení je dosaženo pomocí aproximace LoG. Zatímco SIFT používá pro aproximaci rozdíl gaussovských filtrů, SURF jde ještě dále a aproximuje i rozdíl pomocí jediného filtru. Díky tomu lze pro výpočet použít integrální obraz (sumační tabulku) podle článku [17]. Komplexnost takového výpočtu je *O(1)*, tedy pro výpočet hodnoty jednoho filtru jsou zapotřebí 4 aditivní operace, bez ohledu na velikost filtru. Rychlost celého procesu se tak zrychlila o řád.

Pro vyhodnocení nalezených bodů používá metoda velice podobné postupy jako SIFT. Výsledný deskriptor jednoho charakteristického rysu má díky větší míře aproximace ve vyhodnocovacím procesu poloviční velikost oproti deskriptoru SIFT.

3.2.3 FAST [18]

Ještě rychlejší metodou vhodnou pro real-time aplikace i na méně výkonných zařízeních je algoritmus *Features from Accelerated Segment Test* (FAST). Algoritmus je nutné před použitím správně nastavit a poskytnout mu dobrá data, neboť součástí inicializačního procesu je strojové učení ze snímků pořízených v prostředí, kde bude detekce v budoucnu probíhat.



Obrázek 3.10: Detekce charakteristických rysů pomocí FAST [19]

Postup FAST metody lze shrnout do několika kroků:

1. Algoritmus vyhodnotí intenzitu I_p v právě zkoumaném bodě p.
2. Je určena optimální prahová hodnota t.
3. Ve zkoumaném obrázku je vytvořena pomyslná kružnice s 16 pixely na obvodu, tak jak je to znázorněno v pravé části obrázku 3.10. Bod p je uznán jako dobrý kandidát pro rohový pixel, pokud lze najít nepřerušenou řadu n pixelů po obvodu kružnice, pro které platí, že jejich intenzita je větší než $I_p + t$ nebo menší než $I_p - t$, tedy že jsou rozeznatelně světlejší nebo tmavší než vyšetřovaný bod p. Optimální hodnota n je podle zdroje [18] 12.

Pro rychlejší vyhodnocení bodu 3 byl navržen vysokorychlostní test, po kterém má algoritmus jméno. Nejprve jsou otestovány pixely na pozicích 1 a 9. Pokud ani jeden z nich nemá větší nebo menší intenzitu než je $I_p + t$, resp. $I_p - t$, je bod p okamžitě odmítnut. V opačném případě jsou dále vyšetřeny pixely na pozicích 5 a 13. Pokud je p roh, musí 3 ze zatím zkoumaných 4 pixelů splňovat podmínku intenzity. Pokud bod p testem projde, jsou posléze vyhodnoceny hodnoty intenzity i ve zbývajících pixelech kružnice.

Body, které prošly testem a jsou prohlášeny za rohy, jsou nadále tříděny, aby se odstranily nadbytečné nálezy označující už detekovaný roh. Toho je docíleno prostou podmínkou o minimální povolené vzdálenosti mezi dvěma nalezenými rohy.

Hodnoty t, n a další vnitřní parametry metody jsou určeny pomocí strojového učení na snímcích, které musí algoritmu poskytnout sám uživatel.

Metoda je řádově rychlejší než ostatní algoritmy pro detekci charakteristických rysů, je ovšem daleko méně robustní a chybovější. Detekční algoritmus je velice citlivý na šum a světelné podmínky. Kvalita různých výsledků se může lišit podle poskytnutých tréninkových dat. Po skončení procesu strojového učení je metoda nenáročná nejen na výpočetní výkon, ale i na potřebnou paměť. Je proto vhodná především pro systémy s omezenými výpočetními schopnostmi, jako mohou být vestavěné systémy nebo roboty.

3.2.4 BRIEF [20]

Metoda SIFT používá k popisu charakteristického rysu deskriptor reprezentovaný vektorem o 128 prvcích. Jelikož jednotlivé prvky jsou čísla s desetinnou čárkou, celý vektor zaujímá prostor 512 bytů. Jak již bylo poznamenáno dříve, metoda SURF používá vektor poloviční délky, tedy 256 bytů. Pro popis složitých objektů metody obvykle vytvoří až tisíce deskriptorů. Výsledný objem dat způsobuje, že metody nejsou vhodné pro aplikace pracující s omezenými zdroji.

Jisté úspory paměti může být dosaženo hashováním vektoru deskriptoru nebo aplikací jiné kompresní metody (PCA,LDA). Tím jsou z vektorů získány řetězce binárních číslic, čímž se nejen ušetří paměť, ale navíc lze vyhodnocovat podobnost dvou takovýchto řetězců pomocí Hammingovy vzdálenosti [21], kterou lze pro binární řetězce počítat velice efektivně.

Nevýhodou tohoto přístupu je nutnost nejprve nalézt vektor deskriptoru a teprve potom z něj vypočítat komprimovaný binární řetězec. Algoritmus *Binary Robust Independent Elementary Features* (BRIEF) umožňuje mezikrok sestavení vektoru přeskočit a sestavit binární řetězec přímo z obrázku.

BRIEF neumí sám o sobě charakteristické rysy najít a je nutné ho implementovat spolu s metodou, která polohu rysů určí. Po vyhodnocení poloh středů těchto rysů převezme kontrolu BRIEF, který sestaví binární řetězce. Pomocí sofistikované metody (viz [20]) jsou v okolí bodů sestaveny dvojice pixelů. Pokud má první z této dvojice větší intenzitu než druhý, je do binárního řetězce přidána hodnota 1, v opačném případě 0.

Délka binárního řetězce může být 128, 256 nebo 512 bitů, tedy 16, 32 nebo 64 bytů. Díky menší velikosti reprezentace deskriptoru je zrychlen i celý proces detekce objektů.

3.2.5 ORB [22]

Důvodem vzniku tohoto algoritmu nebyla honba za dalším zlepšením existujících řešení. *Oriented FAST and Rotated BRIEF* (ORB) vznikl jako volně šiřitelná alternativa ke svým předchůdcům (SIFT, SURF...), které jsou patentově chráněné a za jejich užívaní se platí.

Tyto algoritmy jsou sice dostupné v neoficiální verzi OpenCV, ale z pochopitelných důvodů je nelze používat bez omezení na jiné než soukromé účely.

ORB používá modifikaci FAST algoritmu a BRIEF pro výpočet deskriptoru. FAST není schopen určit orientaci charakteristického rysu. Modifikace algoritmu spočívá v přidání další veličiny do deskriptoru, která udává vektor od středu charakteristického rysu do centroidu intenzity dané oblasti. Poloha centroidu je určena jako vážený průměr souřadnic pixelů v oblasti charakteristického rysu, kde jsou intenzity pixelu použity jako váhy.

Svou výkonností ORB předčí své zpoplatněné souputníky. Kvalita detekce je uspokojivá a příliš nezaostává za ostatními metodami.

3.2.6 Určení shody charakteristických rysů

Nejrozšířenější přístupy pro vyhodnocení podobnosti dvou objektů pomocí charakteristických rysů jsou:

- **Porovnávání hrubou sílou**

  Algoritmus označovaný jako *Brute-force Matcher* (BF) postupně porovnává jednotlivé charakteristické rysy hledaného objektu se všemi nalezenými rysy ve zkoumaném obraze. Pokud nebyla nalezena žádná shoda, metoda daný rys nepřiřadí. Pokud bylo nalezeno shod více, metoda obvykle označí deskriptor s nejvyšší hodnotou vzájemné podobnosti. Funkci, která vyhodnocuje a boduje míru podobnosti dvou charakteristických rysů, je potřeba volit podle povahy deskriptoru. Může se například jednat o euklidovskou nebo Hammingovu vzdálenost. [23]

- **FLANN**

  Fast Library for Approximate Nearest Neighbors (FLANN) obsahuje celý soubor algoritmů inteligentně hledající shody deskriptorů. Ze začátku algoritmus funguje s podobnou rychlostí jako BF metody, ale poté, co je nalezeno několik počátečních shod, hledá algoritmus systematičtěji a porovnává nejprve blízké sousedy již spárovaných deskriptorů. Díky tomu je podstatně rychlejší než hledání hrubou silou. Pro většinu aplikací je tento způsob zbytečný, neboť párování deskriptoru trvá jen zlomek doby celého detekčního procesu. S rostoucím množstvím deskriptorů se poměr doby porovnávání ku celkové době detekce zvětšuje, protože náročnost sestavení deskriptorů roste lineárně s počtem charakteristických rysů, zatímco náročnost BF párovacích metod je úměrná kvadrátu počtu rysů. FLANN je tedy výhodné použít pro rozsáhlé aplikace detekující mnoho objektů. [24]

3.3 Template matching

Jedna z nejjednodušších metod detekce známého objektu na neznámém obrazu je *Template Matching* (TM) neboli porovnávání se vzorem. Vzorem je v tomto případě obrázek hledaného objektu reprezentovaný pomocí matice. Vzor je přiložen na zkoumaný obraz. Pro oblast, kde se oba snímky překrývají, je spočítán koeficient vzájemné korelace vzoru a vyšetřovaného obrazu. Vzor se následně posouvá do nové polohy, kde se celý proces opakuje. Výsledkem tohoto kroku je matice korelačních koeficientů M. Pokud má vzor rozměry (w, h) a vyšetřovaný obraz (W, H), potom má matice M rozměry $(W - w + 1, H - h + 1)$. [23]

Dalším krokem je hledání maxima v korelační matici. Pokud je hodnota maxima menší než nastavená mezní hodnota, algoritmus skončí a žádný nález nedetekuje. V opačném případě označuje poloha maxima i polohu hledaného objektu. [23]

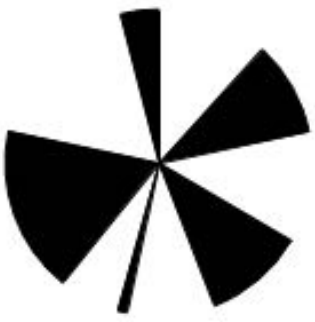
Slabinou TM metody je nutnost znát předem orientaci a velikost hledaného objektu. V opačném případě je nutné hledání iterativně opakovat pro všechny myslitelné velikosti a natočení vzoru na zkoumaném obrázku, čímž se náročnost metody drasticky zvýší a prohledávání jednoho snímku může trvat i několik minut.

Tuto nevýhodu lze částečně kompenzovat tím, že hledaný objekt bude invariantní při rotaci nebo změně měřítka. Tím se odstraní potřeba prohledávání přes danou veličinu iterovat. Pro zbývající veličinu je možné zadat vhodné krajní podmínky, aby se celý proces opakoval co nejméně.

Další možností je převést obraz do logaritmicko-polárních souřadnic, ze kterých lze zjistit natočení a zvětšení objektu. Patřičně natočený a zvětšený objekt je potom možné hledat v původním obraze v kartézských souřadnicích pro nalezení souřadnic objektu. Tuto modifikaci ovšem nelze provést při hledání více objektů podle stejného vzoru.

Příkladem rotačně invariantního vzoru mohou být soustředné kružnice z obrázku 3.11a. Obrázek 3.11b zachycuje možnou podobu tagu invariantního při zvětšování.

(a) Rotačně invariantní tag.



(b) Měřítkově invariantní tag.

Obrázek 3.11: Přehled tagů vhodných pro TM metodu detekce

Nespornou výhodou TM metody je její odolnost proti okluzi. Pokud je hledaný vzor dostatečně unikátní vzhledem ke zbytku zkoumaného snímku, lze ho úspěšně identifikovat i při zakrytí 50 % plochy hledaného předmětu.

3.4 Substrakce pozadí

Popředí obrazu je soubor všech objektů, které jsou zajímavé pro další analýzu v kontextu dané aplikace. Jsou to objekty, které má systém za úkol detekovat, identifikovat, sledovat nebo s nimi provádět jiné vhodné operace. Pozadí obrazu obsahuje vše ostatní, tedy pro systém nezajímavé oblasti snímku, které ke správnému běhu systému nejsou potřeba. Pokud se před dalším zpracováním obrazu podaří pozadí obrazu odfiltrovat, sníží se obtížnost dalších úkolů prováděných s objekty v popředí, neboť algoritmy pracující s obrazem budou mít méně dat, které je nutné zpracovat.

Pokud je podoba pozadí známá (je k dispozici snímek pozadí), je možné získat masku popředí prostým odečtením zkoumaného snímku od snímku pozadí. V místech, kde je rozdíl těchto obrazů nenulový, leží objekty v popředí. Jednoduchost tohoto přístupu dává prostor spoustě nepříznivých vlivů, jako mohou být rozdílné světelné podmínky, stíny objektů v popředí, šum nebo rozdílný úhel kamery pro snímek pozadí a vyšetřovaný snímek. Proto je vhodné snímek rozdílů pozadí a vyšetřovaného obrazu upravit například filtrací, prahováním a morfologickými transformacemi pro odstranění šumu (například pomocí eroze následované dilatací).

Vzniklý binární obraz má hodnotu 1 v místech, kde se vyskytuje objekt v popředí, a hodnotu 0 pro pozadí, lze ho tedy použít jako masku původního vyšetřovaného snímku pro další zpracování.

Jen v málo případech je podoba pozadí předem známá a ani nelze zaručit, že se nebude měnit za běhu celé aplikace. Podobu pozadí lze poté určit přibližně z několika snímků pořízených s vhodnými časovými odstupy. Tato metoda funguje správně jen za předpokladu, že jsou všechny objekty v popředí pohyblivé do té míry, aby bylo pozadí za nimi na některých snímcích vidět. Podobu pozadí lze vyhodnotit nejjednodušeji jako průměr nebo medián intenzit jednotlivých pixelů přes všechny pořízené snímky.

3.4.1 MOG [25]

MOG je algoritmus určený k segmentaci pozadí a popředí. Metoda modeluje pravděpodobnost výskytu různých barev určitého pixelu pro všechny pixely obrazu pomocí součtu K gaussovských rozložení ($K \in < 3; 5 >$). K vyhodnocení pravděpodobnostních modelů dochází online, tedy za chodu aplikace. Pravděpodobná podoba pozadí je určena pomocí nejpravděpodobnějších barev daného pixelu.

3.4.2 MOG2 [26]

MOG2 se od svého předchůdce liší jen jednou podstatnou věcí. Zatímco MOG k modelování pravděpodobnosti používá pevně stanovený počet normálních rozdělení, MOG2 tento počet stanoví sám pro každý pixel zvlášť. Díky tomu je algoritmus adaptabilnější a spolehlivější.

3.4.3 GMG [27]

GMG metoda přizpůsobuje pravděpodobnostní modely podoby pozadí během chodu programu. Model je na základě nových dat upraven pomocí Bayesovy věty. Metoda dává nově naměřeným datům větší váhu než datům starším. Po základní analýze a sestavení mapy pozadí je na mapu aplikována řada morfologických filtrů na odstranění drobných chyb a šumu. Metoda je robustnější než MOG nebo MOG2, ale náročnější na výpočetní výkon.

3.5 Detekce pomocí barev [28]

Objekt je tím lépe detekovatelný, čím jsou jeho vizuální vlastnosti unikátnější oproti zbytku vyšetřovaného snímku. Hledaný objekt může být charakteristický například tvarem nebo barvou. Zatímco detekce tvaru je už matematicky i výpočetně relativně složitější záležitost, detekce konkrétní barvy je velice primitivní. Každý pixel vyšetřovaného obrazu je porovnán s hledanou barvou, metoda pixel označí 1, pokud jsou si barvy podobné, v opačném případě je pixel označen hodnotou 0.

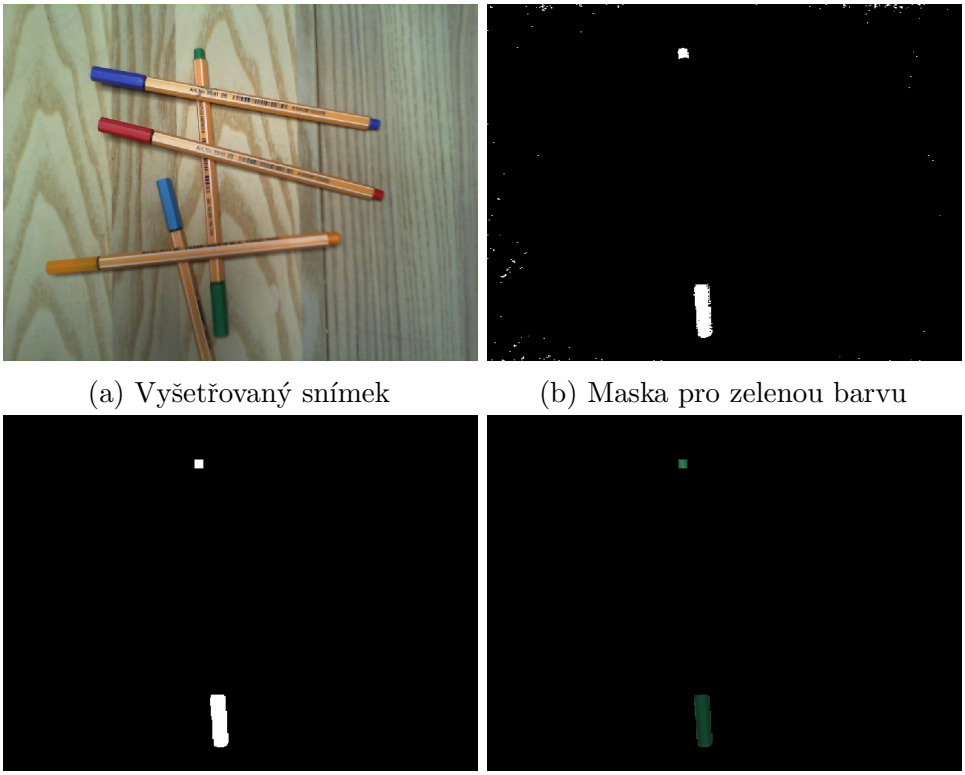
Tímto postupem vznikne mapa míst s hledanou barvou, která může být po dalších úpravách použita jako maska pro původní snímek. Mezi časté úpravy patří filtrování na odstranění šumu a morfologické transformace na odstranění nezajímavě malých oblastí nebo zacelení děr v oblastech velkých.

Jelikož je barva pixelu reprezentována běžně 3 hodnotami RGB, vyvstává otázka, jakým způsobem bude vyhodnocena podobnost dvou barev. Prostým vymezením intervalu pro jednotlivé hodnoty RGB vznikne v RGB prostoru kvádr, který ovšem není ideálním tvarem pro popis určité barvy v různých intenzitách. Pokud by měl kvádr relativně malé rozměry, metoda nebude schopna detekovat stejnou barvu v různých kontrastech a v různých světelných podmínkách. Při zvětšování kvádru dojde k detekci nežádoucích odstínů dalších barev. Lepší barevný popis lze dostat vymezením prostoru v RGB pomocí složitějších funkcí zohledňujících všechny 3 složky RGB současně. Další možností je transformace barevného vyjádření z RGB do jiného barevného prostoru, který je pro popis konkrétních odstínů vhodnější.

Barevný prostor HSV používá 3 složky: odstín, sytost a jas (Hue, Saturation, Value).

Odstín hledané barvy je tedy reprezentován jen jednou souřadnicí, pro detekci lze použít interval povolených hodnot pro hodnoty odstínů. Sytost a jas barvy hledaného předmětu závisí převážně na světelných podmínkách, jejich povolené hodnoty lze vymezit širším intervalem. K popisu hledané barvy v HSV je kvádr vhodným tělesem.

Obrázek 3.12 demonstruje nalezení zelených objektů pomocí detekce barev v HSV prostoru.



(a) Vyšetřovaný snímek (b) Maska pro zelenou barvu

(c) Maska po odstranění šumu (d) Extrahovaný zelený objekt

Obrázek 3.12: Detekce objektu pomocí filtrace barev

3.6 Konvoluční neuronové sítě

Tato kapitola byla zpracována zejména ze studijních podkladů [29] a článku [30].

Konvoluční neuronové sítě (CNN) (*Convolution Neural Network*) se na první pohled liší od klasických neuronových sítí formátem vstupních dat. Zatímco klasická neuronová síť požaduje vstupní data ve formě jednorozměrného vektoru, CNN využívá jako vstup matici. Je tedy vhodná pro zpracování obrazů.

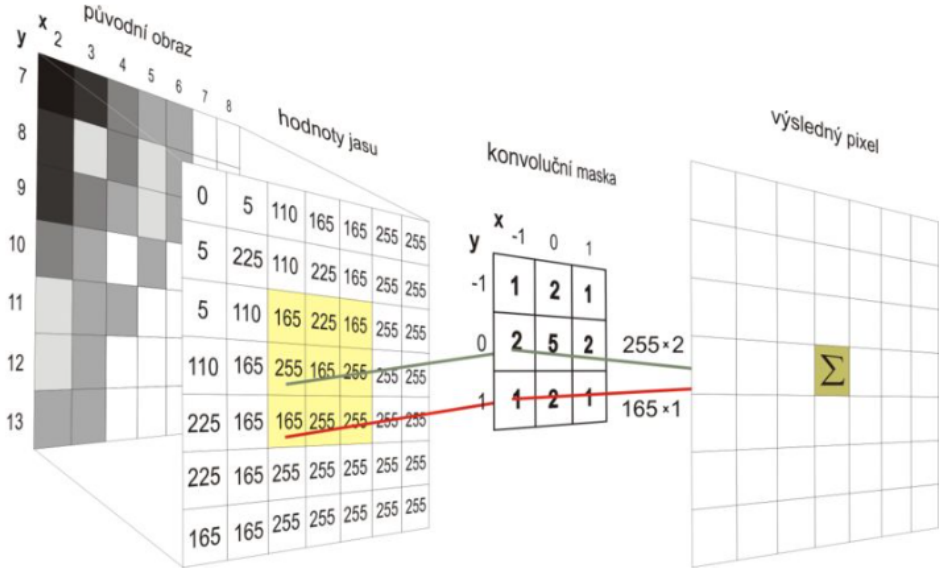
Před samotnými neurony obsahuje CNN řadu konvolučních vrstev, které mají za cíl identifikovat významné rysy hledaných objektů, a to mimo jiné hrany a rohy. Parametry konvolučních filtrů jsou stejně jako váhové vektory neuronů předmětem učícího procesu celé sítě pomocí principů back-propagation.

Konvoluce ve dvojrozměrném prostoru je matematicky definovaná podle rovnice

$$y(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h(k_1, k_2) x(n_1 - k_1, n_2 - k_2) \quad ,$$

kde y představuje výstupní obraz konvoluce provedené na snímku x pomocí konvolučního filtru h. [31]

Konvoluci dále vysvětluje ilustrační příklad na obrázku 3.13.



Obrázek 3.13: Ukázka konvoluce obrazu [31]

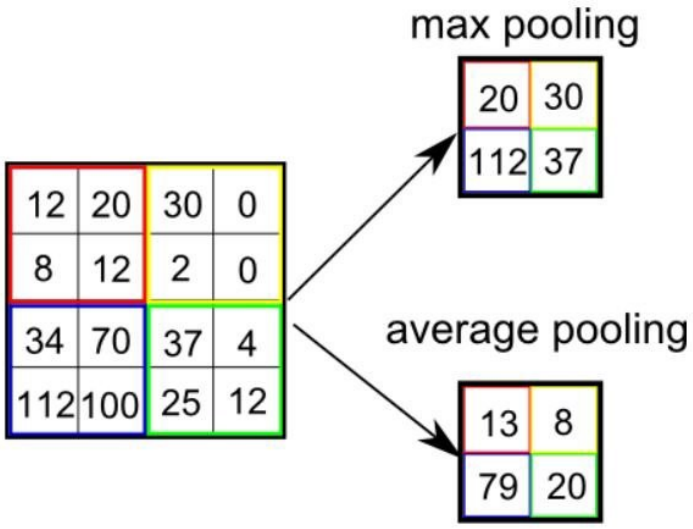
Výsledný obraz konvoluce nejčastěji obsahuje vysoké hodnoty intenzit pixelů v místech, kde se vyskytuje rys, který konvoluční filtr hledá (například hrana). Nezajímavá místa mají hodnoty intenzit nižší.

Každý snímek prochází sérií různých konvolučních filtrů a je generována řada nových snímků. Jelikož se objem dat tímto procesem znásobil, je následně řazena vrstva typu pooling, která redukuje velikost výstupních obrazů z konvoluční vrstvy.

Poolingová vrstva opět obsahuje masku, která se posouvá po obraze, podobně jako se posouvá filtr při konvoluci. Velikost posuvu je ovšem větší než 1 pixel a odpovídá velikosti poolingové masky. Jelikož je při každém kroku vygenerována hodnota jednoho nového pixelu, zmenší se velikost výsledného obrazu z (N, M) na $(N/K_1, M/K_2)$, kde (K_1, K_2) jsou rozměry poolingové masky.

Nejpoužívanější poolingovou vrstvou jsou metody max pooling a average pooling, které vyhodnocují hodnotu podle maxima, resp. průměru hodnot pod poolingovou maskou. Ilustrační příklad poolingu je zachycen na obrázku 3.14.

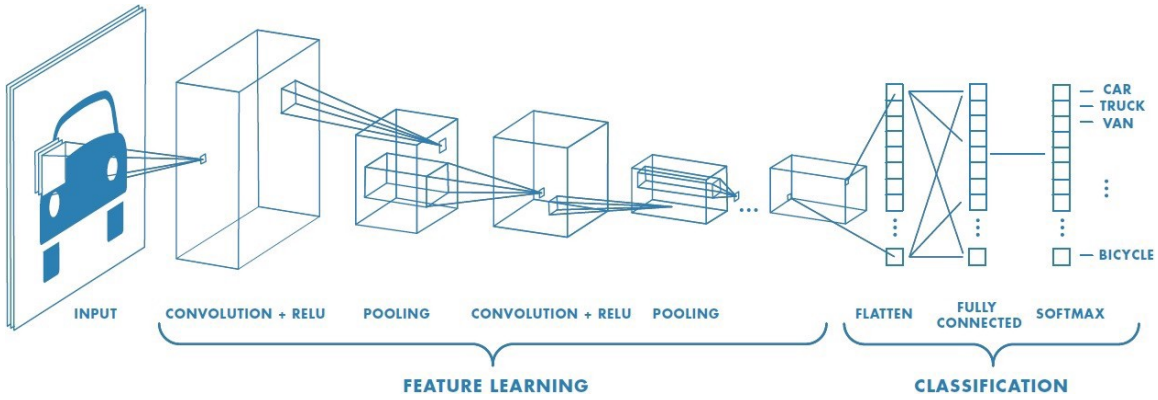
CNN střídají sérii konvolučních a poolingových vrstev. Jelikož každá konvoluční vrstva obsahuje několik konvolučních filtrů, počet obrazů jdoucích sítí se s každou vrstvou násobí.



Obrázek 3.14: Ukázka poolingové vrstvy [32]

Za konvolučními a poolingovými vrstvami následuje vrstva typu flatten, která uspořádá data (v tuto chvíli 3D) do jednorozměrného vektoru. Tento vektor je použit jako vstup do sítě neuronů. Poslední vrstvy sítě jsou tvořeny neurony, které udávají míru náležitosti zkoumaného obrazu určitému objektu. Pro každý hledaný objekt obsahuje poslední vrstva právě jeden neuron.

Kompletní schéma CNN je znázorněno na obrázku 3.15.



Obrázek 3.15: Ukázka konvoluční neuronové sítě [32]

Každý neuron sítě obsahuje váhový vektor $\mathbf{w}$. Délka váhového vektoru je shodná s počtem vstupů daného neuronu. Výstup neuronu je získán jako skalární součin vektoru vah $\mathbf{w}$ a vstupního vektoru $\mathbf{x}$:

$$y = \mathbf{w}^T \cdot \mathbf{x} \tag{3.1}$$

Aby neuronová síť dokázala modelovat i nelineární systémy, je běžné výstup z neuronu modifikovat další funkcí. Jednou z nejpoužívanější funkcí je Relu *(Rectified linear unit)*.

Po použití této funkce lze rovnici neuronu opravit na

$$y = max(0, \mathbf{w}^T \cdot \mathbf{x}) \quad . \tag{3.2}$$

Učení CNN je možné provádět tzv. s učitelem nebo bez učitele. V prvním případě je nutné tréninková data manuálně označit a tím dát síti informaci, o jaký objekt se jedná. Ve druhém případě probíhá učení na základě třídění podobných objektů do tzv. klastrů (skupin). Neuronová síť je schopna objekt na novém obrázku přiřadit do jedné z již existujících skupin objektů.

Běžnou praxí je, že zkoumaný snímek obsahuje více než jeden hledaný objekt. V takovém případě je nutné nejprve identifikovat oblasti, kde se hledaný objekt může vyskytovat. Přístupů k tomuto problému je celá řada.

- **Faster R-CNN [33]**
  Tento algoritmus je nástupcem metody Fast R-CNN, která ke svému fungování využívala další metodu zvanou Selective Search. Selective Search je algoritmus pro generování zájmových oblastí, jenž z každého snímku generuje stovky oblastí, které jsou dále jednotlivě analyzovány pomocí CNN. Z tohoto důvodu byl Fast R-CNN nevhodný pro aplikace v reálném čase. Faster R-CNN používá na generování zájmových oblastí neuronovou síť *Region Proposal Network* (RPN), která je na obraz aplikována před samotnou CNN. Výstupem RPN jsou souřadnice obdélníků opisujících pravděpodobný objekt. I přes svou náročnost na výpočetní výkon je algoritmus schopen detekovat objekty v reálném čase.

- **YOLO [34]**
  Síť mající název z angl. *You Only Look Once* je schopná detekovat více objektů při jediném průchodu neuronovou sítí. Obraz je rozdělen do mřížky. Každé pole mřížky slouží jako výchozí bod pro generování zájmových oblastí. Zájmové oblasti prochází CNN, jejímž výstupem je pravděpodobnost, že objekt v právě zkoumané oblasti je pozadí, míra příslušnosti k jednotlivým hledaným objektům a údaje o změně velikosti a polohy zájmové oblasti tak, aby lépe přiléhala k detekovanému objektu.

- **SSD [35]**
  Jedná se taktéž o síť schopnou detekovat více objektů při jednom průchodu. Zájmové oblasti jsou generovány z příznakové mapy, která je získána zpracováním obrazu několika konvolučními vrstvami. Výstup ze sítě SSD je podobný jako výstup z YOLO. Hlavní odlišností je, že pravděpodobnost příslušnosti objektu k pozadí není oddělena od pravděpodobnosti příslušnosti k jiným objektům. Je tedy součástí stejného vektoru a s pozadím je nakládáno obdobně jako s hledaným objektem.

3.7 Detekce pomocí histogramů

Histogram obrazu je graf vyjadřující distribuci intenzity všech pixelů. Pro každou hodnotu pixelu udává procentuální zastoupení stejných pixelů ve vyšetřovaném snímku. Histogram lze zpracovávat pro jednokanálové černobílé snímky, pro každý kanál zvlášť u barevných snímků nebo jako vícerozměrný diagram pro vícekanálové obrazy. Pro účely detekce objektů se používají histogramy z jednokanálového černobílého snímku.

Pro úspěšnou detekci je nejprve nutné znát histogram hledaného objektu. Ten je možné získat analýzou části snímku, na kterém se objekt vyskytuje. Před samotným výpočtem je vhodné na obraz objektu aplikovat masku tak, aby vyhodnocení histogramu probíhalo jen na pixelech příslušících zkoumanému objektu.

Jelikož je histogram pro každý objekt unikátní, lze ho použít jako číselné vyjádření identifikace objektu. Metody detekující objekty pomocí histogramů vyhodnocují zkoumaný snímek a každé jeho části přiřazují pravděpodobnost příslušnosti k histogramu hledaného objektu. Všechny pravděpodobnosti příslušností jsou uloženy do dvojrozměrné mapy, jejíž oblasti s vysokou intenzitou vyjadřují pravděpodobný výskyt hledaného objektu.

Algoritmy používající tyto principy jsou MeanShift a CamShift, o nichž je pojednáno dále.

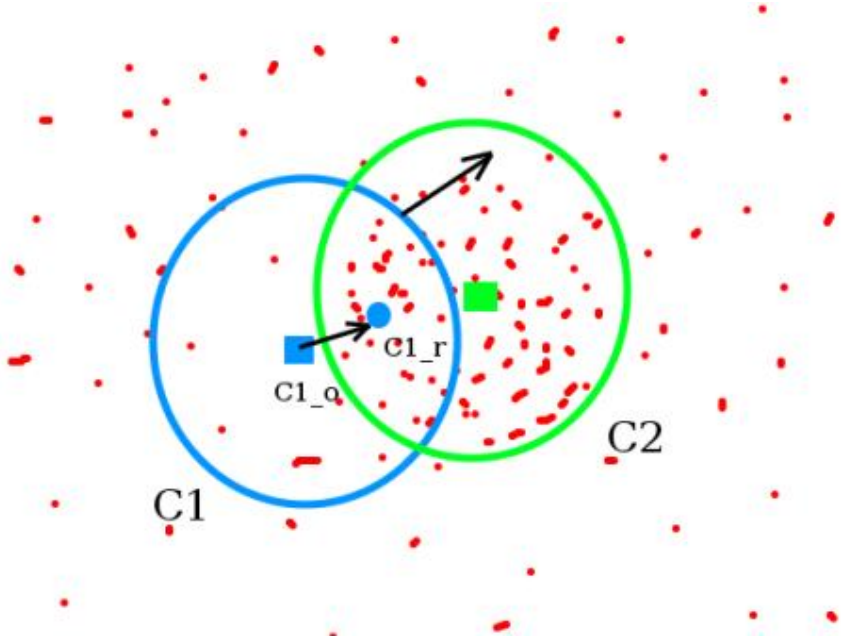
3.7.1 MeanShift [36]

Algoritmus MeanShift potřebuje kromě výše zmíněného ještě parametry velikosti hledaného objektu a počáteční polohu, ze které odstartuje prohledávání. Velikost hledaného objektu je popsána velikostí obdélníku objektu opsaného.

Metoda nejprve spočítá histogram hledaného objektu, poté sestaví mapu pravděpodobností příslušnosti k objektu každého pixelu ve zkoumaném obraze. Postup hledání oblasti s nejpravděpodobnějšími příslušnostmi je zachycen na obrázku 3.16.

Kolem počáteční polohy, která je označena jako $C1_o$, je ohraničena oblast o velikosti hledaného objektu $C1$. Tato oblast je použita jako maska mapy pravděpodobnosti příslušností. Je vyhodnoceno těžiště pravděpodobností jako vážený průměr souřadnic pixelů v oblasti, jako váhy jsou použity hodnoty pravděpodobnosti příslušnosti. Získané těžiště $C1_r$ je použito jako nová poloha pro další iterace algoritmu. Proces končí, když se nově získaný střed oblasti neliší od středu z předchozí iterace. Oblast $C2$ je výsledkem toho algoritmu.

Metoda tedy v každé iteraci posouvá vyšetřovanou oblast ve směru gradientu růstu pravděpodobnosti příslušnosti, výsledná oblast představuje pouze lokální maximum. Z tohoto důvodu je metoda vhodná pro sledování již jednou identifikovaných objektů na videu, nikoli pro detekci objektu bez přibližné znalosti jeho předchozí polohy.



Obrázek 3.16: Detekce charakteristických rysů pomocí FAST [37]

3.7.2 CamShift [38]

Problémem metody MeanShift je její neschopnost rozpoznat velikost objektu na snímku. Pokud se objekt na videu pohybuje ve směru ke kameře, jeví se větší. Jelikož je sledovaný objekt větší než maska, pomocí které je hledán, algoritmus neoznačí objekt celý, ale jen jeho část. Při výraznější disproporci velikostí může dokonce dojít k úplné ztrátě sledovaného objektu.

CamShift (*Continuously Adaptive MeanShift*) nejprve aplikuje klasický MeanShift algoritmus. Po nalezení nové polohy sledovaného objektu je vypočítána nová velikost sledovaného objektu, a dokonce i jeho orientace proložením mapy elipsou. Elipse je opsán obdélník, který slouží jako maska pro další iteraci MeanShift algoritmu. Proces postupně konverguje k elipse, která nejlépe opisuje sledovaný objekt. Výstupem algoritmu je natočený obdélník opisující poslední elipsu.

4 Výběr vhodných metod CV

Pro přehlednost byly možné metody vycházející z rešerše v kapitole 3 shrnuty v tabulkách. Každá tabulka obsahuje nejdůležitější výhody a nevýhody daných řešení. Problematika detekce objektů byla rozdělena na úlohu detekce poloh aktuátorů a detekce stavu indikačních LED.

4.1 Detekce poloh aktuátorů

Tabulka 4.1 porovnává možné přístupy k detekci aktuátorů.

Tabulka 4.1: Porovnání metod pro detekci aktuátorů

Metoda	Výhody	Nevýhody
Systém referenčních tagů Pohyblivá i statická část aktuátoru je označena sérií tagů. Tagy jsou nalezeny pomocí metod strojového vidění podle typu použitých tagů. Poloha aktuátoru je vyhodnocena podle vzájemného posunutí pohyblivé a statické části aktuátoru.	• Univerzální metoda pro různé designy aktuátorů. • Možnost rozlišit velké množství objektů. • Možnost výběru konkrétního systému pro optimalizaci výpočetního času a spolehlivosti.	• Náchylné na okluzi. • Nutnost fyzického zásahu do procesu a instalace. • Nutná dostatečná ostrost snímků.
Detekce charakteristických rysů Na hledaném objektu jsou identifikovány unikátní hrany a rohy, které slouží k identifikaci objektu na neznámém snímku. Poloha aktuátoru je vyhodnocena ze vzájemného posunutí charakteristických rysů na pohyblivé a statické části.	• Jednoduchá implementace. • Zásah do zkoumaného procesu není nutný. • Schopný identifikovat velké množství rozličných aktuátorů.	• Některé aktuátory mohou obsahovat malý počet charakteristických rysů. Takové je nutno označit dalšími značkami. • Problém s identifikací stejných aktuátorů mezi sebou, zvláště při posunu kamery. • Volně dostupný je pouze algoritmus ORB.

Tabulka 4.1: Porovnání metod pro detekci aktuátoru

Metoda	Výhody	Nevýhody
Template matching aktuátoru Pořízené fotografie aktuátorů ve všech významných polohách jsou hledány na zkoumaném snímku pomocí TM metody. Poloha aktuátoru je vyhodnocena podle významnosti nálezů pro různé hledané polohy.	• Jednoduchá implementace. • Zásah do zkoumaného procesu není nutný. • Schopný identifikovat velké množství rozličných aktuátorů. • Odolný proti částečné okluzi.	• Náročné na výpočetní výkon, nutno hledat všechny myšlené orientace a velikosti aktuátoru. • Problém s identifikací stejných aktuátorů mezi sebou, zvláště při posunu kamery. • Problémová identifikace tvarově neunikátních aktuátorů, špatně odlišitelných od pozadí.
Template matching tagu Pohyblivá i statická část aktuátoru je označena sérií tagů vhodných pro TM. Tagy jsou nalezeny pomocí TM metody. Poloha aktuátoru je vyhodnocena podle vzájemného posunutí pohyblivé a statické části aktuátoru.	• Univerzální metoda pro různé designy aktuátorů. • Odolnost vůči částečné okluzi nebo rozostření snímků.	• Náročné na výpočetní výkon, nutno hledat všechny myšlené orientace nebo velikosti tagu. • Nutnost fyzického zásahu do procesu a instalace tagů.
Konvoluční neuronové sítě O detekci aktuátorů v různých polohách se stará CNN. Výstupem sítě je pravděpodobnost náležitosti objektu k jednotlivým aktuátorům v různých polohách.	• Univerzální a robustní metoda. • Zásah do zkoumaného procesu není nutný. • Při správném nastavení velká odolnost vůči okluzi, rozmazání snímku nebo posunutí kamery.	• Složité nastavení na straně uživatele. • Problém s identifikací stejných aktuátorů mezi sebou. Nutné řešit další metodou.
Substrakce pozadí před použitím dalších metod Nejprve je snímek filtrován a jsou hledány pouze objekty v popředí. Aktuátory jsou na předzpracovaném obraze detekovány některou z dalších zmíněných metod.	• Menší složitost vyšetřovaného snímku. • Zásah do zkoumaného procesu není nutný.	• Budou detekovány jen části v pohybu nebo části, jež byly v pohybu bezprostředně před detekcí. • Navýšení výpočetního času.
Detekce pomocí histogramů Hledaný objekt je nejprve vyfocen a je určen jeho histogram. Na dalších snímcích je objekt hledán pomocí MeanShift nebo CamShift algoritmu.	• Jednoduchá implementace. • Zásah do zkoumaného procesu není nutný.	• Problémy s detekcí rychle se pohybujících objektů. • Problém s identifikací stejných aktuátorů mezi sebou. • Citlivé na světelné podmínky a pozadí.

Monitorované procesy mohou mít značně rozdílný charakter. Některé procesy mohou klást zvýšené nároky na spolehlivost a robustnost metody detekce. Systém se při použití v těžkém průmyslu může potýkat s prašným, špatně osvětleným nebo zakouřeným prostředím. Naopak pro procesy v čistém prostředí bude pravděpodobně klíčová rychlost a snímkovací frekvence kamer. Není proto jednoduché určit optimální metodu CV globálně. Z těchto důvodů byl zvolen modulární přístup k řešení problému. Systém bude obsahovat více metod CV, konkrétní metodu vybere uživatel podle povahy monitorovaného procesu.

Na základě výše uvedené tabulky jsou jako první implementovány metody detekcí pomocí systému referenčních tagů a template matching tagu.

Systém referenčních tagů je nenáročná metoda jednoduchá na implementaci a relativně snadně implementovatelná poučeným pracovníkem do zkoumaného procesu. Její největší výhodou je přesnost a rychlost detekce, díky tomu je vhodná do rychlých procesů v čistém prostředí nebo v prostředích, kde lze zaručit, že nedojde k okluzi tagů nebo jejich zašpinění.

Detekce tagů pomocí metody template matching je oproti tomu řádově pomalejší. V případě vhodně zvolených tagů a správného nastavení si ovšem poradí i se zkresleným tvarem detekovaného tagu, a to i při velké okluzi. Tato metoda je vhodným kandidátem do pomalejších procesů v průmyslovém prostředí.

Substrakce pozadí není v CAFDIS implementována. Systém dosahuje dobrých výsledků i bez tohoto předzpracování, které by vedlo ke ztrátě informací o nepohyblivých, pomalu se pohybujících nebo zřídka pohyblivých objektech.

Díky tomu, že jsou obě zvolené metody založeny na detekci tagů, práce s daty bude po detekci stejná. Práce se systémem bude tudíž pro uživatele u obou metod jednotná, a tedy snazší. Z pohledu uživatele je klíčovou operací správné označení zkoumaného procesu tagy.

4.2 Detekce stavu indikačních LED

Tabulka 4.2 porovnává možné přístupy stavu LED.

Tabulka 4.2: Porovnání metod pro detekci LED

Metoda	Výhody	Nevýhody
Substrakce pozadí a detekce barev Nejprve je snímek filtrován a jsou hledány pouze objekty, které se v čase mění. Je tedy možné detekovat LED jen tehdy, pokud zrovna přechází mezi stavy. Stav LED je určen pomocí detekce barev.	• Nevyžaduje zásah do zkoumaného procesu. • Nenáročné na výpočetní výkon. • Měnící se LED jsou systémem samy nalezeny, není nutná jejich inicializace.	• Detekuje stav pouze v okamžiku změny. • Obtížná detekce při posunutí kamery v okamžiku změny stavu LED. • Problém s rozpoznáním více LED mezi sebou. Nutné řešit další metodou. • Detekovaná barva značně závisí na světelných podmínkách.
Systém referenčních tagů a detekce barev Okolí LED je označeno sérií tagů. Tagy jsou nalezeny pomocí metod strojového vidění podle typu použitých tagů. Přesná poloha LED je vyhodnocena na základě souřadnic nalezených referenčních tagů. Stav LED je určen pomocí detekce barev.	• Univerzální metoda pro různé designy indikačních LED. • Možnost rozlišit velké množství různých LED.	• Pro umístění referenčních tagů v okolí indikační LED často nemusí být místo. • Nutnost fyzického zásahu do procesu a instalace tagů. • Detekovaná barva značně závisí na světelných podmínkách.
Konvoluční neuronové sítě O detekci stavu indikačních LED se stará CNN. Síť identifikuje LED a vyhodnotí míru pravděpodobností jednotlivých stavů.	• Zásah do zkoumaného procesu není nutný. • Při správném nastavení velká odolnost vůči posunutí kamery a změně světelných podmínek.	• Velice složité nastavení a učení sítě na straně uživatele. • Problém s identifikací stejných LED mezi sebou.

Na základě výše uvedené tabulky byla do systému CAFDIS implementována detekce stavu indikačních LED pomocí detekce referenčních tagů a detekce barev. Tento způsob umožní spolehlivou lokalizaci polohy LED díky sadě referenčních tagů, kterými bude monitorovaný prostor označen. Pro účely detekce polohy LED lze použít podobné, nebo i stejné tagy jako pro detekci poloh aktuátorů. Pro uživatele je díky podobnosti metod obsluha systému jednodušší.

Díky univerzálnosti této metody je možné použít stejných principů pro detekci dalších informací o procesu. Na základě vyhodnocení barvy v určité oblasti lze kontrolovat

přítomnost výrobku na určitém místě, kontrolovat světelné podmínky podle barvy referenční plochy, odhalit přítomnost cizího tělesa v kontrolované oblasti nebo i částečně kontrolovat polohu ručiček analogových přístrojů. Vše samozřejmě za předpokladu, že v průběhu kontrolované události dojde ke změně barvy dané oblasti.

Z tohoto důvodu bude tato metoda, primárně určena pro detekci stavu indikačních LED, nadále zmiňována jako detekce barvy oblasti nebo jen detekce oblasti.

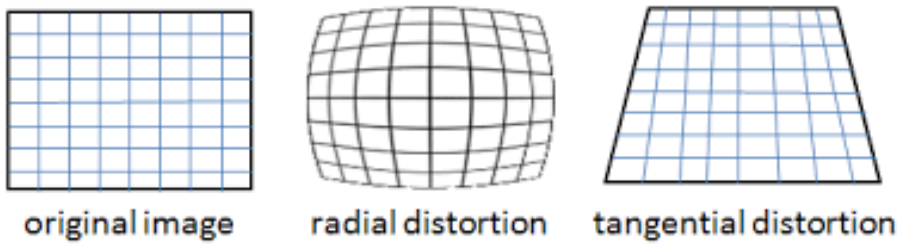
5 Kalibrace kamery [39]

Obraz z digitálních kamer je zatížen řadou chyb a zkreslení. Chyby obrazu mohou ovlivnit ostrost snímku, barevné podání objektů a tvary snímaných objektů. Jelikož je systém CAFDIS založen především na detekování souřadnic referenčních tagů, jsou to právě chyby tvaru, které budou kvalitu měření ovlivňovat nejvíce.

Chyby barevného podání nejsou zásadní ani pro detekci barvy indikační LED. Detekovaná barva nebude sice odpovídat barvě reálné LED, ale to nemusí správnost detekce nijak ovlivnit. Výstupem detekování barvy LED bude totiž její stav, a nikoli přesná barva. Tento stav je možné vyhodnotit porovnáváním detekované barvy s barvami ve stavech zapnuto a vypnuto pořízených stejnou kamerou, tedy i zatíženou stejným barevným zkreslením.

Chyby tvaru jsou způsobeny systematickými výrobními nedostatky a vadami kamery. Mohou být způsobeny neideálním tvarem čoček, mimoběžností os čoček nebo nepřesně umístěným CCD čipem. Díky povaze původu těchto chyb je možné velkou část z nich digitálně korigovat nebo úplně odstranit. Nejzávažnější chyby tvaru jsou radiální a tangenciální zkreslení.

Radiální zkreslení je způsobeno rozdílným zvětšením obrazu v závislosti na vzdálenosti od optické osy objektivu. Ve většině případů zvětšení s rostoucí vzdáleností od osy pouze roste nebo pouze klesá. V takových případech nastává tzv. poduškovité, resp. soudkovité radiální zkreslení. Ukázku soudkovitého radiálního zkreslení je možné vidět na obrázku 5.1.



Obrázek 5.1: Radiální a tangenciální zkreslení [40]

Radiální zkreslení lze kompenzovat soustavou rovnic

$$x_{korigovana} = x(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \quad , \tag{5.1}$$

$$y_{korigovana} = y(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \quad , \tag{5.2}$$

kde

$$r^2 = y^2 + x^2 \quad . \tag{5.3}$$

Proměnné x, y představují původní nekorigované souřadnice pixelu v normalizovaných souřadnicích. Normalizované souřadnice mají svůj počátek přesunut do optické osy, tedy do středy obrázku, a jsou poděleny ohniskovou vzdáleností vyjádřenou v pixelech, jedná se tudíž o bezrozměrné veličiny. Korigované proměnné $x_{korigovana}, y_{korigovana}$ jsou taktéž vyjádřeny v normalizovaných souřadnicích a je nutné je přepočíst do původního systému.

Koeficienty k_1, k_2, k_3 jsou předmětem kalibrace a je nutné je experimentálně určit pro konkrétní kameru.

Tangenciální zkreslení nastává, když optická osa objektivu není kolmá na rovinu, na kterou je promítán obraz. Některé oblasti snímků se potom mohou jevit blíže, než ve skutečnosti jsou. Příklad tangenciálního zkreslení je vyobrazen v pravé části obrázku 5.1.

Podobně jako radiální zkreslení lze i tangenciální vadu tvaru kompenzovat pomocí podobné soustavy rovnic

$$x_{korigovana} = x + [2p_1xy + p_2(r^2 + 2x^2)] \quad , \tag{5.4}$$

$$y_{korigovana} = x + [p_1(r^2 + 2y^2) + 2p_2xy] \quad , \tag{5.5}$$

kde opět platí

$$r^2 = y^2 + x^2 \quad . \tag{5.6}$$

Samotná kalibrace pracuje s modelem zobrazení bodu z reálného světa do obrazu. Model uvažující výše zmíně korekce zkreslení lze popsat rovnicemi podle [41]:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{R} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t} \tag{5.7}$$

$$x' = x/z \tag{5.8}$$

$$y' = y/z \tag{5.9}$$

$$x'' = x'(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) + [2p_1x'y' + p_2(r^2 + 2x'^2)] \tag{5.10}$$

$$y'' = y'(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) + [p_1(r^2 + 2y'^2) + 2p_2x'y'] \tag{5.11}$$

$$r = x'^2 + y'^2 \tag{5.12}$$

$$u = f_x x'' + c_x \tag{5.13}$$

$$v = f_y y'' + c_y \tag{5.14}$$

Transformační matice rotace $\mathbf{R}$ a translační vektor $\mathbf{t}$ udávají polohu kamery v pomyslném globálním souřadnicovém systému. Souřadnice X, Y, Z udávají polohy zkoumaného bodu v reálném světě v globálním souřadnicovém systému. Souřadnice (u, v) udávají

vzdálenost bodu z levého horního rohu na snímku v pixelech. (x', y') a (x'', y'') jsou bezrozměrné souřadnice před resp. po kompenzaci zkreslení. Parametry f_x, f_y udávají ohniskovou vzdálenost objektivu kamery v osách x, resp. y, tato vzdálenost je vyjádřená v pixelech. Konstanty c_x, c_y popisují souřadnice optického středu na snímaném obraze, jejich jednotkou je taktéž pixel.

K výpočtům kalibračních parametrů $(k_1, k_2, k_3, p_1, p_2)$, konstant kamery (f_x, f_y, c_x, c_y) a v neposlední řadě transformační matice a vektoru $\mathbf{R}, \mathbf{t}$ je zapotřebí zaznamenat velké množství bodů. U každého snímaného bodu je potřeba znát nejen jeho souřadnice na snímku kamery, ale je nutné též definovat jeho polohu v prostoru. Pokud v modelu nebude uvažována korekce zkreslení, a bude tedy platit

$$x'' = x' \quad , \tag{5.15}$$

$$y'' = y' \quad , \tag{5.16}$$

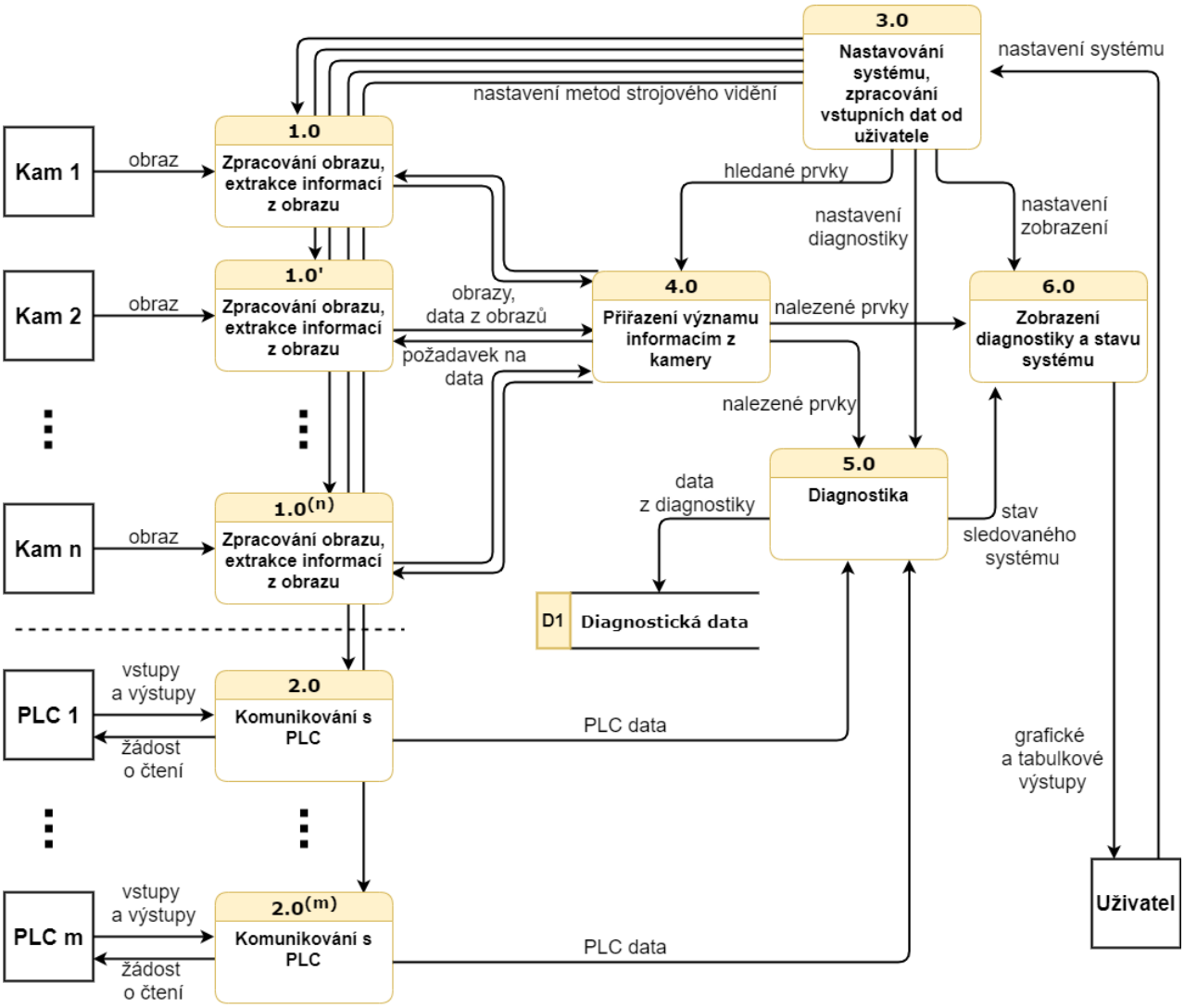
budou souřadnice bodů z reálného světa odpovídat odměřeným souřadnicím na nekalibrovaném snímku. Pro výpočet konstant korekce zkreslení je nutné dosazovat souřadnice (u, v), na kterých by vyšetřovaný bod na snímku měl být, nikoli skutečně naměřené souřadnice (u, v). Toho lze dosáhnout snímáním bodů rozmístěných ve známé geometrii.

Nejběžnějším obrazcem používaným ke kalibraci kamery je šachovnice. Rohy šachovnice jsou snadno detekovatelné a mají jasně stanovenou pozici na šachovnici. V globálním souřadnicovém systému lze šachovnici umístit do roviny xy, potom mají všechny body šachovnice nulovou souřadnici z. Velikost lze definovat tak, že hrana jednoho čtverce na šachovnici bude odpovídat 1, počátek globálního systému lze umístit do jednoho z rohů šachovnice a osy x, y sjednotit s hranami šachovnice. Jednotlivé rohy šachovnice budou poté zaujímat souřadnice $(0, 0, 0), (1, 0, 0), (2, 0, 0), ..., (0, 1, 0), (1, 1, 0), (2, 1, 0), ...$ Samotnou kalibraci je poté možné provést například funkcí `calibrateCamera()` z knihovny OpenCV [42]. Detekci rohů šachovnice lze uskutečnit pomocí funkce `findChessboardCorners()` ze stejné knihovny. Poté, co jsou nalezeny korekční koeficienty tvarových vad, je možné získat nezkreslený snímek funkcí `undistort()`, jejíž vstupní argumenty lze získat z výstupy funkce `calibrateCamera()`.

Pro CAFDIS je kalibrace obrazu potřebná především pro správné vyhodnocení polohy indikačních LED pomocí souřadnic referenčních tagů. Jelikož se předpokládá, že LED může na snímku z kamery měřit jen pár pixelů, je nezbytné určit její polohu co nejpřesněji.

6 Analýza CAFDIS

Základní schéma vnitřní struktury systému CAFDIS je zachyceno pomocí diagramu datových toků na obrázku 6.1.



Obrázek 6.1: Diagram datových toků

K systému lze připojit několik kamer, jejichž počet je omezen pouze hardwarovými limitacemi zařízení, na kterém systém poběží. Každá kamera má přiřazen svůj vlastní proces na zpracování obrazu, který je nastaven uživatelem tak, aby detekoval hledané tagy.

Uživatel v programu vytvoří virtuální modely reálných prvků monitorovaného procesu. Virtuální prvky jsou definovány a popsány pomocí pozic a velikostí hledaných tagů. V procesu 4.0 jsou nalezené tagy porovnány s informacemi o virtuálních prvcích a model

prvku je aktualizován, díky čemuž lze vyhodnotit jeho momentální stav nebo pozici, například natočení motoru nebo barvu indikační LED.

O komunikaci s PLC se stará proces 2.0, který je rovněž unikátní pro každý připojený PLC a konfigurovatelný podle aktuálních potřeb a požadavků.

Data z PLC a virtuálních prvků jsou zpracována v procesu 5.0. Podle rozhodovacích podmínek, které rovněž systému zadá uživatel, proces diagnostiky vyhodnotí aktuální stav systému jako přípustný, nebo jako chybový. Data o stavu systému jsou průběžně ukládána, v případě poruchy monitorovaného procesu nebo systému CAFDIS mohou napomoci ke zjištění příčiny poruchy. Výstup diagnostiky i podrobná data ostatních procesů může uživatel sledovat prostřednictvím uživatelského rozhraní zastoupeného procesem 6.0.

6.1 Implementace

Program CAFDIS je napsán v jazyce Python [43] s využitím především knihoven OpenCV [42] a PyQT5 [44].

Zdrojový kód programu byl kvůli přehlednosti rozdělen do dvou souborů:

- **Core.py**
  obsahuje většinu funkcí z procesů 1.0, 2.0, 4.0 a 5.0. V souboru jsou implementovány všechny algoritmy popsané v této stati práce. Sám o sobě není soubor *core.py* spustitelný, ale obsahuje množinu všech důležitých objektů a funkcí potřebných k běhu programu.

- **GUI.py**
  je nadstavba souboru *core.py*. Využívá jeho objekty a funkce a řadí a řetězí jejich výstupy a vstupy do logického funkčního celku. Hlavním účelem tohoto souboru je ovšem vytvoření grafického uživatelského rozhraní (GUI), skrze které přebírá kontrolu nad spuštěním jednotlivých funkcionalit souboru *core.py* uživatel. Soubor *GUI.py* také řeší multithreading celé aplikace pomocí nástrojů PyQT5 a pythonovského modulu *threading*.

Multithreading

Jednotlivá vlákna jsou tvořena objekty, které dědí z třídy `QThreading` knihovny PyQT5. Metoda `run()` takového objektu běží po jejím zavolání skrze metodu `start()` v samostatném vláknu.

Systém CAFDIS používá pro proces 1.0 každé z kamer samostatné vlákno. Proces 4.0 je pomyslně rozdělen mezi příslušná vlákna procesů 1.0. Samostatné vlákno má také každý proces na komunikaci s PLC 2.0. Posledním procesem se samostatným vláknem je proces

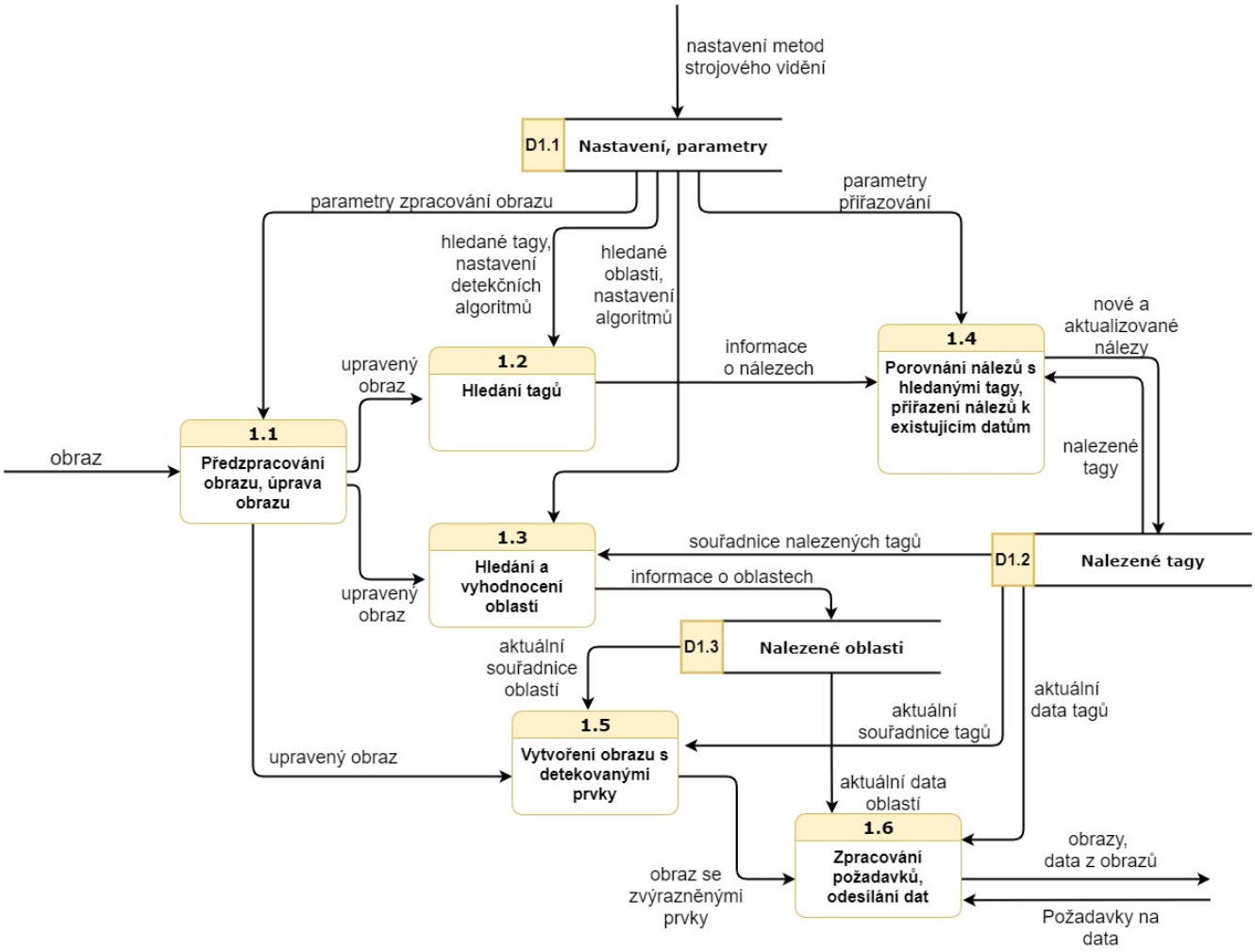
diagnostiky 5.0. Zbývající procesy 3.0 a 6.0 a některé další podprocesy týkající se GUI běží na výchozím vlákně aplikace.

Je nutné zmínit, že vlákna `QThreading` nemusí odpovídat a většinou neodpovídají fyzickým vláknům procesoru. Místo toho jsou v programu vedena jako vlákna virtuální a výpočetní výkon je jim přiřazován dynamicky podle výpočetní náročnosti a nastavené priority. Přiřazení procesorového času jednotlivým vláknům řeší knihovna PyQT5.

Komunikace a synchronizace jednotlivých vláken mezi sebou je řešena pomocí signálů z knihovny PyQT5. Pro bližší informace o fungování multithreadingu pomocí této knihovny je možné nahlédnout do [44] nebo [45].

6.2 Proces zpracování obrazu

Schéma procesu zpracování obrazu je zachyceno pomocí diagramu datových toků na obrázku 6.3.



Obrázek 6.2: Diagram datových toků procesu 1.0

Obraz z kamery je nejprve předzpracován v procesu 1.1. Jde především o kalibraci snímku pro digitální odstranění optických vad, úpravu kontrastu nebo změnu rozlišení. Upravený snímek je předán procesům *hledání tagů* (1.2) a *hledání a vyhodnocování oblastí* (1.3).

Výstupem procesu 1.2 je matice obsahující informace o jednotlivých nálezech: především souřadnice, velikost a typů nalezeného tagu. Tato data dále putují do procesu 1.4.

Proces 1.4 aktualizuje objekty uložené v datové paměti $D1.2$ *nalezené tagy*. Každý uložený objekt reprezentuje jeden tag. Objekt obsahuje zejména informace o poloze, velikosti a typu nalezeného tagu, v neposlední řadě také čas poslední úspěšné detekce a míru spolehlivosti dané detekce.

V procesu 1.3 jsou konkrétní tagy z datové paměti $D1.2$ použity pro vyhodnocení aktuálních poloh hledaných oblastí. Každá oblast je výřezem z původního snímku. Referenční tagy zaručují, že se výřez bude nacházet stále na stejném místě z pohledu globálního souřadného systému v reálném prostředí. Proces dále vyhodnotí barvu každé nalezené oblasti. Na základě barvy je oblasti přiřazen stav. Systém CAFDIS zatím podporuje rozlišení dvou stavů. Informace o nalezených oblastech a jejich stavech jsou uloženy do datové paměti $D1.3$.

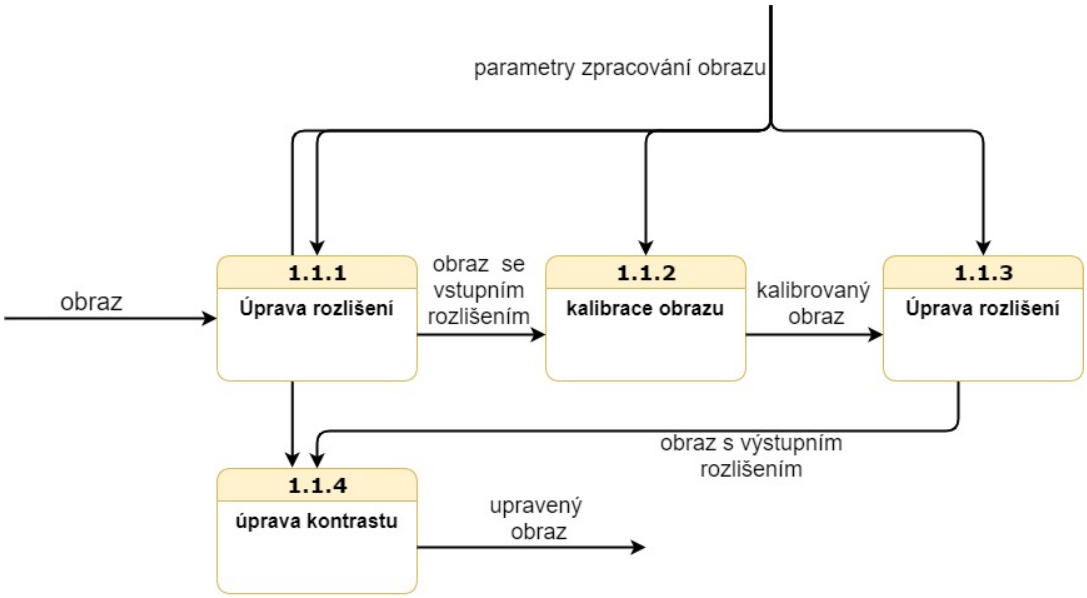
Proces 1.5 do obrazu graficky zaznačí nalezené tagy a oblasti. Tento obraz je poté prostřednictvím procesu 1.6 odeslán dále, kde je nakonec zobrazen prostřednictvím GUI.

Datová paměť $D1.1$ obsahuje všechna potřebná data statického charakteru, která zadává uživatel podle zkoumaného procesu. Mezi tato data patří vzhled a počty hledaných tagů, parametry předzpracování obrazu, relativní souřadnice oblastí vůči referenčním tagům, klíče pro vyhodnocení stavu oblastí a mnohé další.

6.2.1 Proces předzpracování obrazu

Schéma procesu zpracování obrazu je zachyceno pomocí diagramu datových toků na obrázku 6.3.

Nejdůležitějším předzpracováním obrazu je kalibrace, které se věnovala kapitola 5. Proces 1.1.2 předpokládá znalost všech potřebných kalibračních koeficientů z dřívější kalibrace a obsahuje jen jednoduché odstranění zkreslení pomocí některé z funkcí knihovny OpenCV. Z postupu procesu pro nalezení kalibračních koeficientů vyplývá, že tyto koeficienty jsou pevně spjaty s použitým rozlišením. Důkazem mohou být rovnice modelu kalibrace $u = f_x x'' + c_x, v = f_y y'' + c_y$, kde parametry (c_x, c_y) jsou souřadnice optického středu snímku v pixelech. Parametry (c_x, c_y) budou tedy rozdílné pro různá rozlišení. Řešením tohoto problému může být striktní dodržování rozlišení příslušné kalibrace, přepočet všech kalibračních koeficientů do nového rozlišení nebo provedení nové kalibrace pro nové rozlišení. Systém CAFDIS umožňuje provést novou kalibraci a jednotlivé ka-



Obrázek 6.3: Diagram datových toků procesu 1.1

librační koeficienty měnit během chodu aplikace, stejně tak jako měnit rozlišení před a po kalibraci tak, aby byla kalibrace prováděna na obrazu konstantní velikosti.

Úprava kontrastu v procesu 1.4 slouží k vyvážení rozložení intenzit barev napříč pořízeným snímkem. Implementovaná metoda umožňuje pomocí algoritmu *Contrast Limited Adaptive Histogram Equalization* (CLAHE) vyrovnat histogramy intenzit lokálních oblastí v obraze a tím zajistit dobrý kontrast nejen obrazu jako celku, ale i dílčích objektů. Navíc je schopna částečně odstranit šum díky omezení maximálního kontrastu malých oblastí. [46]

Většina moderních digitálních kamer koriguje kontrast do velké míry automaticky a řazení CLAHE filtru proto nemusí být vždy nutné, nebo ani žádoucí, jelikož se jedná o další zátěž pro výpočetní výkon. Proto je úprava kontrastu v systému CAFDIS vypínatelnou funkcí.

Kalibrace

Samotnou kalibraci iniciuje uživatel prostřednictvím GUI. Kalibrace otevře nové okno s obrazem z vybrané kamery. Uživatel kameru namíří na připravenou šachovnici rozměru 8x8 pokud možno umístěnou do roviny tak, aby šachovnice zabírala většinovou část plochy snímku. Kalibrační podprogram automaticky sejme snímky v nejméně půlsekundových časových rozestupech. Pokud je snímek podprogramem vyhodnocen jako dobrý, tzn. na snímku jsou detekovatelné všechny potřebné body šachovnice s dostatečnou ostrostí, je snímek uložen pro další zpracování. Uživatel mezitím pohybuje kamerou nebo šachovnicí tak, aby uložené snímky zachycovaly šachovnici v rozličných polohách a pod různými úhly. Tento proces končí ve chvíli, kdy je uloženo 30 validních snímků.

Poté je okno podprogramu automaticky zavřeno a program vyhodnotí souřadnice rohů šachovnice se subpixelovou přesností pomocí funkce `cornerSubPix()` z knihovny OpenCV. Subpixelová přesnost je určena na základě intenzit pixelů v okolí hrany. Jedná se o odhad místa s největším gradientem roviny proložené intenzitami pixelů. Dále program řeší soustavu rovnic z kapitoly 5 funkcí `calibrateCamera()` z knihovny OpenCV.

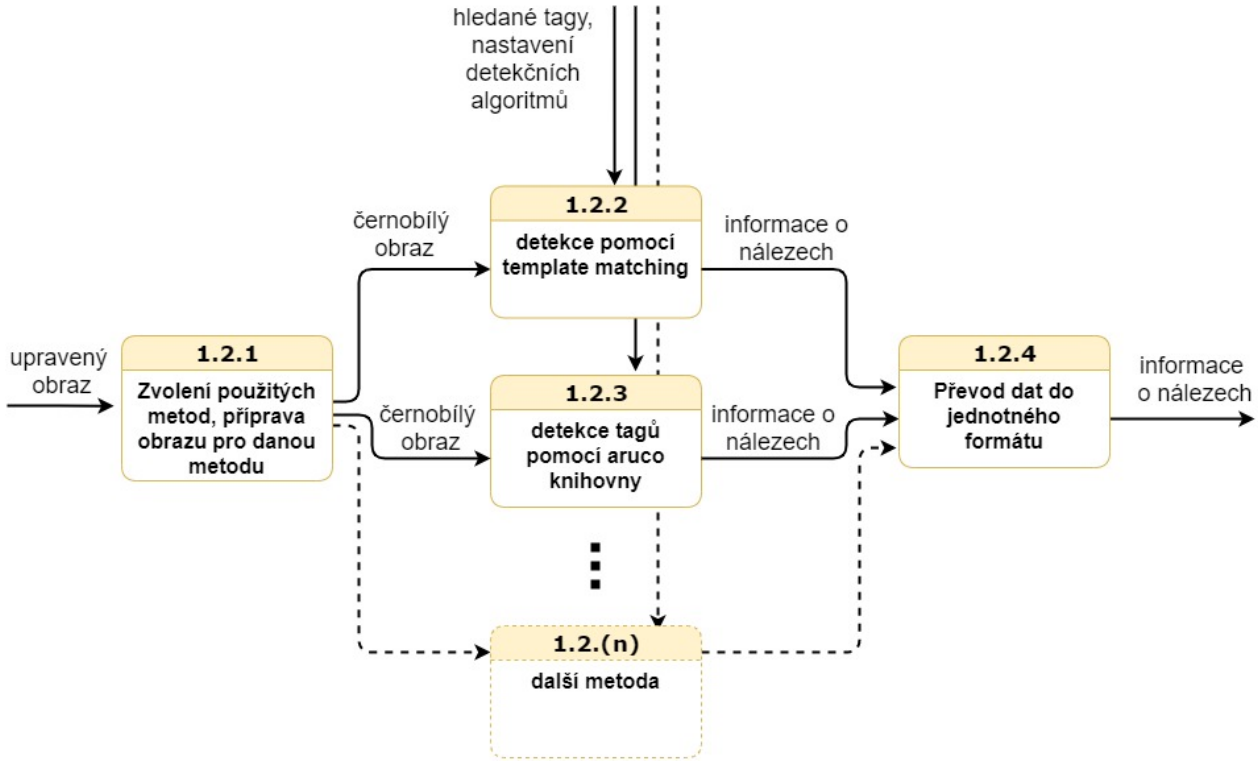
Obrazy, na kterých byla kalibrace prováděna, jsou posléze zkalibrovány a je určena absolutní odchylka kalibrovaných souřadnic rohů šachovnice od ideálního tvaru šachovnice. Průměr jednotlivých odchylek přes všechny body a snímky je zobrazen v terminálu GUI jako kalibrační chyba.

Poslední funkcí podprogramu je uložení kalibračních koeficientů do složky *Kalibrace* pod názvem kamery s datem a časem provedené kalibrace. Kalibraci je poté nutné ještě načíst do CAFDIS stiskem tlačítka *procházet soubory* v sekci věnované kalibraci v GUI.

Kalibrační podprogram je umístěn ve funkci `calibrateCamera` třídy `Camera`.

6.2.2 Proces hledání tagů

Schéma procesu zpracování obrazu je zachyceno pomocí diagramu datových toků na obrázku 6.4.



Obrázek 6.4: Diagram datových toků procesu 1.2

Předzpracovaný obraz je v procesu 1.2.1 ještě dále upraven, aby co nejlépe vyhovoval potřebám dané metody. Jak již bylo řečeno v části 4.1, systém CAFDIS dokáže detekovat

tagy pomocí dvou metod: Template Matching (TM) a Systémy referenčních tagů (FMS). Systém je navržen tak, aby byl lehce rozšiřitelný o další metody. Proces 1.2.4 převádí data z detekčních algoritmů do jednotného formátu. Data jsou v každém průchodu algoritmu uložena do matice. Jednotlivé řádky odpovídají jednotlivým tagům a obsahují informaci o souřadnicích, velikosti a typu tagu, míře spolehlivosti a času detekce.

Hledání tagů pomocí TM metody

Jak již bylo nastíněno v kapitole 3.3, hledání tagů pomocí TM metody je realizováno iterativně pro všechny myšlené velikosti nebo natočení hledaného vzoru. Nastavení tohoto iteračního cyklu je v rukou uživatele systému. Systém CAFDIS v současné verzi podporuje hledání rotačně invariantních tagů a umožňuje pro každý hledaný vzor pomocí TM nastavit minimální hledanou velikost tagu V_m, maximální hledanou velikost tagu V_M a velikost kroku k, o kterou se hledaný vzor v každé iteraci zvětšuje z minimální postupně až na maximální myšlenou velikost.

Mimo to lze nastavit koeficient zmenšení f. Koeficientem zmenšení je vyděleno rozlišení obrazu i hledaného tagu, obraz i tag jsou převedeny do nového rozlišení, ve kterém je aplikována TM metoda. Jelikož nese zmenšený obraz méně detailů, jsou výsledky detekce méně spolehlivé a mohou se objevit zejména falešně pozitivní nálezy. Výhodou je ovšem značné zrychlení celého procesu. Pokud je koeficient zmenšení $f = 2$, prohledává algoritmus jen čtvrtinové množství pixelů. Protože je rozlišení obrazu koeficientem zmenšeno jen pro TM metodu, zůstanou detaily obrazu zachovány pro jiné metody v procesu 1.2.

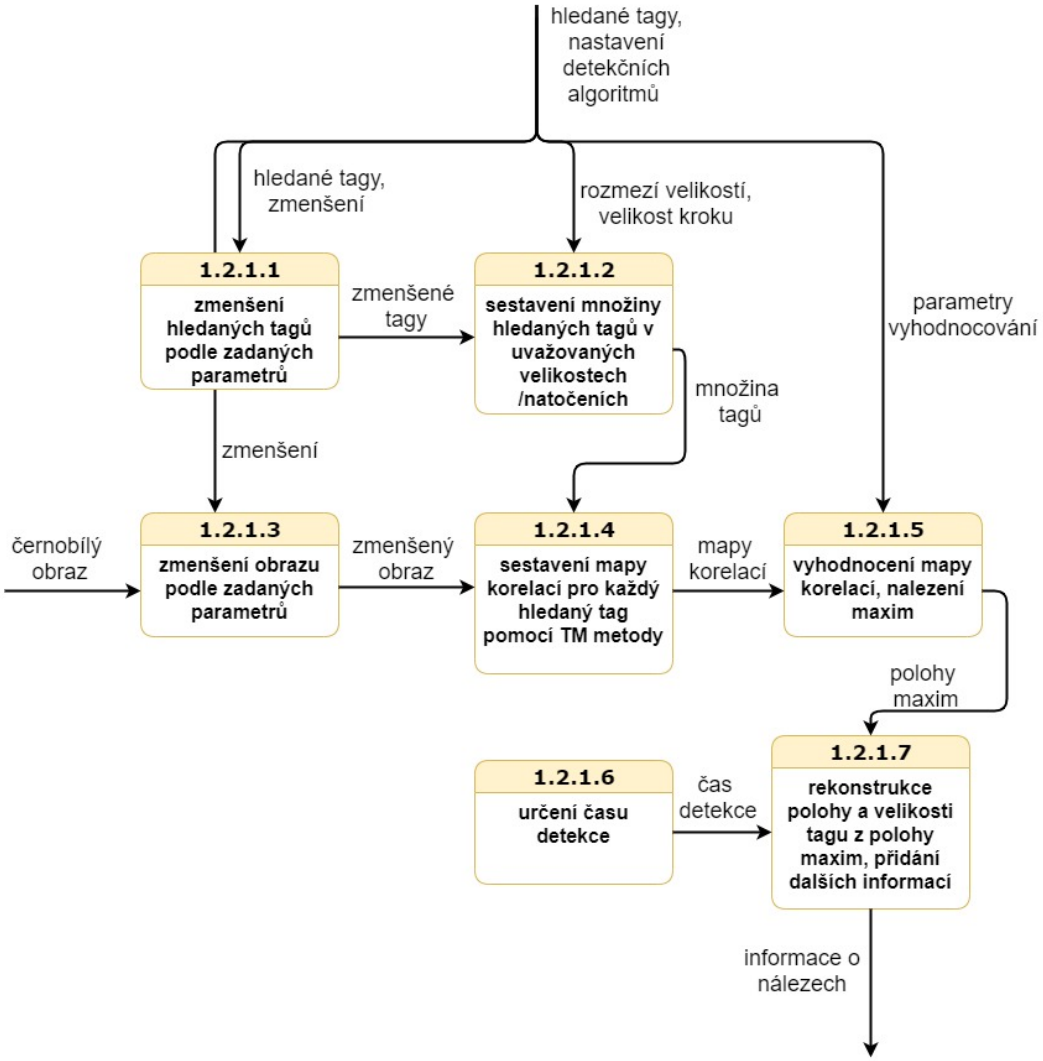
Procesu hledání tagů pomocí TM metody je zobrazen pomocí diagramu datových toků na obrázku 6.5.

Procesy 1.2.1.1 a 1.2.1.3 převádějí hledaný tag, resp. zkoumaný obraz do nového rozlišení daného uživatelským nastavením. Proces 1.2.1.2 vygeneruje pole tagů všech myšlených velikostí, taktéž podle uživatelského nastavení. Proces 1.2.1.4 počítá samotnou mapu korelačních koeficientů pomocí funkce `matchTemplate()` z knihovny OpenCV [42]. Funkce `matchTemplate()` obsahuje několik algoritmů výpočtu mapy korelačních koeficientů, jádrem všech z nich je Diskrétní Fourierova transformace (DFT), kterou lze k výpočtu s výhodou použít, protože je řádově rychlejší než klasičtější přístupy. Pro každou myšlenou velikost tagu je vygenerována jedna 2D mapa. Dvojrozměrné mapy jsou sloučeny do trojrozměrného pole korelačních koeficientů K.

$$K = K(x, y, z) \tag{6.1}$$

Kde z udává pořadové číslo 2D mapy a x, y jsou souřadnice v této mapě.

Mapy korelačních koeficientů K jsou zpracovány v procesu 1.2.1.5. Parametry tohoto



Obrázek 6.5: Diagram datových toků procesu 1.2.1

procesu jsou především počet hledaných tagů a nejmenší přípustná hodnota korelačního koeficientu. Parametry opět nastavuje uživatel skrze GUI.

Pro hledání souřadnic n tagů se nabízí možnost vybrat n maximálních hodnot z pole K. Tento způsob identifikace ovšem povede k vícenásobným detekcím stejného tagu (nálezy budou vzájemně posunuty jen o pár pixelů).

Proto pracuje algoritmus vyhodnocení map podle následujících kroků:

1. Nalezení maxima M přes všechny mapy korelačních koeficientů K.

2. Ověření, zda je M větší než prahová hodnota korelačního koeficientu nastavená uživatelem. Pokud je hodnota maxima menší, algoritmus je ukončen.

3. Extrakce polohy maxima (X_M, Y_M, Z_M), kde souřadnice Z_M udává číslo 2D mapy, která obsahuje maximum, a X_M, Y_M jsou souřadnice maxima v mapě Z_M. Pokud je maximální hodnota přítomna na více místech, je uvažováno pouze první z nich.

4. Uložení nálezu maxima M a jeho souřadnic (X_M, Y_M) do pole nálezů.

5. Ověření, zda bylo již nalezeno hledané množství tagů – v takovém případě je algoritmus ukončen.

6. Vynulování hodnot map korelačních koeficientů podle vztahu

$$\begin{gathered} \forall t, \forall u, \forall w; \quad K(X_M + t, Y_M + u, w) = 0; \\ t \in < -d; d >; \quad u \in < -d; d >; \quad w \in < 0, p >, \end{gathered} \tag{6.2}$$

   kde p je počet jednotlivých 2D map v K, tedy i počet uvažovaných velikostí hledaného tagu. Parametr d udává velikost oblasti se středem v bodě (X_M, Y_M), která bude v každé z map vynulována.

   Tím jsou odstraněny potenciálně pozitivně vyhodnocené body v okolí již potvrzených nálezů, další nález se tedy může nacházet nejblíže ve vzdálenosti d od dřívějších nálezů.

7. Opakování algoritmu do jeho ukončení některou z podmínek.

Po ukončení algoritmu procesu 1.2.1.5 je pole nálezů předáno procesu 1.2.1.7, který nejprve vypočítá velikost V každého nalezeného tagu podle rovnice

$$V = V_m + k \cdot Z_M \quad , \tag{6.3}$$

kde V_m značí minimální hledanou velikost, k je velikost kroku a Z_M je souřadnice polohy nálezu odpovídající dané velikosti tagu. Dále jsou souřadnice tagu přepočítány do středu tagu (původně souřadnice označovaly levý horní okraj). Hodnota M je považována za míru spolehlivosti detekce. Každému detekovanému tagu je přiřazen kód odpovídající hashi hledaného vzoru. Na závěr je každému nálezu přiřazen čas detekce, který odpovídá času pořízení vyšetřovaného snímku.

Hledání tagů pomocí knihovny ArUco

Proces 1.2.3 staví z velké části na bohatých možnostech knihovny ArUco, která je součástí knihovny OpenCV. Stručný popis detekce je shrnut v několika krocích.

1. Sestavení objektů hledaných sad ArUco tagů podle zadaní uživatele. Objekty jsou vytvořeny funkcí `aruco.Dictionary_get(dict)`, kde `dict` je identifikátor sady tagu, který lze specifikovat buď odpovídajícím číselným kódem, nebo makrem například ve tvaru `aruco.DICT_6X6_50`. Označení `DICT_6X6_50` lze interpretovat tak, že daná sada tagů obsahuje štítky s 2D kódem o rozměru 6x6, který je tím pádem teoreticky

schopný reprezentovat celkem 2^{36} unikátních čísel. Sada `DICT_6X6_50` ovšem obsahuje podle posledního uvedeného čísla jen 50 unikátních tagů. Nevyužitý potenciál počtu kombinací je použit na zabezpečení detekovaného identifikátoru tagu pomocí paritních bitů a CRC. Díky tomu je detekční algoritmus nejen schopen odhalit drtivou většinu falešně pozitivních nálezů, ale lze do jisté míry i kompenzovat vlivy okluze na některé části tagu a správně rekonstruovat identifikátor tagu. [47]

2. Vytvoření parametrů detekce pomocí funkce `aruco.DetectorParameters_create()`. Tyto parametry lze teoreticky nadále upravovat nebo nastavovat ručně. Experimentálně bylo zjištěno, že výchozí nastavení parametrů je pro účely systému CAFDIS dostačující. Proto nastavení těchto parametrů není skrze GUI podporováno.

3. Provedení samotné detekce hledané sady tagů na zkoumaném snímku pomocí funkce `aruco.detectMarkers(gray,aruco_dict,parameters=parameters)`, kde `gray` je černobílý obraz, na kterém je detekce prováděna, `aruco_dict` je hledaná sada tagů a `parameters` je slovník parametrů detekce. Funkce předává programu 3 návratové hodnoty:

   - vektor souřadnic rohů všech úspěšně identifikovaných tagů,
   - vektor identifikátorů všech úspěšně identifikovaných tagů,
   - vektor souřadnic rohů nepotvrzených nálezů.

4. Určení velikosti každého tagu ze souřadnic rohů jako průměrnou vzdálenost sousedících rohů pomocí euklidovské nebo manhatonské metriky. Určení souřadnic tagu jako těžiště všech rohů daného tagu.

Všechny úspěšně nalezené tagy jsou zobrazeny v tabulkové podobě v GUI. V tabulce jsou k dispozici všechny zjištěné údaje o tazích, včetně barevné indikace spolehlivosti detekce a času poslední úspěšné detekce. Po kliknutí na záznam z tabulky je tag graficky označen na snímku z kamery. Podrobněji se GUI věnuje kapitola 6.6.

6.2.3 Přiřazení nalezených tagů k existujícím datům

Proces 1.4 je zodpovědný za správné uložení dat z procesu 1.2 do datové paměti $D1.2$. Do procesu 1.4 vstupují informace o nálezech v tabulkové podobě, v paměti $D1.2$ jsou data uložena do objektů třídy `TagData`. Každý nalezený tag má vlastní objekt, ve kterém jsou uloženy informace o typu tagu, jeho ID, popřípadě hash a odkazy na další tagy, které slouží jako referenční tagy pro účely výpočtu transformačních matic a definování souřadných systému, jak bude rozebráno v dalších kapitolách. Dále obsahují několik parametrů týkajících se převážně jejich zobrazení v GUI. Nejdůležitějším atributem

objektu je proměnná `data`, kde jsou uloženy informace o jednotlivých detekcích v čase (poloha, velikost, čas detekce, spolehlivost) v tabulkové podobě.

Správné přiřazení dat z detekce mezi jednotlivé objekty třídy `TagData` je tedy klíčové pro spoustu dalších funkcí (určení poloh oblastí, určení poloh aktuátorů...) systému CAFDIS.

Je jasné, že metoda přiřazení se bude lišit podle povahy detekovaných tagů a detekčních algoritmů. Tagy detekované knihovnou ArUco s sebou nesou informaci o svých ID, tedy jednoznačný identifikátor každého tagu, na základě kterého lze přiřazení provést jednoznačně. Tagy detekované TM metodou jsou označeny hashem hledaného vzoru. Pokud je hledáno více tagů stejného vzoru, je nutné je mezi sebou rozlišit.

Pro snadnější inicializaci programu je možné v GUI pomocí políčka s názvem *hledat nové tagy* měnit funkci přiřazovacího algoritmu. Pokud je hledání nových tagů povoleno, je pro každý nález, který se nepovede vybranou metodou přiřadit, vytvořen nový objekt třídy `TagData`. Poté, co jsou všechny hledané tagy alespoň jednou úspěšně detekovány, je vhodné hledání nových tagů vypnout. Díky tomu nebude při nepovedené detekci růst celkový počet tagů, který by posléze při přiřazování dalších nálezů způsobil další chyby.

Systém CAFDIS umožňuje přiřazovat TM tagy jednou ze dvou metod. O výběru použité metody rozhoduje parametr `simplePairingMethod` v objektu `params` ve třídě `Tag`. Tento parametr je nastaven na hodnotu *True* a prostřednictvím GUI jej zatím nelze měnit. Systém tedy bez programového zásahu používá jednoduchou přiřazovací metodu.

Jednoduchá přiřazovací metoda

Jednoduchá přiřazovací metoda provede pro každý nález následující operace:

1. Vypočítání množiny vzdáleností d_i mezi zkoumaným nálezem a i-tým již dříve detekovaným objektem třídy `TagData` se stejným hashem podle vzorce

$$d_i = |x_n – x_{di}| + |y_n – y_{di}| + k \cdot \frac{max(V_n, V_{di}) - min(V_n, V_{di})}{min(V_n, V_{di})} \quad , \tag{6.4}$$

   kde (x_n, y_n) jsou souřadnice nálezu a (x_{di}, y_{di}) souřadnice i-tého dříve nalezeného tagu. V_n a V_{di} udávají velikosti nálezu resp. dříve nalezeného i-tého tagu. Parametr k udává citlivost vzdálenosti vzhledem k rozdílu velikostí V_n a V_{di} a je nastaven na hodnotu 100; měnit lze pouze v kódu, nikoli pomocí GUI.

2. Nalezení minima vzdáleností d_i a odpovídajícího nejbližšího objektu třídy `TagData`; nalezené minimum musí být menší než hodnota daná parametrem `distanceTM` z objektu `params`.

3. Přiřazení informací o vyšetřovaném nálezu nejbližšímu objektu třídy `TagData`.

Tento algoritmus neošetřuje případy, kdy by jednomu objektu třídy `TagData` bylo přiřazeno více nálezů, i když je toto chování nežádoucí. Nelze totiž zaručit, že nejdříve přiřazený nález do daného objektu skutečně patří, ukáže-li se, že i další nález je zmíněnému objektu nejblíže.

Komplexní přiřazovací metoda

Nedokonalost jednoduché metody odstraňuje komplexní přiřazovací metoda, která nepřiřazuje nálezy jednotlivě, ale počítá vhodnost dané kombinace přiřazení přes všechny nálezy a objekty třídy `TagData`. Postupuje následovně:

1. Roztřídění TM nálezů do skupin $S_{h1}, S_{h2}, \ldots$ se stejným hashem. Každá skupina následně prochází algoritmem zvlášť.

2. Vytvoření skupin tagů $T_{h1}, T_{h2}, \ldots$ z objektů třídy `TagData` se stejnými hashi nalezenými v kroku 1.

3. Porovnání velikostí skupin S_{hi} a T_{hi}. Doplnění na stejnou velikost prázdnými objekty tak, že počet prvků S_{hi} i T_{hi} je N_i.

4. Sestavení všech permutací $P_{i,j}$ skupiny S_{hi}.

5. Vyhodnocení celkové míry rozdílnosti $m_{i,j}$ jednotlivých permutací $P_{i,j}$ se skupinou T_{hi} podle vztahu

$$m_{i,j} = \sum_{k=1}^{N_i} \left(|x_{Pi,j,k} – x_{Ti,k}| + |y_{Pi,j,k} – y_{Ti,k}| + \right. \\ \left. + c \cdot \frac{max(V_{Pi,j,k}, V_{Ti,k}) - min(V_{Pi,j,k}, V_{Ti,k})}{min(V_{Pi,j,k}, V_{Ti,k})} \right)^d \quad , \quad (6.5)$$

   kde $x_{Pi,j,k}, y_{Pi,j,k}$ jsou souřadnice k-tého tagu permutace $P_{i,j}$ a $x_{Ti,k}, y_{Ti,k}$ jsou souřadnice k-tého tagu skupiny T_{hi}. $V_{Pi,j,k}$ a $V_{Ti,k}$ odpovídají velikostem k-tého tagu permutace $P_{i,j}$, resp. skupiny T_{hi}. Pokud byl k-tý tag skupiny S_{hi} nebo T_{hi} v kroku 3 nahrazen prázdným objektem, je daný sčítanec sumy roven 0. Parametr c udává citlivost míry rozdílnosti na rozdíl velikostí porovnávaných tagů, parametr d lze označit jako citlivost na odlehlé hodnoty, tedy vzdálenější tagy v porovnání.

6. Nalezení nejmenší míry rozdílnosti

$$m_{i,min} = \min_j m_{i,j}$$

   a permutace $P_{i,min}$ příslušící danému minimu.

7. Přiřazení nalezených tagů skupiny S_{hi} objektům třídy `TagData` ve skupině T_{hi} v pořadí daném permutací $P_{i,min}$.

Tato metoda není vhodná pro velké počty TM tagů se stejným hashem, jelikož její výpočetní složitost lze odvodit jako $O(N_i! \cdot N_i)$ (N_i sčítanců v sumě pro každou z $N_i!$ permutací). Z těchto důvodů není možné metodu zvolit skrze GUI. Metodu lze vybrat manuálně změnou proměnné `simplePairingMethod` ve třídě `Parameters` na hodnotu *False*. Samotná metoda je k nalezení ve třídě `Tag` ve funkci `updateTagsData` pod statí `if not self.params.getSiplePairingMethod():`, kde je možné ji optimalizovat nebo změnit.

6.2.4 Hledání a vyhodnocení oblastí

Oblasti jsou v programu reprezentovány souřadnicemi čtyřúhelníku, který ohraničuje výřez obrazu. Souřadnice čtyřúhelníku závisí na přiřazených referenčních tazích, čtyřúhelník proto mění svou polohu v závislosti na změně poloh přiřazených tagů. Tím je nejen zajištěna invariantnost detekce na posuv kamery, ale lze i detekovat oblasti na pohyblivých částech zařízení, pokud je tato pohyblivá část označena referenčními tagy.

Pro přepočet souřadnic čtyřúhelníku do nové polohy je potřeba znát polohy přiřazených referenčních tagů ($(x_{1,ref}, y_{1,ref})$, $(x_{2,ref}, y_{2,ref})$, ...) dané oblasti ve výchozím stavu, který byl ideálně určen ze statického snímku, a souřadnice oblasti ve výchozím stavu ($(u_{1,ref}, v_{1,ref})$, $(u_{2,ref}, v_{2,ref})$, ...). V libovolném okamžiku detekce po výchozím nastavení jsou souřadnice referenčních tagů ($(x_{1,akt}, y_{1,akt})$, $(x_{2,akt}, y_{2,akt})$, ...). Cílem procesu 1.3 je nalezení souřadnic ($(u_{1,akt}, v_{1,akt})$, $(u_{2,akt}, v_{2,akt})$, ...) hledané oblasti.

Pro sestavení rovnic popisujících výše zmíněný výpočet byly použity funkce knihovny OpenCV `getAffineTransform()` a `getPerspectiveTransform()`. Tyto funkce dokáží nalézt transformační matice mezi souřadnými systémy definovanými 3, resp. 4 odpovídajícími body. Postup určení souřadnic oblasti se liší podle počtu použitých referenčních tagů.

- **1 referenční tag** definuje pouze polohu souřadného systému, nikoli natočení. Pomocí souřadnic 1 referenčního tagu ve výchozí a aktuální poloze lze vypočítat posuv souřadného systému spojeného s daným tagem jako prostý rozdíl souřadnic v obou polohách. Tento posuv lze reprezentovat transformační maticí

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & x_{1,akt} - x_{1,ref} \\ 0 & 1 & y_{1,akt} - y_{1,ref} \\ 0 & 0 & 1 \end{pmatrix} \quad . \tag{6.6}$$

- **2 referenční tagy** definují nejen posuv, ale i natočení a zvětšení jedné osy celého souřadného systému. Místo přímého určení transformační matice systém definuje 3.

virtuální tag v poloze dané rovnicemi

$$\begin{aligned} x_{3,ref} &= x_{1,ref} + y_{1,ref} - y_{2,ref} \quad , \\ y_{3,ref} &= y_{1,ref} + x_{2,ref} - x_{1,ref} \quad , \\ x_{3,akt} &= x_{1,akt} + y_{1,akt} - y_{2,akt} \quad , \\ y_{3,akt} &= y_{1,akt} + x_{2,akt} - x_{1,akt} \quad . \end{aligned}$$

Tyto 3 tagy vždy tvoří rovnoramenný pravoúhlý trojúhelník a jsou použity pro výpočet transformační matice pomocí 3 známých tagů.

- **3 referenční tagy** určují navíc ještě zkosení a změnu měřítka zvlášť pro každou osu. Výpočet transformační matice je realizován pomocí funkce `getAffineTransform()`, která přijímá vektor 3 referenčních souřadnic a vektor 3 nových souřadnic. Funkce vrací transformační matici velikosti $2x3$, proto je k matici ještě přidán řádek $(0, 0, 1)$, čímž se zajistí unifikovaný rozměr všech matic pro další výpočty.

- **4 referenční tagy** nesou informaci o perspektivě, se kterou se musí v reálných aplikacích při velkých pozorovacích úhlech počítat. Funkce `getPerspectiveTransform()` přijímá celkem 4 odpovídající si souřadnice bodů. Výstupem je transformační matice 3x3, jejíž poslední řádek nemusí obsahovat $(0, 0, 1)$.

Souřadnice oblasti lze pomocí transformační matice $\mathbf{T}$ spočítat pomocí vztahu

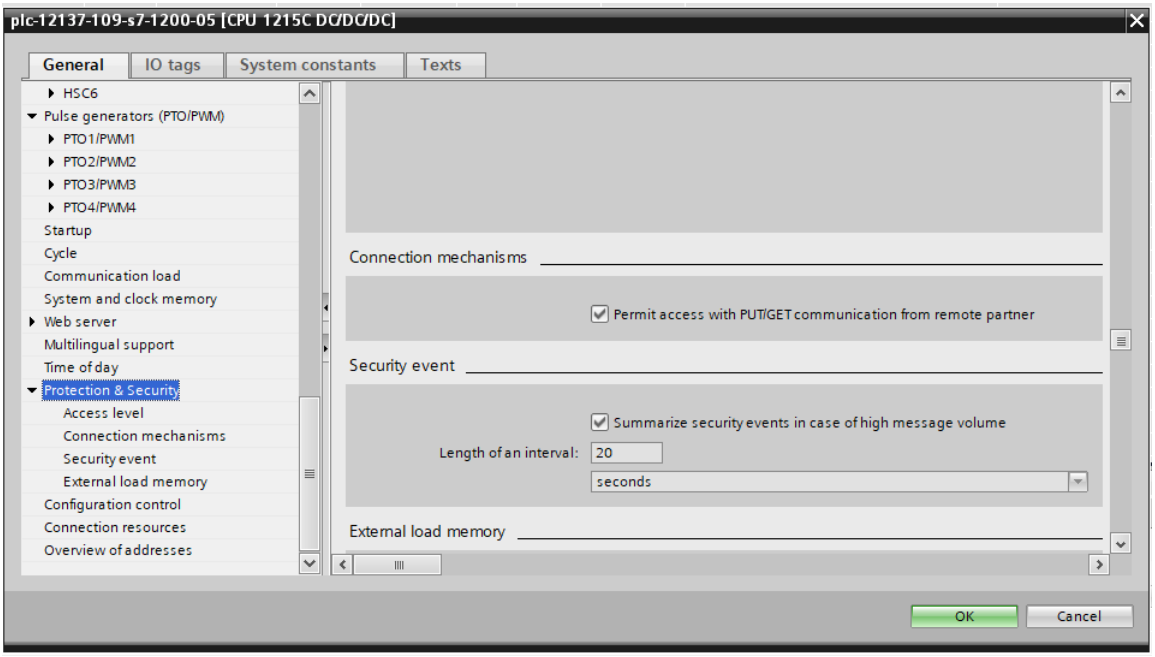
$$\begin{bmatrix} u_{i,akt} \cdot w \\ v_{i,akt} \cdot w \\ w \end{bmatrix} = \mathbf{T} \cdot \begin{bmatrix} u_{i,ref} \\ v_{i,ref} \\ 1 \end{bmatrix} \quad . \tag{6.7}$$

Kvůli uvažované perspektivě nemusí být poslední prvek výstupního vektoru $w = 1$. V takovém případě je nutné hodnoty výstupního vektoru hodnotou w vydělit – tím lze dostat hledané hodnoty $(u_{i,akt}, u_{i,akt})$ [48]. Se znalostmi souřadnic oblasti v aktuální poloze lze určit výřez monitorovaného obrazu a tak získat vzorek barvy oblasti pro další zpracování. Systém CAFDIS používá pro popis oblasti obdélník, jehož střed je popsán souřadnicemi (u_1, v_1). Souřadnice středu obdélníku $(u_{1,akt}, v_{1,akt})$ jsou vypočítány podle výše uvedených principů. Délky hran obdélníku jsou fixně nastaveny a transformací souřadnic se nemění. Hrany jsou taktéž vždy rovnoběžné s osami zkoumaného snímku. Referenční pozici tagů a oblasti nastaví uživatel prostřednictvím GUI. Při vytvoření nové oblasti je uživatel vyzván, aby označil oblast tažením myši na snímku zkoumaného procesu. Souřadnice středu a rozměry označené oblasti jsou uloženy a zobrazeny v GUI, kde je možné je manuálně editovat. Uživatel poté přiřadí oblasti 0 až 4 referenční tagy z tabulky nalezených tagů. Podrobněji se GUI věnuje kapitola 6.6.

6.3 Komunikace s PLC

Proces 2.0 zprostředkovává komunikaci mezi systémem CAFDIS a PLC monitorujícím zkoumaný proces. Komunikaci zajišťuje knihovna Snap7 [49], která slouží pro ovládání a monitorování chodu PLC firmy Siemens.

Před inicializací komunikace mezi systémem CAFDIS a PLC je nejdříve vhodné povolit komunikaci na straně PLC. Toho lze docílit nastavením parametrů prostřednictvím programu SIMATIC STEP 7 [50]. V *Settings* → *General* → *Protection & Security* → *Connection mechanisms* je nutné zaškrtnout pole *Permit acces with PUT / GET communication with remote partner*.

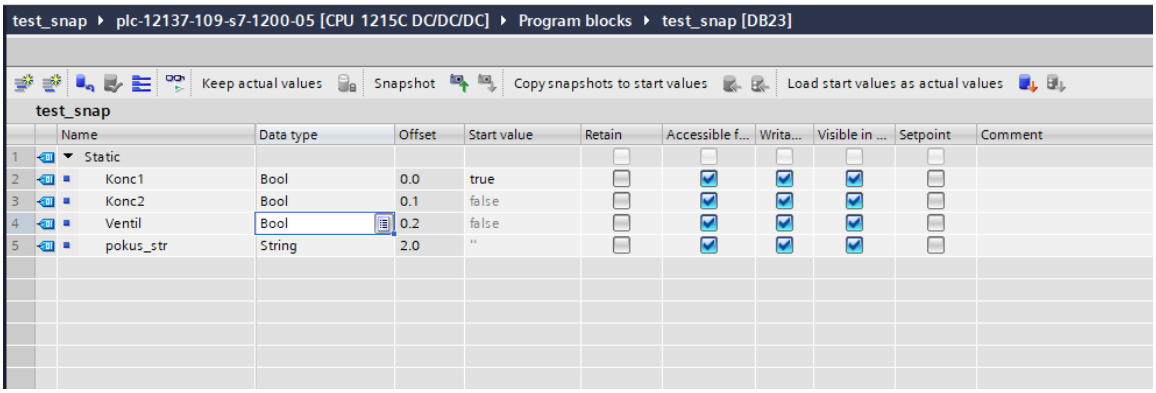


Obrázek 6.6: Povolení přístupu vzdáleným zařízením

Systém CAFDIS implementuje čtení proměnných z PLC. Data, která má systém přečíst, je vhodné umístit do samostatné databáze. Obrázek 6.7 zachycuje nastavení bloku databáze s číslem 23 v programu SIMATIC STEP 7. Důležitý je sloupec *offset*, který udává relativní adresu dané proměnné vzhledem k počátku datového bloku. Tato adresa je jediný způsob identifikace proměnné na straně systému CAFDIS. Tím končí potřebné úpravy programu PLC.

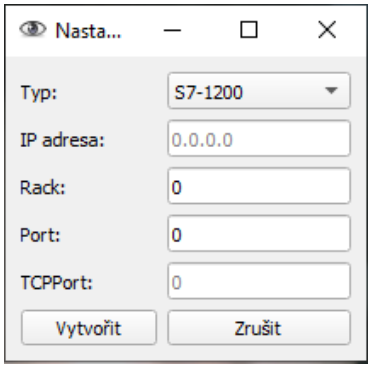
Knihovna Snap7 je rozsáhlá, leč minimálně dokumentovaná. Proto jsou metody této třídy ve zdrojovém kódu CAFDIS doplněné o další vysvětlení a poznámky. Pro lepší pochopení knihovny Snap7 lze kromě oficiální dokumentace doporučit zdroj [51].

Pro navázání komunikace s PLC je potřeba do CAFDIS zadat několik dalších parametrů. Nejdůležitějšími jsou lokální IP adresa PLC a port. Funkce `connect` objektu třídy `snap7.client.Client`, který zastupuje PLC, umožňuje zadat ještě další parametry: Rack



Obrázek 6.7: Databáze proměnných připravených ke čtení

a TCPPort. Všechny zmíněné proměnné lze nastavit prostřednictvím GUI, jak ukazuje obrázek 6.8.



Obrázek 6.8: Databáze proměnných připravených ke čtení

Kliknutím na tlačítko *vytvořit* se systém pokusí navázat komunikaci s nastavenými parametry. Pokud vše proběhne úspěšně, vytvoří se v levé části GUI tabulkový editor. V editoru lze specifikovat proměnné, které má CAFDIS číst. Pro čtení je nutné znát umístění datového bloku, tedy například DB23 z obrázku 6.7, dále je potřeba u každé proměnné přesně definovat její adresu a typ, kterým se mají data na dané adrese interpretovat.

V posledním sloupci tabulky je zobrazena aktuální hodnota proměnné. Aby nedošlo k zahlcení PLC dotazy na čtení, je perioda aktualizace čtených proměnných nastavena na 50 ms. Tuto hodnotu nelze změnit skrze GUI, v případě potřeby lze manuálně změnit ve třídě `PLCWorker` nastavením proměnné `delay`.

Pro přidání kompatibility s PLC dalších značek je nutné napsat novou třídu dědící z třídy `PLC`. V této třídě je potřeba implementovat stejné funkce jako ve třídě `SeimensPLC`, a to především `connect()`, `update()`, `addItem()`, `read()` a `disconnect()`. Referenci na tuto třídu je nutné přidat do třídy `control` do slovníku `PLCTypes`. GUI poté nový typ PLC automaticky zobrazí v rozbalovacím menu *PLC*.

6.4 Přiřazení významu informacím z kamer

Proces 4.0 *Přiřazení významu informacím z kamer* obstarává vytvoření a vyhodnocení veličin, které jsou posléze použity v procesu 5.0 pro diagnostiku. Zatímco proces 1.0 zpracovával obraz a získával data o polohách tagů a oblastí nejčastěji v tabulkové formě, proces 4.0 využívá tato data a rekonstruuje stav zkoumaného procesu, a to především polohy akčních členů a stavy oblastí. Povaha tohoto procesu je opět zcela v rukou uživatele pomocí GUI.

6.4.1 Určení polohy akčních členů

Aktuátory používané ve výrobních procesech jsou rozmanitých druhů a tvarů. Charakter pohybu, který vykonávají, může být velice jednoduchý a snadno popsatelný. Jednoúčelové manipulátory obvykle vykonávají jen posuvný nebo rotační pohyb. Detekce a identifikace takového pohybu není složitá. Oproti tomu robotická ramena, víceúčelové manipulátory nebo stroje s více stupni volnosti vykonávají pohyby poměrně složité. Z informací o polohách tagů jsou složené pohyby aktuátorů hůře identifikovatelné, neboť se nemusí jednat o pohyby rovinné, nýbrž o pohyby v prostoru.

Na definování polohy složitějších aktuátorů je proto potřeba zavést souřadný systém spojený s pohyblivou částí. Souřadný systém plně definuje umístnění v prostoru a může sloužit pro vyhodnocení přípustné nebo chybové polohy aktuátoru. Jelikož je jedním z požadavků na systém odolnost proti posunutí kamery, a navíc je možné, že statická část vyšetřovaného aktuátoru bude připevněna na pohyblivý člen aktuátoru jiného, je na statickou část aktuátoru umístěn rovněž souřadný systém sloužící jako referenční.

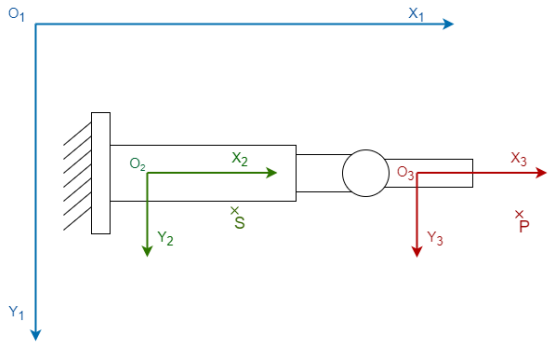
Souřadné systémy lze vytvořit pomocí skupiny tagů umístěných na statickou, resp. pohyblivou část aktuátoru. Pokud je pro daný souřadný systém dostačující znát jen jeho polohu, je nutné použít alespoň 1 tag. Pomocí 2 tagů lze určit natočení kolem osy kolmé na rovinu snímku. Vykonává-li aktuátor složitější pohyb, je nutné použít 3 tagy, kterými lze navíc určit i rotaci kolem zbylých os. Pokud je sledovaný objekt blízko kamery a perspektiva hraje nezanedbatelnou roli, je vhodné použít tagy 4.

Popis polohy zkoumaných souřadnicových systému je reprezentován pomocí transformačních matic.

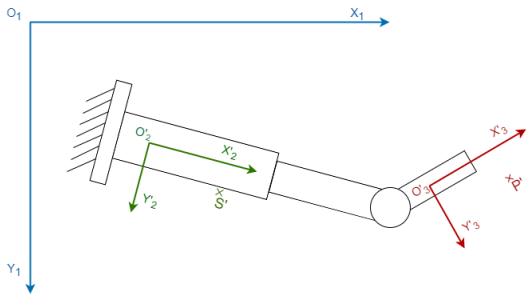
Transformační matice aktuátoru

Knihovna OpenCV disponuje nástroji, kterými lze získat transformační matice pro transformaci v rovině. I z toho důvodu jsou souřadné systémy vyhodnocovány jako rovinné v rovině snímku kamery.

Schéma aktuátoru se souřadnými systémy je zachyceno na obrázcích 6.9 a 6.10. Obrázek 6.9 představuje výchozí polohu aktuátoru, ve které byly pro statickou a pohyblivou část definovány referenční souřadné systémy (x_2, y_2), resp. (x_3, y_3). Při změně polohy kamery nebo aktuátoru se poloha těchto systémů změní. Souřadné systémy v obecné poloze jsou označeny (x'_2, y'_2), resp. (x'_3, y'_3).



Obrázek 6.9: Výchozí poloha aktuátoru při nastavení systému



Obrázek 6.10: Obecná poloha aktuátoru v průběhu měření

Pro každý bod S spojený se statickou částí a každý bod P spojený s pohyblivou částí potom platí:

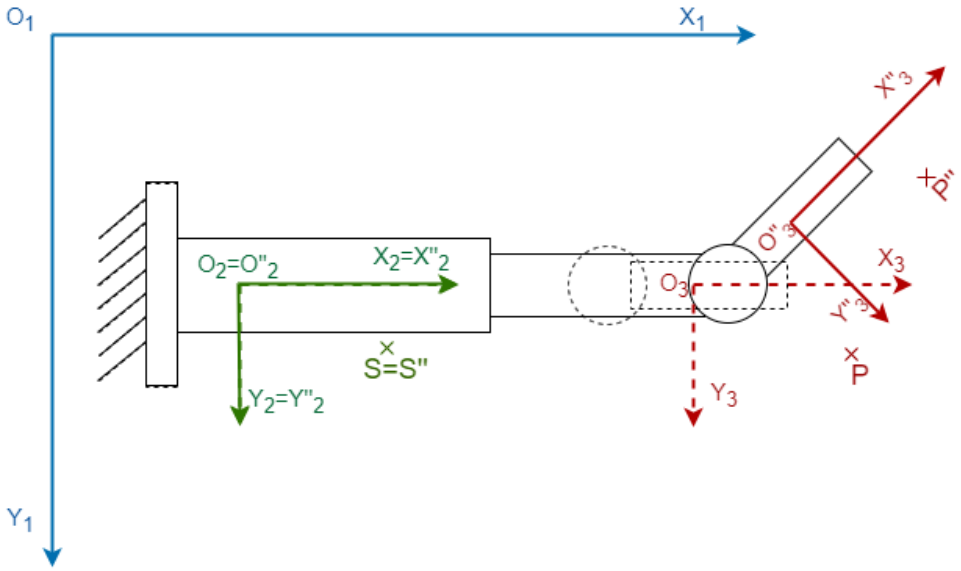
$$\begin{aligned} {}^{2}\mathbf{r}_{2S} &= {}^{2'}\mathbf{r}_{2'S'} \ , \\ {}^{3}\mathbf{r}_{3P} &= {}^{3'}\mathbf{r}_{3'P'} \ . \end{aligned} \tag{6.8}$$

Pomocí funkce `getAffineTransform()` nebo `getPerspectiveTransform()` z knihovny OpenCV a znalosti 3, resp. 4 dvojic odpovídajících si bodů v referenčním a posunutém systému lze získat transformační matice, které splňují rovnice:

$$\begin{aligned} {}^{1}\mathbf{T}_{22'} \cdot {}^{1}\mathbf{r}_{1S} &= {}^{1}\mathbf{r}_{1S'} \ , \\ {}^{1}\mathbf{T}_{33'} \cdot {}^{1}\mathbf{r}_{1P} &= {}^{1}\mathbf{r}_{1P'} \ . \end{aligned} \tag{6.9}$$

Transformační matice ${}^{1}\mathbf{T}_{33'}$ není pro popis pohybu aktuátorů vhodná, protože nezohledňuje možný pohyb statické části, a navíc je vztažena ke globálnímu systému (x_1, y_1). Lepší představu o pohybu aktuátoru lze dostat po zarovnání statických částí aktuátoru ve zkoumané a referenční poloze, tak jak to ukazuje obrázek 6.11. Zarovnání vzniklo transformací celého aktuátoru z obecné polohy do polohy referenční pomocí transformační matice ${}^{1}\mathbf{T}_{22'}$. Jelikož byly transformovány všechny body aktuátoru, lze psát

$${}^{1}\mathbf{T}_{22'} \cdot {}^{1}\mathbf{r}_{1P''} = {}^{1}\mathbf{r}_{1P'} \ . \tag{6.10}$$



Obrázek 6.11: Vyrovnání obecné polohy podle statické části aktuátoru

Žádanou transformační maticí je matice ${}^{1}\mathbf{T}_{33''}$, pro kterou lze sestavit vztah

$$ {}^{1}\mathbf{T}_{33''} \cdot {}^{1}\mathbf{r}_{1P} = {}^{1}\mathbf{r}_{1P''} \ . \tag{6.11} $$

Vyjádřením ${}^{1}\mathbf{r}_{1P''}$ z rovnice 6.10 a dosazením do 6.11 je získáno

$$ {}^{1}\mathbf{T}_{33''} \cdot {}^{1}\mathbf{r}_{1P} = {}^{1}\mathbf{T}_{22'}^{-1} \cdot {}^{1}\mathbf{r}_{1P'} \ . \tag{6.12} $$

Dosazením za ${}^{1}\mathbf{r}_{1P'}$ z rovnice 6.9 se výraz upraví na

$$ {}^{1}\mathbf{T}_{33''} \cdot {}^{1}\mathbf{r}_{1P} = {}^{1}\mathbf{T}_{22'}^{-1} \cdot {}^{1}\mathbf{T}_{33'} \cdot {}^{1}\mathbf{r}_{1P} \ , \tag{6.13} $$

z čehož lze lehce vyjádřit výsledný tvar transformační matice

$$ {}^{1}\mathbf{T}_{33''} = {}^{1}\mathbf{T}_{22'}^{-1} \cdot {}^{1}\mathbf{T}_{33'} \ . \tag{6.14} $$

Zbývá už jen transformovat rovnici ze souřadného systému (x_1, y_1) do vhodnějšího (x_3, y_3). Pro transformaci bodu mezi souřadnými systémy platí:

$$ \begin{aligned} {}^{1}\mathbf{r}_{1P} &= \mathbf{T}_{13} \cdot {}^{3}\mathbf{r}_{3P} \\ {}^{1}\mathbf{r}_{1P''} &= \mathbf{T}_{13} \cdot {}^{3}\mathbf{r}_{3P''} \ . \end{aligned} \tag{6.15} $$

Dosazením vztahu 6.15 do rovnice 6.11 a drobnou úpravou lze získat

$$\mathbf{T}_{13}^{-1} \cdot {}^{1}\mathbf{T}_{33''} \cdot \mathbf{T}_{13} \cdot {}^{3}\mathbf{r}_{3P} = {}^{3}\mathbf{r}_{3P''} \ . \tag{6.16}$$

Z rovnic 6.14 a 6.16 je odvozen finální tvar transformační matice popisující polohu pohyblivého členu aktuátoru:

$${}^{3}\mathbf{T}_{33''} = \mathbf{T}_{13}^{-1} \cdot {}^{1}\mathbf{T}_{22'}^{-1} \cdot {}^{1}\mathbf{T}_{33'} \cdot \mathbf{T}_{13} \ . \tag{6.17}$$

Transformační matice $\mathbf{T}_{13}^{-1}$ slouží zejména pro stanovení středu rotace. V globálním souřadném systému by byl střed rotace dynamické části aktuátoru umístěn do levého horního rohu zkoumaného snímku. Skutečný pohyb aktuátoru by poté byl na základě vzniklé transformační matice obtížně představitelný. Matici $\mathbf{T}_{13}^{-1}$ lze zvolit mnoha způsoby. Pro účely stanovení středu rotace postačí uvažovat translační matici, která umístí střed souřadného systému na statickou část aktuátoru. Souřadnice statické části aktuátoru lze získat například jako souřadnice jednoho z tagů, kterými je statická část označena. Systém CAFDIS používá pro určení souřadnic statické části první ze zadaných tagů. Pokud jsou souřadnice tohoto tagu v globálním souřadném systému x_t, y_t, potom platí, že

$$\mathbf{T}_{13}^{-1} = \begin{pmatrix} 1 & 0 & x_t \\ 0 & 1 & y_t \\ 0 & 0 & 1 \end{pmatrix} \quad . \tag{6.18}$$

Rozbor transformační matice

Nalezením transformační matice mezi aktuální a výchozí polohou aktuátoru problém nekončí. Pro praktické použití je potřeba transformaci vyjádřit prozaičtější formou pochopitelnou i pro nezasvěceného uživatele. Lepší představu o poloze aktuátoru můžeme například získat z koeficientů pro rotaci a translaci. Jelikož je transformační matice ${}^{3}\mathbf{T}_{33''}$ získána pomocí projekcí systému v prostoru do roviny, je nutné kromě zmíněných parametrů uvažovat ještě zvětšení a zkosení pro obě osy x, y. Transformační matice bude mít při neuvažování perspektivy obecně tento tvar:

$${}^{3}\mathbf{T}_{33''} = \begin{pmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{pmatrix} \quad . \tag{6.19}$$

Transformace může mít obecně některé z těchto parametrů:

- rotace,
- posuv ve směru osy x,
- posuv ve směru osy y,
- zvětšení v ose x,
- zvětšení v ose y,
- zkosení osy x k ose y,
- zkosení osy y k ose x.

Jak je vidět, matice obsahuje 6 nezávislých hodnot, ale parametrů transformace je 7. Existuje tedy nekonečný počet možných rozkladů matice na tuto sérii parametrů. Jedním ze způsobů, jak dostat jednoznačné hodnoty parametrů, je neuvažování jednoho z nich. Bylo zvoleno, že zkosení osy y k ose x bude vždy nulové, zbývá tedy 6 parametrů, které jsou jednoznačně určeny transformační maticí.

Koeficienty e, f v matici 6.19 udávají posuv ve směru x, resp. y. Poslední řádek matice nemá pro povahu transformace význam. Zbývá rozklíčovat matici

$$\begin{pmatrix} a & c \\ b & d \end{pmatrix} \quad . \tag{6.20}$$

Po aplikaci QR rozkladu se matice 6.20 rozpadne na

$$\begin{pmatrix} a & c \\ b & d \end{pmatrix} = \begin{pmatrix} a/r & -b/r \\ b/r & a/r \end{pmatrix} \cdot \begin{pmatrix} r & (ac+bd)/r \\ 0 & \Delta/r \end{pmatrix} \quad , \tag{6.21}$$

kde

$$r = a^2 + b^2 \neq 0 \quad , \tag{6.22}$$

$$\Delta = ad - bc \quad . \tag{6.23}$$

Matice $\mathbf{R}$ v rovnici 6.21 je dále rozložena na

$$\begin{pmatrix} r & (ac+bd)/r \\ 0 & \Delta/r \end{pmatrix} = \begin{pmatrix} r & 0 \\ 0 & \Delta/r \end{pmatrix} \cdot \begin{pmatrix} 1 & (ac+bd)/r^2 \\ 0 & 1 \end{pmatrix} \quad . \tag{6.24}$$

Dosazením do 6.21 lze získat výraz

$$\begin{pmatrix} a & c \\ b & d \end{pmatrix} = \begin{pmatrix} a/r & -b/r \\ b/r & a/r \end{pmatrix} \cdot \begin{pmatrix} r & 0 \\ 0 & \Delta/r \end{pmatrix} \cdot \begin{pmatrix} 1 & (ac+bd)/r^2 \\ 0 & 1 \end{pmatrix} \quad , \tag{6.25}$$

který lze zkráceně zapsat jako

$$\begin{pmatrix} a & c \\ b & d \end{pmatrix} = \mathbf{A} \cdot \mathbf{B} \cdot \mathbf{C} \quad . \tag{6.26}$$

Matice **A** je ortogonální matice. Její prvky jsou vždy menší než 1 ($r = a^2 + b^2$). Lze ji tudíž přepsat do známého tvaru

$$\begin{pmatrix} a/r & -b/r \\ b/r & a/r \end{pmatrix} = \begin{pmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{pmatrix} \quad , \tag{6.27}$$

ze kterého lze lehce vyjádřit úhel rotace transformační matice **A** jako

$$\phi = arccos(a/r) \cdot sign(b) \quad . \tag{6.28}$$

Matice **B** nemůže být matice rotace, protože prvky na hlavní diagonále mají rozdílnou velikost. Po transformaci libovolného vektoru touto maticí se složka osy x zvětší r-krát a složka osy y je zvětšena Δ/r-krát. Jedná se tedy o matici změny měřítka. O zvětšeních M lze prohlásit, že

$$M_x = r = a^2 + b^2 \quad , \tag{6.29}$$

$$M_y = \Delta/r = \frac{ad - bc}{a^2 + b^2} \quad . \tag{6.30}$$

Poslední matice **C** není maticí rotace, protože nerotuje osy x, y o stejný úhel. Souřadnice y zůstává po transformaci maticí **C** nezměněna, zatímco souřadnice x je posunuta přímo úměrně velikosti souřadnice y a koeficientu $(ac + bd)/r^2$. Jedná se tudíž o zkosení osy y, nová směrnice osy y je právě $(ac + bd)/r^2$ vůči ose x. Úhel zkosení γ lze tedy vyjádřit rovnicí

$$\gamma = arctan\left(\frac{ac + bd}{r^2}\right) \quad . \tag{6.31}$$

Pokud platí $a = 0 \wedge b = 0$, tedy $r = 0$, nelze tento zápis použít a je nutné rozklad pozměnit na

$$\begin{pmatrix} a & c \\ b & d \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} -c/s & d/s \\ -d/s & -c/s \end{pmatrix} \cdot \begin{pmatrix} \Delta/s & 0 \\ 0 & s \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ (ac + bd)/s^2 & 1 \end{pmatrix} \quad , \tag{6.32}$$

kde

$$s = c^2 + d^2 \neq 0 \quad , \tag{6.33}$$

$$\Delta = ad - bc \quad . \tag{6.34}$$

Zkráceně lze zápis rovnice 6.32 napsat jako

$$\begin{pmatrix} a & c \\ b & d \end{pmatrix} = \mathbf{R}(\pi/2) \cdot \mathbf{A'} \cdot \mathbf{B'} \cdot \mathbf{C'} \quad . \tag{6.35}$$

Matice $\mathbf{R}(\pi/2)$ je ortogonální matice, kterou je možné zapsat jako transformační matici rotace pro úhel $\pi/2$:

$$\mathbf{R}(\pi/2) = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} cos(\pi/2) & -sin(\pi/2) \\ sin(\pi/2 & cos(\pi/2) \end{pmatrix} \quad . \tag{6.36}$$

Matice **A'** je obdobně jako matice **A** taktéž maticí rotace. Platí, že

$$\begin{pmatrix} -c/s & d/s \\ -d/s & -c/s \end{pmatrix} = \begin{pmatrix} cos(\psi) & -sin(\psi) \\ sin(\psi) & cos(\psi) \end{pmatrix} \quad . \tag{6.37}$$

Potom úhel rotace ψ lze vyjádřit jako

$$\psi = arccos(-c/r) \cdot sign(-d) \quad . \tag{6.38}$$

Výsledná rotace po aplikaci obou transformačních matic $\mathbf{R}(\pi/2)$ a **A'** je

$$\phi' = \pi/2 + \psi = \pi/2 + arccos(-c/r) \cdot sign(-d) \quad . \tag{6.39}$$

Z matice **B'** lze získat zvětšení pro osy x a y ve tvaru

$$M'_x = \Delta/s = \frac{ad - bc}{c^2 + d^2} \quad , \tag{6.40}$$

$$M'_y = s = c^2 + d^2 \quad . \tag{6.41}$$

Matice **C'** tentokrát udává zkosení osy x, které je dáno rovnicí

$$\gamma = arctan\left(\frac{ac + bd}{s^2}\right) = arctan\left(\frac{ac + bd}{(c^2 + d^2)^2}\right) \quad . \tag{6.42}$$

Jednotlivé výsledky dekompozice transformační matice jsou shrnuty v tabulce 6.1.

Kromě již zmíněné dekompozice na základní pohyby umožňuje systém CAFDIS skrz GUI přejmenovat vytvořený virtuální prvek, přidávat a odebírat referenční tagy pohyblivé a statické části a měnit virtuální model pohyblivého prvku. GUI také poskytuje informace o času poslední úspěšné detekce a hodnotách všech zkoumaných veličin.

Současná verze CAFDIS implementuje pouze obecný model pohyblivých prvků podle výše zmíněných rovnic. Tento model se dobře hodí pro pneumatické aktuátory s několika

Parametr	$a \neq 0 \vee b \neq 0$	$c \neq 0 \vee d \neq 0$
Rotace	$arccos(a/r) \cdot sign(b)$	$\pi/2 + arccos(-c/r) \cdot sign(-d)$
Posuv ve směru osy x	e	e
Posuv ve směru osy y	f	f
Zvětšení v ose x	$a^2 + b^2$	$\frac{ad-bc}{c^2+d^2}$
Zvětšení v ose y	$\frac{ad-bc}{a^2+b^2}$	$c^2 + d^2$
Zkosení	$arctan\left(\frac{ac+bd}{(a^2+b^2)^2}\right)$	$arctan\left(\frac{ac+bd}{(c^2+d^2)^2}\right)$

Tabulka 6.1: Souhrn vzorců pro dekompozici transformační matice

definovanými polohami. Model aktuátoru neumí pracovat s perspektivou, proto je zbytečné označovat statickou nebo pohyblivou část více než 3 tagy. Systém je navržen tak, aby implementace dalších modelů byla co nejjednodušší. Stačí vytvořit novou třídu, která bude dědit z třídy `Element`, a přidat tuto třídu do seznamu `elementTypes` ve třídě `Camera`. Pro podrobnější pokyny a informace o fungování elementů lze nahlédnout do okomentovaného zdrojového kódu.

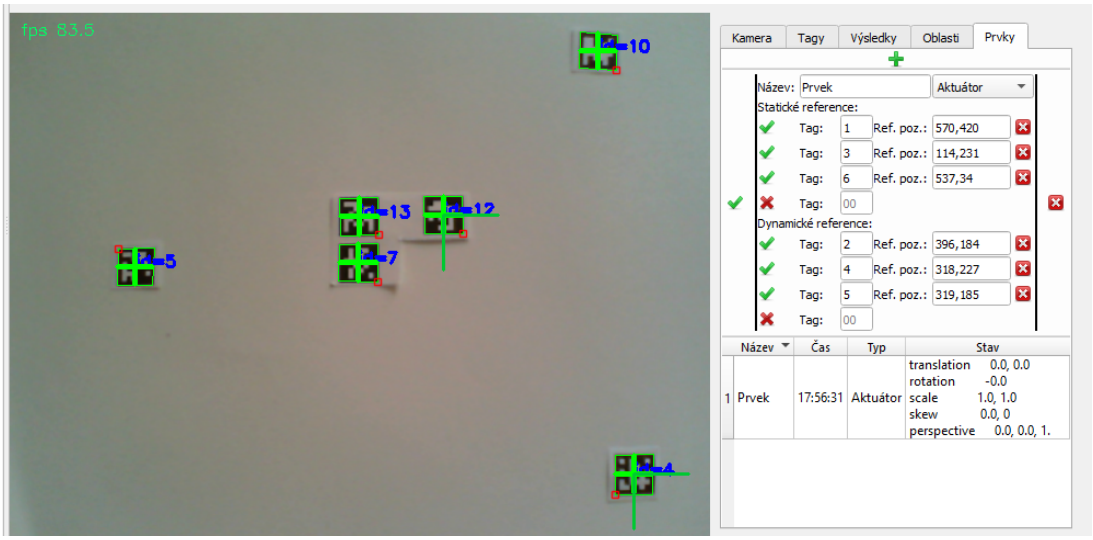
Další modely by mohly lépe popisovat pohyby rotačních motorů, dopravníků a dalších zařízení s nepřetržitým pohybem. Mezi veličiny získané z pohybu akčního členu by mohla patřit rychlost pohybu, nebo i zrychlení.

Demonstrace určení polohy

Celý algoritmus byl otestován na sérii tagů. Pro označení statické a pohyblivé části pomyslného aktuátoru byly použity 3 tagy. Systém je ve své výchozí poloze zachycen na obrázku 6.12.

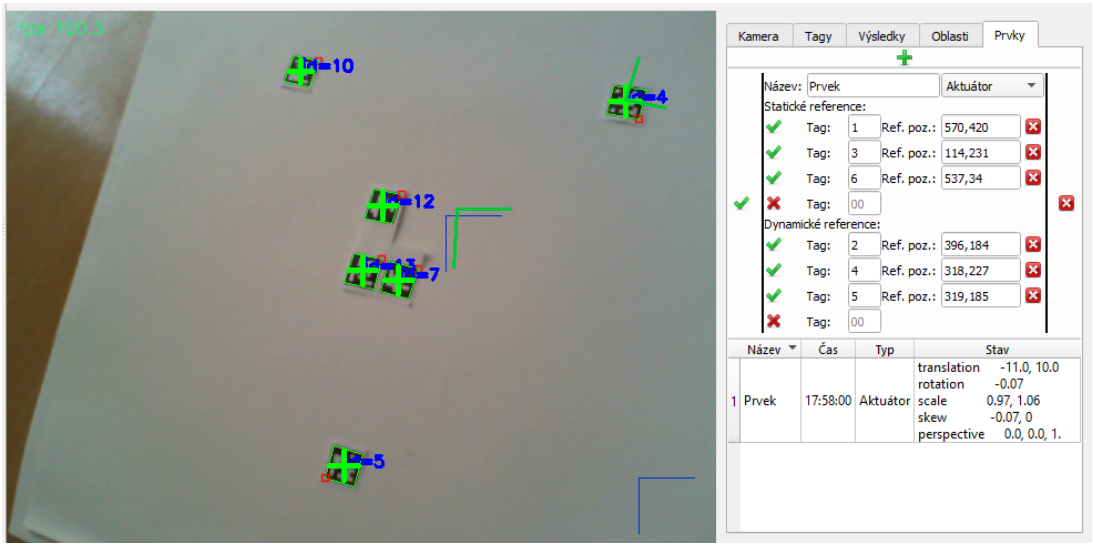
Tagy s modře označeným identifikačním číslem 5, 10 a 4 jsou použity jako statické reference, kterým odpovídají čísla 1, 3 a 6 v tabulce nalezených tagů. Tagy s ID 7, 12 a 13 představují pohyblivou část aktuátoru. V pravé části obrázku je zobrazen panel s nastavením aktuátoru a výsledky určení polohy. Jelikož se jedná o výchozí polohu, jsou všechny parametry polohy aktuátoru, kromě zvětšení, nulové.

Na obrázku 6.13 je stejné rozmístění tagů zachyceno z jiného úhlu a vzdálenosti. Výsledky dekompozice by měly být totožné s výsledky z výchozí polohy. Je vidět, že pro účely CAFDIS se výsledky shodují dostatečně přesně. Drobné odchylky mohou být způsobeny nepřesnou kalibrací kamery nebo zanedbáním vlivů perspektivy. Dvě modré a zelené úsečky poblíž pohyblivé části pomyslného aktuátoru představují osy souřadného systému v referenční, resp. aktuální poloze. Osy zeleného souřadného systému jsou získány



Obrázek 6.12: Výchozí poloha tagů pro demonstraci algoritmu určení polohy aktuátorů

transformací bodů modrého systému pomocí transformační matice ${}^{3}\mathbf{T}_{33''}$. Vzájemné posunutí, zvětšení a natočení těchto dvou útvarů odpovídá zjištěným hodnotám.

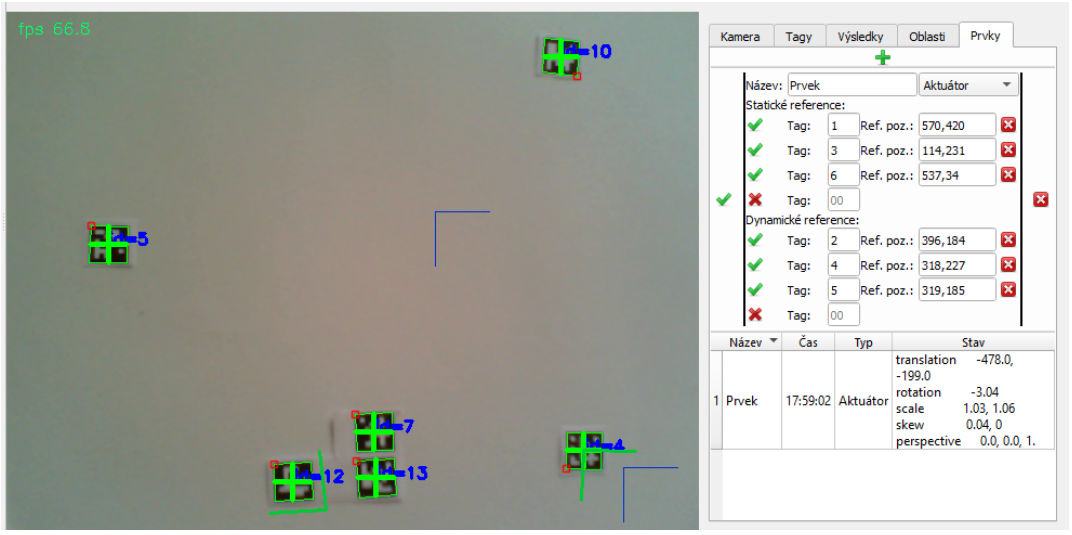


Obrázek 6.13: Výchozí poloha tagů po posunutí kamery

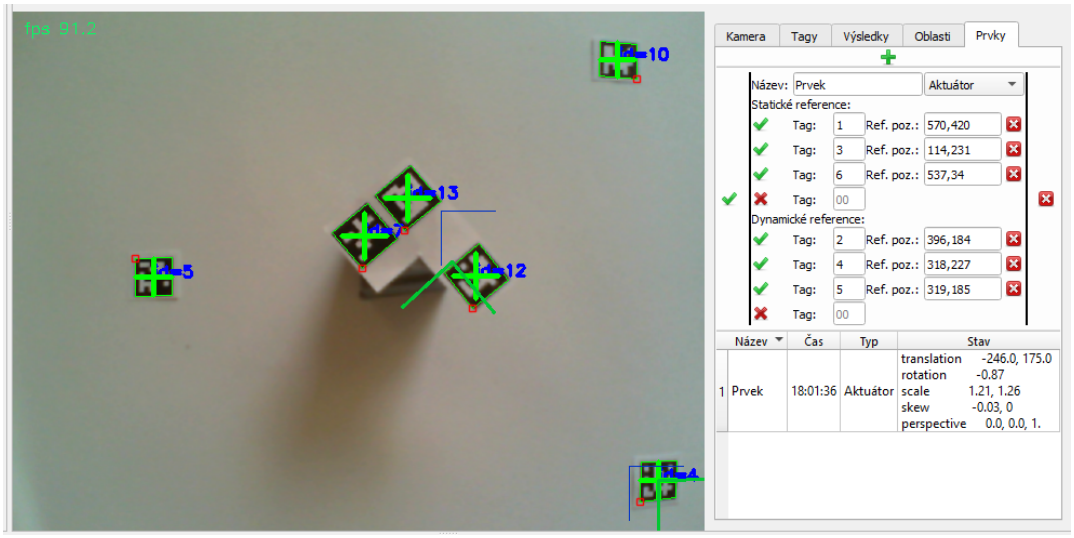
Na obrázcích 6.14 je zobrazena detekce pomyslného aktuátoru a určení jeho polohy při posunutí tagů označujících pohyblivou část. V první poloze na 6.14a je pohyblivá část otočena a posunuta, čemuž odpovídá i výsledek určení polohy (CAFDIS udává rotaci v radiánech).

Na obrázku 6.14b je pohyblivá část posunuta směrem ke kameře. Hodnoty koeficientů zvětšení vzrostly patřičným způsobem.

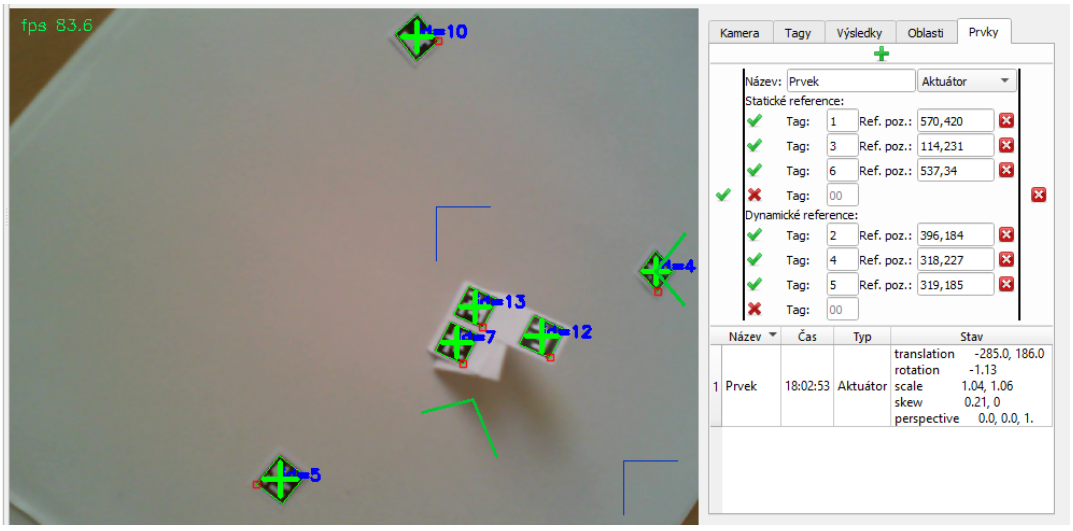
Poslední obrázek 6.14c zachycuje naklonění pohyblivé části z roviny snímku. Tomuto naklonění odpovídá koeficient zkosení.



(a) Poloha 1



(b) Poloha 2



(c) Poloha 3

Obrázek 6.14: Nová poloha tagů pro demonstraci algoritmu určení polohy aktuátorů

6.4.2 Určení stavu oblastí

Procesem 1.3 byl získán výřez snímku zachycující zkoumanou oblast. Dále je potřeba vyhodnotit barvu oblasti a poté i její stav.

Určení barvy

Pro vyhodnocení barvy je nejprve barevný prostor oblasti převeden do prostoru HSV z důvodů uvedených v kapitole 3.5. Převod je realizován funkcí `cvtColor()` z knihovny OpenCV. Vzniklý obraz lze popsat funkcí

$$f = f(x, y, k) \quad , \tag{6.43}$$

kde x, y jsou souřadnice pixelu v obraze a k je kanál z HSV prostoru. Výsledná barva (H_v, S_v, V_v) oblasti jako celku je získána jednou z následujících operací:

- Podle principu maxima

$$\begin{aligned} H_v &= \max_{x,y}(f(x, y, H)) \\ S_v &= \max_{x,y}(f(x, y, S)) \\ V_v &= \max_{x,y}(f(x, y, V)) \end{aligned} \tag{6.44}$$

- Podle principu minima

$$\begin{aligned} H_v &= \min_{x,y}(f(x, y, H)) \\ S_v &= \min_{x,y}(f(x, y, S)) \\ V_v &= \min_{x,y}(f(x, y, V)) \end{aligned} \tag{6.45}$$

- Jako aritmetický průměr

$$\begin{aligned} H_v &= \sum_{x=1}^{N}\sum_{y=1}^{M}\frac{1}{N*M}(f(x, y, H)) \\ S_v &= \sum_{x=1}^{N}\sum_{y=1}^{M}\frac{1}{N*M}(f(x, y, S)) \\ V_v &= \sum_{x=1}^{N}\sum_{y=1}^{M}\frac{1}{N*M}(f(x, y, V)) \end{aligned} \tag{6.46}$$

- Jako medián

$$\begin{aligned} H_v &= \underset{x,y}{\mathbf{Me}}(f(x,y,H)) \\ S_v &= \underset{x,y}{\mathbf{Me}}(f(x,y,S)) \\ V_v &= \underset{x,y}{\mathbf{Me}}(f(x,y,V)) \end{aligned} \tag{6.47}$$

To, který postup bude pro danou oblast použit, nastavuje uživatel prostřednictvím GUI. Ukázalo se, že obecně nejlepší metodou je medián, který není citlivý na barevné změny na ploše oblasti vzniklé vniknutím malého cizího objektu nebo nepřesným umístěním oblasti na zkoumaný objekt. Pokud většina plochy oblasti obsahuje hledanou barvu, medián ji detekuje poměrně dobře.

Princip maxima je velice užitečný při hledání světlé barvy na tmavém pozadí. V takovém případě, pokud je alespoň jeden pixel hledané světlé barvy uvnitř oblasti, bude pomocí principu maxima detekován. Díky tomu nemusí oblast přesně kopírovat obrys hledaného objektu, stačí, když bude světlý hledaný objekt uvnitř jinak tmavé oblasti. Z toho důvodu je tato metoda dobře využitelná pro detekci malých LED na tmavých senzorech nebo tmavých PLC.

Princip minima je z podobných důvodů vhodný pro detekci tmavých objektů na jinak světlém pozadí.

Aritmetický průměr nemá zatím v systému CAFDIS přílišné využití. Může být vhodný pro detekci barev u oblastí vyhodnocujících více barevných stavů, nynější verze systému ovšem dokáže pracovat jen se 2 stavy každé oblasti.

Implementace další metody vyhodnocení barev je velice jednoduchá. Stačí do třídy `Region` přidat funkci obstarávající vyhodnocení barvy a její referenci umístit do pole `methodsFunctions` ve stejné třídě. Podrobnější informace lze vyčíst z komentářů ve zdrojovém kódu.

Určení stavu

Dalším krokem vyhodnocení oblastí je určení příslušnosti dané barvy k jednomu z definovaných stavů. Stavy definuje uživatel prostřednictvím GUI. Systém CAFDIS umožňuje definovat pro každou oblast 2 barvy odpovídající stavům *zapnuto*, resp. *vypnuto*. Barvy lze vybrat z barevné palety nebo zadat jako RGB kód v desítkové nebo hexadecimální soustavě.

Systém obsahuje kritérium podobnosti barev v HSV prostoru ve tvaru $(\Delta H, \Delta S, \Delta V)$. Výchozí nastavení kritéria je $(15, 100, 100)$. Pokud se detekovaná barva oblasti liší v jednotlivých složkách od barvy definovaného stavu v HSV prostoru maximálně o hodnotu danou kritériem podobnosti, je oblast vyhodnocena jako příslušící danému stavu. Pokud více

stavů splňuje podmínku danou kritériem podobnosti, je oblasti přiřazen stav nejpodobnější, tedy takový stav, jehož definovaná barva má od detekované barvy nejmenší euklidovskou vzdálenost. Pokud je vzdálenost stejná pro více stavů, je pravdivě vyhodnocen stav *zapnuto*.

Pro účely implementovaného dvoustavového modelu je do kódu přidána další podmínka. Pokud nesplňuje detekovaná barva kritérium podobnosti ani pro jeden definovaný stav, je i tak oblasti přiřazen stav nejpodobnější. Kritérium podobnosti tudíž pro dvoustavový model postrádá smysl, lze ji využít pro implementaci vícestavových modelů. Podmínku lze změnit nebo vypnout ve zdrojovém kódu ve třídě `Region` ve funkci `evaluateState()`.

Ve stejné funkci lze implementovat vícestavový model oblasti, proměnnou reprezentující barvu daného stavu je ještě nutné přidat do funkce `__init__()` ve stejné třídě. Na rozdíl od ostatních navrhovaných úprav systému v této práci vyžaduje přidání stavu ještě rozsáhlejší zásah do GUI, a to do třídy `CameraSettingWindow`, konkrétně do funkce `regionsRebuild`, která zobrazuje uživatelské rozhraní vybrané oblasti, do třídy `RegionsTableModel`, kde je nutné přidat další stav do záhlaví zobrazované tabulky v proměnné `header`, a naposledy ve třídě `Region` je nutné aktualizovat funkci `getData()`, která předává informace do GUI.

6.5 Diagnostika

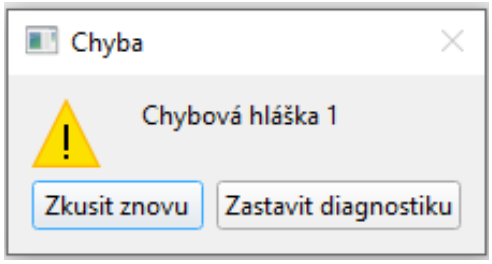
Diagnostika je systémem CAFDIS vyhodnocována pomocí pravdivostní tabulky popisující všechny myslitelné chybové stavy. Jelikož může být každý monitorovaný proces zcela odlišný, celou pravdivostní tabulku vytváří uživatel pomocí GUI. Obecně lze použitou pravdivostní tabulku vyjádřit zápisem podle tabulky 6.2.

Případ 1	Událost 11	Událost 12	Událost 13	...	Chybová hláška 1
Případ 2	Událost 21	Událost 22	Událost 23	...	Chybová hláška 2
Případ 3	Událost 31	Událost 32	Událost 33	...	Chybová hláška 3

Tabulka 6.2: Ukázka pravdivostní tabulky použité pro diagnostický modul

Jednotlivé řádky představují případy (v programu vedeno pod třídou s označením `Case`). Každý případ reprezentuje určitý chybový stav, který po své aktivaci vyvolá odpovídající chybovou hlášku. Jednotlivé případy se skládají z událostí. Případ je vyhodnocen jako aktivní, pokud jsou všechny události daného případu vyhodnoceny jako aktivní. Při vzniku více chyb současně se chyby zobrazují v pořadí, v jakém jsou uvedeny v pravdivostní tabulce. Další chyba je zobrazena teprve po reakci na chybu předešlou. Při vzniku chyby je diagnostika automaticky pozastavena a uživatel je upozorněn zvukovým signálem (podle nativního zvukového efektu chybových hlášek použitého OS) a vyskakovacím oknem, které je ve svém výchozím nastavení zobrazeno na obrázku 6.15.

Uživatel má možnost tlačítkem *zkusit znovu* opět spustit diagnostiku a zavřít chybovou



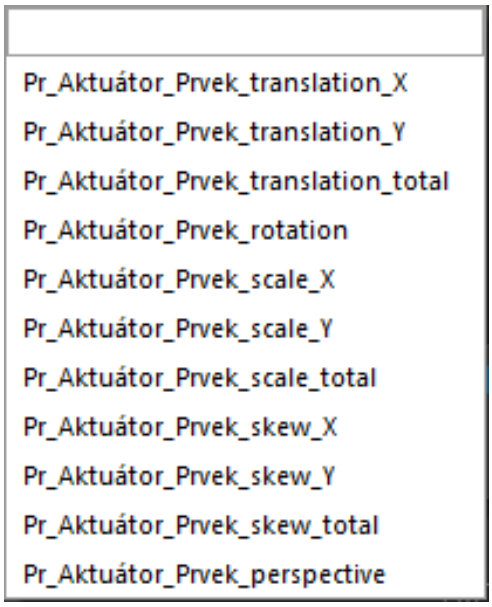
Obrázek 6.15: Ukázka vyskakovacího okna s chybovou hláškou

hlášku. Pokud je chyba odstraněna, systém CAFDIS dále běží se stejným nastavením jako před vznikem chyby. Pokud chyba přetrvává, okno s chybovou hláškou se ihned objeví znovu.

Tlačítkem *zastavit diagnostiku* lze zavřít vyskakovací okno bez opětovného spuštění procesu diagnostiky. Nehrozí proto další zobrazení chybových hlášek. Uživatel má poté možnost přezkoumat nastavení systému a stavy proměnných zodpovědných za vyvolání chyby nebo chybový případ odstranit či upravit.

Jednotlivé veličiny zkoumaného procesu lze přiřadit událostem pravdivostní tabulky pomocí automaticky generovaného a aktualizovaného seznamu všech dostupných informací o zkoumaném procesu. Událost je vyhodnocena jako pravdivá, pokud přiřazená veličina splňuje podmínky zadané uživatelem. Veličinami mohou být stavy jednotlivých oblastí, stavy proměnných PLC nebo veličiny pohyblivých prvků popsané v tabulce 6.1. Seznam veličin je k dispozici pod každou událostí v GUI pod rozbalovací nabídkou.

Ukázka seznamu veličin jednoho aktuátoru s názvem *Prvek* je zachycena na obrázku 6.16.



Obrázek 6.16: Seznam měřitelných veličin jednoho aktuátoru

Uživatel může zvolené veličině v události zadat jednu a více podmínek ze seznamu:

- rovno zadané hodnotě,
- nerovno zadané hodnotě,
- větší než zadaná hodnota,
- menší než zadaná hodnota.

Jelikož není zaručena synchronnost detekce všech potřebných elementů pro vyhodnocení události, je možné zadat i *tolerované trvání*. Po tuto dobu bude pravdivě vyhodnocená událost stále považována za nepravdivou. Tím je možné například potlačit vliv špatné detekce jednoho snímku za předpokladu, že správně vyhodnocený snímek bude detekován nejpozději do doby dané hodnotou *tolerované trvání*.

Každému případu, tedy řádku pravdivostní tabulky, lze prostřednictvím GUI nastavit název chyby, chybovou hlášku a hodnotu *tolerované trvání*, která má podobný význam jako u událostí. Pravdivě vyhodnocený případ, tedy případ s pravdivě vyhodnocenými všemi událostmi, bude po dobu danou hodnotou *tolerované trvání* ještě považován za nepravdivý.

Uživatel má možnost, kromě již zmíněného nastavení a přidávání dalších případů a událostí, kdykoliv pozastavit a zase spustit diagnostický proces, díky čemuž lze předejít nechtěným vyvoláním chyb zvláště během nastavování celé diagnostiky.

Diagnostika systému CAFDIS nemá vnitřní paměť stavů zkoumaného procesu a nepodporuje tudíž sekvenční logiku. Z tohoto důvodu není vhodná pro monitorování všech druhů chyb. Díky robustnosti diagnostického modulu nebude problém možnosti diagnostiky rozšířit v některé z dalších verzí CAFDIS. Podrobnější informace o fungování diagnostiky a o možnostech dalšího rozšíření lze vyčíst z komentářů ve zdrojovém kódu ve třídách `Diagnosis`, `Case`, ve třídě obstarávající grafické zobrazení `DiagnosisWindow` a třídě obstarávající správu multithreadingu `DiagnosisWorker`.

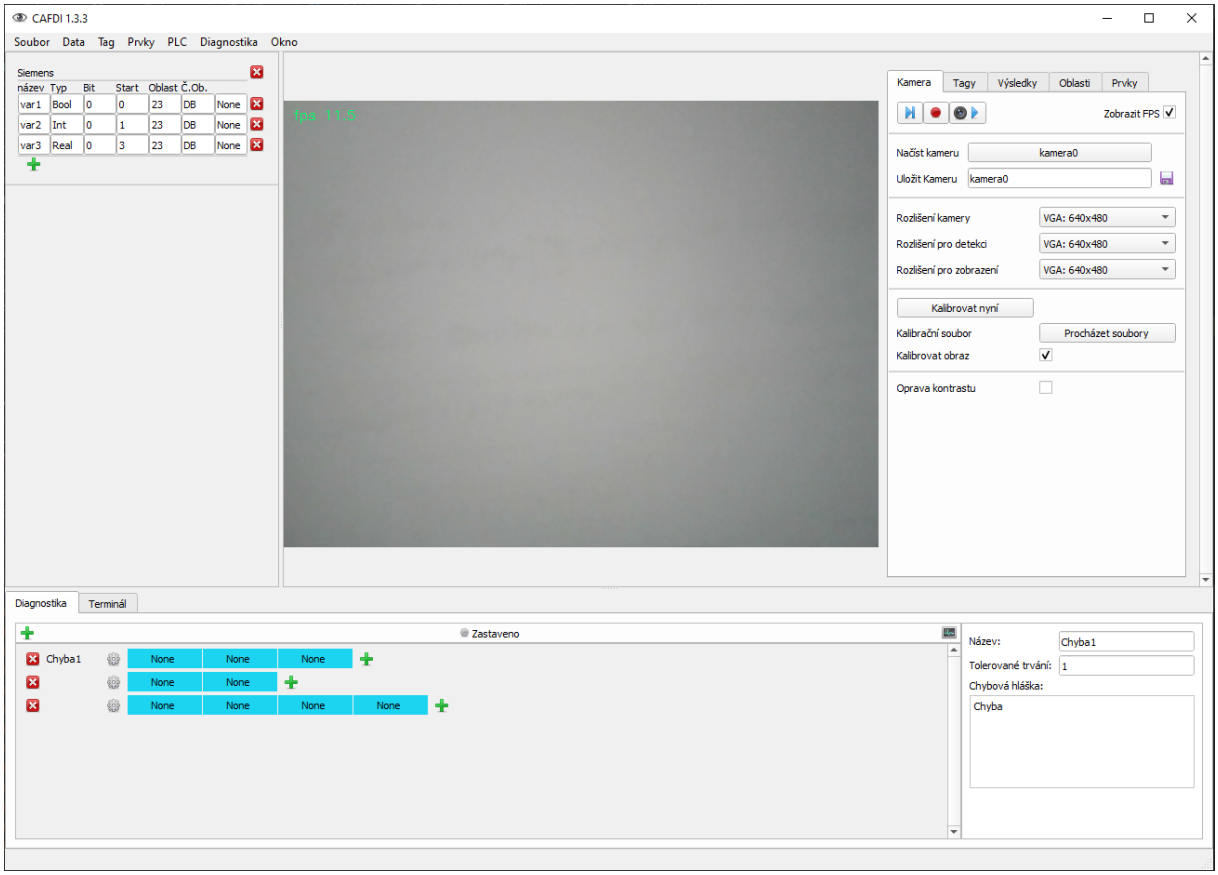
6.6 Grafické uživatelské rozhraní (GUI)

Pro naprogramování GUI byla použita knihovna PyQT5 [44]. Tato knihovna umožňuje pokročilou správu multithreadingu a vytváření tzv. signálů pro synchronizaci komunikace mezi jednotlivými vlákny.

Základní vzhled aplikace je možné vidět na obrázku 6.17.

6.6.1 Základní popis

V levé části je sekce pro nastavení čtení dat z PLC. Přidání nového PLC se provádí pomocí rozbalovací nabídky v horní části okna.



Obrázek 6.17: Grafické uživatelské rozhraní

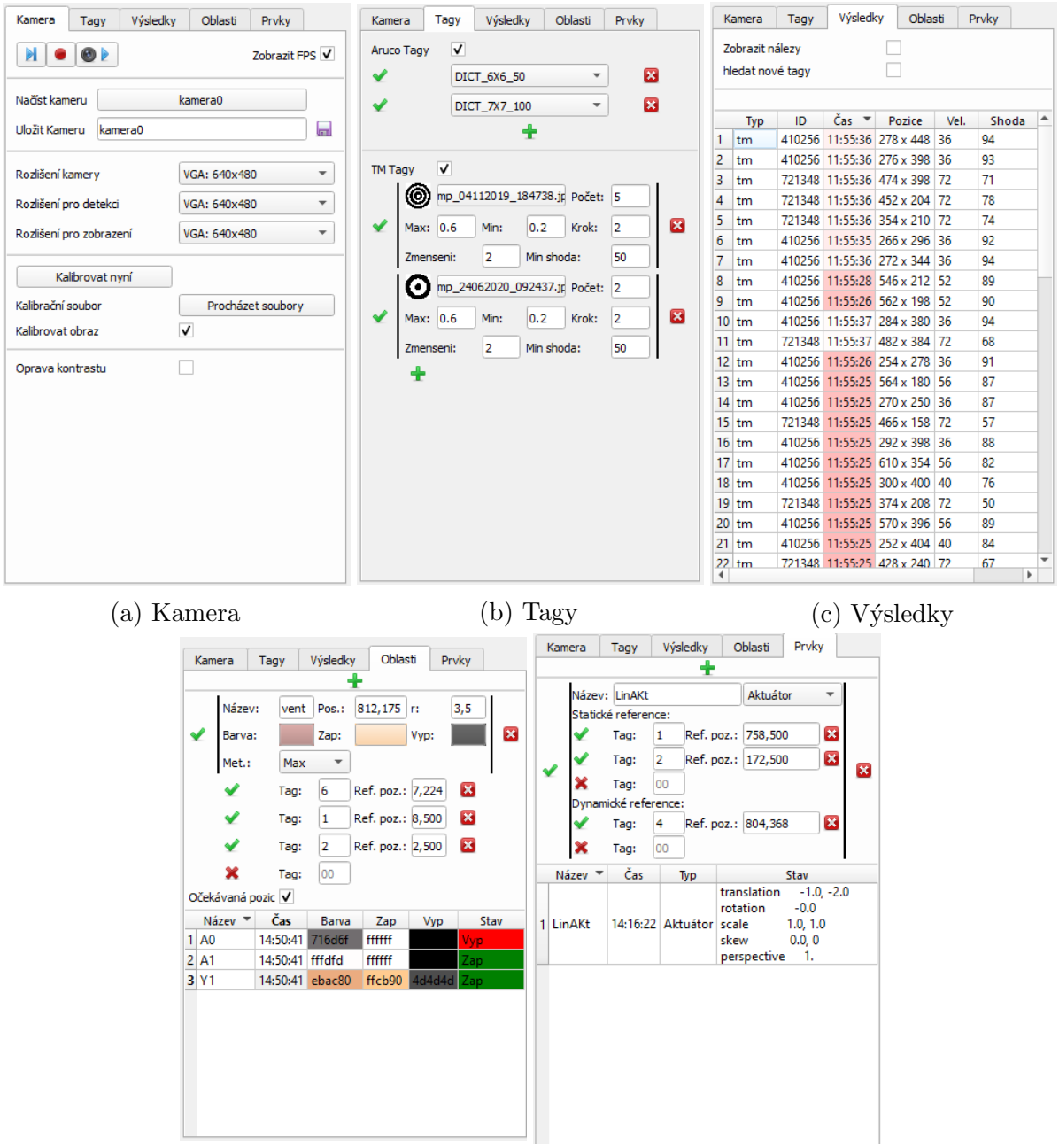
Dolní část okna je věnována diagnostice a terminálu, do kterého je přesměrován standardní i chybový vystup programu. Editace diagnostiky je intuitivní. Nový řádek je možné vložit po klepnutí na zelené plus přímo pod záložkou diagnostika. Jednotlivé buňky se do řádků vkládají kliknutím na zelené plus na konci daného řádku. Po klepnutí na modrou buňku lze daný element editovat pomocí nabídky v pravé části sekce diagnostiky. Tlačítko s ikonou ozubeného kola vyvolá nabídku nastavení pro celý řádek, které je zobrazeno taktéž v pravé části spodní sekce.

Diagnostiku je možné zastavovat a znovu spouštět tlačítkem s ikonou monitoru vpravo od popisu stavu diagnostiky (na obrázku 6.17 nápis *zastaveno*).

Zbytek okna tvoří sekce pro nastavení a zobrazení obrazů z kamer. Jednotlivé kamery jsou v GUI řazeny pod sebou. Každé kameře je přiřazen jeden panel s nastavením, který lze vidět v pravé horní části okna.

6.6.2 Nastavení zpracování obrazu

Panel nastavení kamery obsahuje několik záložek. Při nastavování systému CAFDIS na konkrétní monitorovaný proces je vhodné procházet tyto záložky postupně zleva doprava.



(a) Kamera (b) Tagy (c) Výsledky

(d) Oblasti (e) Prvky

Obrázek 6.18: Panel nastavení kamery

Kamera

Záložka 6.18a obsahuje obecné nastavení kamery. Tlačítkem s ikonou objektivu v horní části záložky je možné pozastavit snímání obrazu, ale detekce štítků a další procesy spojené s vyhodnocováním snímku běží dále. Tento režim je vhodný pro optimalizaci nastavení parametrů detekce nebo pro definování referenčních tagů pro oblasti a prvky, neboť je zaručeno, že se poloha štítků během nestavování nezmění.

Levým tlačítkem z trojice ovládacích tlačítek lze zastavit nejen snímání kamery, ale i celé vlákno obsluhující danou kameru. Tlačítko nahrávání je neaktivní, tato funkcionalita

zatím nebyla implementována.

Dále je možné celé nastavení kamery uložit nebo načíst. Doporučená varianta ovšem spočívá v ukládání a načítání konfigurace celé aplikace přes rozbalovací menu *Soubor*. Díky tomu je možné se vyhnout případným kolizím v programu týkajícím se sdílených proměnných mezi jednotlivými moduly systému.

Význam dalších nastavení v záložce kamera je zřejmý a nepotřebuje komentář.

Tagy

Záložka 6.18b umožňuje uživateli definovat hledané tagy a nastavit parametry detekčních algoritmů.

První část záložky je věnována ArUco tagům. Uživatel může zvolit z kompletního seznamu sad tagů implementovaných do knihovny ArUco.

V druhé části lze nastavovat detekci tagů hledaných pomocí TM metody. Systém CAFDIS v této verzi umožňuje detekci pouze rotačně invariantních tagů. Každá sekce odpovídá jednomu hledanému vzoru, který lze vybrat ze souboru na disku ve standardních obrázkových formátech (jpg, jpeg, png...). Pro každý vzor je možné nastavit parametry vysvětlené v kapitole 6.2.2.

Výsledky

Tato záložka obsahuje důležité zaškrtávací políčko *hledat nové tagy*, které má značný vliv na algoritmus přiřazení nálezů existujícím tagům, jak je popsáno v kapitole 6.2.3.

Dále záložka nabízí jen přehled nalezených tagů s možností seřadit data podle každého ze sloupců. Pole s časy detekce mění barvu podle stáří poslední úspěšné detekce. Tím je možné uživatele lépe upozornit na chybu v nastavení systému.

Po kliknutí na záznam z tabulky levým tlačítkem myši je příslušný tag označen křížem na snímku z kamery. Zelený kříž znamená, že se na daném snímku podařilo tag detekovat, červený kříž udává poslední známou polohu daného tagu.

Klik pravým tlačítkem vyvolá kontextové menu, kde je možné záznam odstranit nebo zrušit jeho označování na snímku.

Oblasti

V horní části záložky 6.18d lze kliknutím na zelené plus vytvořit novou oblast. Její poloha a rozměry jsou definovány uživatelem tažením myši na snímku v novém okně. Více podrobností se lze dočíst v kapitole 6.2.4.

Všechny vytvořené oblasti jsou zobrazeny v tabulce ve spodní části záložky. Tabulku je možné opět řadit podle všech sloupců. Jednotlivé záznamy jsou pro větší přehlednost

barevně formátovány. Po kliknutí na záznam tabulky jsou podrobnosti o dané oblasti zobrazeny v horní části záložky, kde je možné oblast editovat. Význam jednotlivých parametrů byl již rozebrán v kapitolách 6.2.4 a 6.4.2.

Referenční tagy lze zadat pomocí jejich indexů, které odpovídají očíslovanému řádku v záložce *Výsledky*. Tyto indexy se tudíž automaticky mění při jiném řazení tabulky výsledků. Uživatel může měnit i referenční souřadnice tagů, které jsou generovány podle aktuálních souřadnic tagu v okamžiku zadání daného tagu jako referenčního příslušné oblasti. Proto je doporučeno všechny reference pro danou oblast zadávat při pozastaveném snímání kamery, aby nedošlo k posunu kamery mezi vložením jednotlivých indexů.

Prvky

Poslední záložka 6.18e je věnována pohyblivým členům. Podobně jako záložka *Oblasti* umožňuje záložka 6.18e vytvářet nové prvky po kliknutí na tlačítko s ikonou zeleného plus. Vytvořený prvek je ihned přidán do tabulky všech prvků. Po kliknutí na záznam tabulky se v horní části panelu zobrazí informace o daném prvku pro editaci. Lze měnit typ prvku, současné verze CAFDIS ovšem podporuje jen aktuátor podle modelu z kapitoly 6.4.1. Každému prvku typu *Aktuátor* lze přiřadit až 4 referenční tagy pro statickou a dynamickou část pomocí indexů tagů ze záložky *Výsledky* (Současná verze modelu aktuátoru umožňuje využít informace jen ze 3 tagů, viz kapitola 6.4.1). Stejně jako u oblastí platí ze stejných důvodů doporučení zadávat referenční tagy při pozastaveném snímání kamery.

Výsledky dekompozice polohy aktuátoru jsou k nahlédnutí v posledním sloupci tabulky aktuátorů.

6.6.3 Další funkcionality

Mezi další funkcionality GUI patří například editor TM tagů ukrývající se v rozbalovacím menu *Tag*. Editor umožňuje vytvářet pouze základní rotačně invariantní vzory tagů. Editor lze ukončit stiskem klávesy *enter* nebo *escape*. Po stisknutí *enter* je vzor navíc uložen ve formátu jpg do složky */Tags/TM/* v adresáři, kde je program uložen.

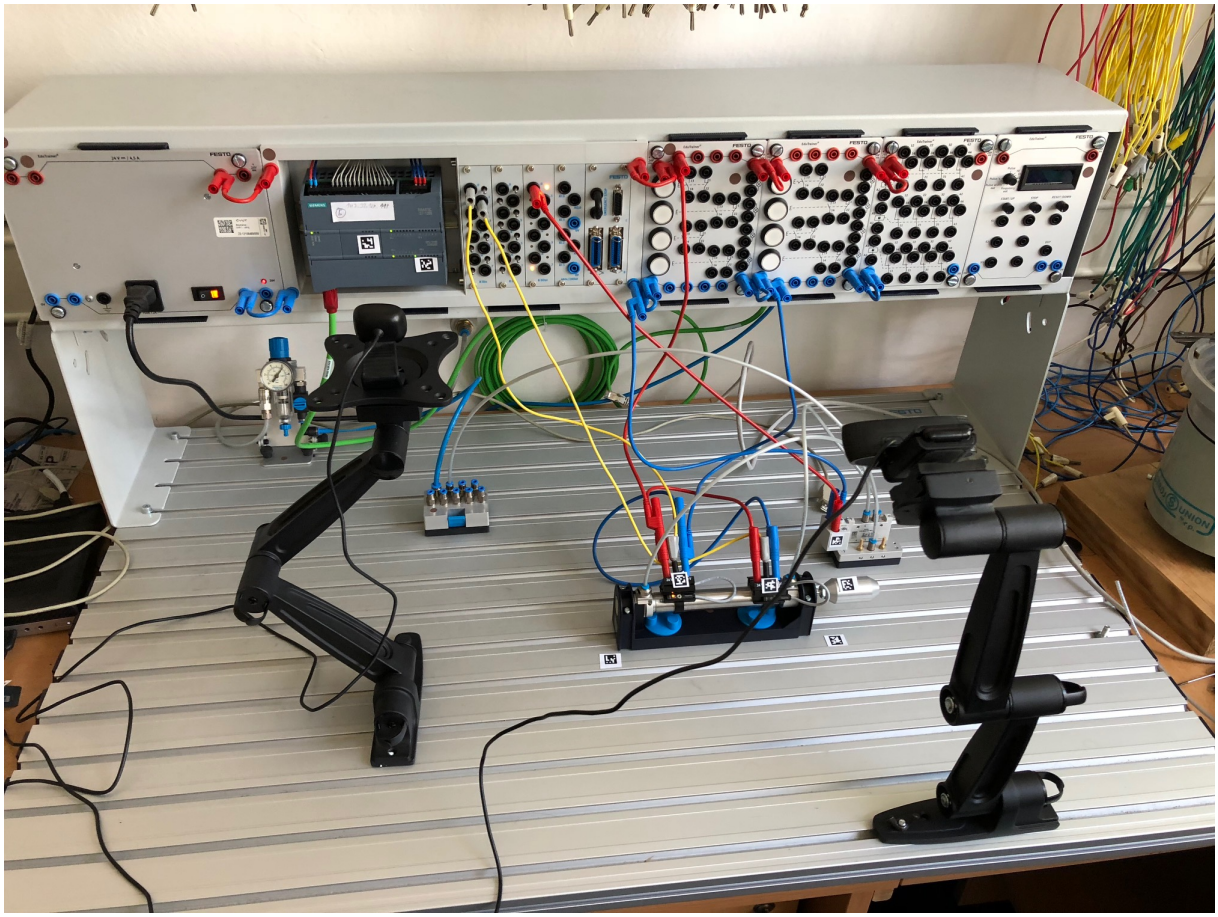
Celou konfiguraci programu lze ukládat nebo načítat prostřednictvím menu *Soubor* nebo známou klávesovou zkratkou *CTRL+S*, resp. *CTRL+O*.

V případě nejasností obsahuje CAFDIS ještě nápovědu ve formě vysvětlujícího textu na spodní liště okna aplikace, jak je vidět na obrázku 6.19.

Obrázek 6.19: Nápověda na spodní liště

7 Demonstrace fungování CAFDIS

Systém byl otestován na PLC Siemens S7-1200 v didaktickém provedení s jedním dvojčinný pneumatickým lineárním aktuátorem ovládaným monostabilním rozvaděčem. Pracoviště je zachyceno na obrázku 7.1. K monitorování jsou použity 2 kamery. Jedna sleduje indikační LED na PLC, druhá kamera, s větším rozlišením, monitoruje samotný proces, a to konkrétně polohu aktuátoru a stavy indikační LED koncových senzorů a pneumatického rozvaděče.



Obrázek 7.1: Pracoviště na testování CAFDIS

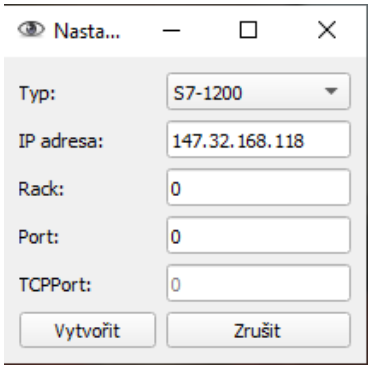
PLC byl naprogramován ve vývojovém prostředí SIMATIC STEP 7 (TIA Portal). Řídící program nejprve přestaví rozvaděč, který vysune lineární aktuátor. Po sepnutí koncového senzoru A1 ve vysunuté poloze program počká 5 s, poté je rozvaděč přestavěn zpět a motor se vrací do výchozí polohy, kde sepne druhý koncový senzor A0. Po jeho sepnutí program čeká 5 s a pokračuje od začátku opětovnou aktivací ventilu.

PLC tedy pracuje se 3 proměnnými: 2 koncovými senzory a 1 rozvaděčem. Nastavení a umístění těchto proměnných v programu SIMATIC STEP 7 bylo již ukázáno na obrázku

6.7 v kapitole 6.3.

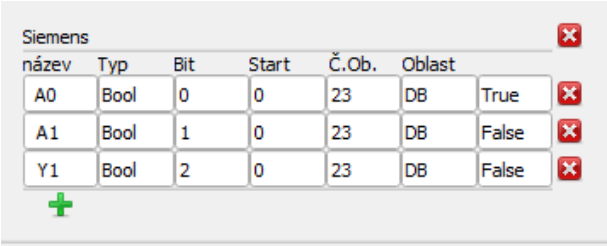
7.1 Nastavení komunikace s PLC

Připojení PLC k systému CAFDIS proběhlo bez potíží s nastavením podle 7.2.



Obrázek 7.2: Nastavení připojení k PLC

Proměnné, které systém z PLC čte, jsou popsány tabulkou v GUI, která je zobrazena na obrázku 7.3. Poslední sloupec ukazuje aktuální hodnotu dané proměnné. Název proměnné v systému CAFDIS nemusí odpovídat názvu v PLC programu.



Obrázek 7.3: Nastavení proměnných ke čtení z PLC

7.2 Oblasti

PLC, který snímá druhá kamera, je označen pomocí 2 ArUco tagů, které jsou použity jako reference 3 oblastí na PLC, které představují 3 indikační LED (2 koncové senzory, 1 rozvaděč). Částečná invariantnost při pohybu kamery je demonstrována na obrázku 7.4. Oblast jedné z LED je značena modrým obdélníkem (LED se nachází zhruba uprostřed v horní řadě). Je vidět, že obdélník přiléhá na indikační LED uspokojivě přesně i po posunu kamery. K vyhodnocení barvy oblasti je použit výpočet pomocí mediánu z kapitoly 6.4.2.

Výrazně lépe je na tom přesnost určení oblastí kamery monitorující zkoumaný proces, kde jsou ke každé oblasti použity 3 referenční tagy. Výsledky zkoušky posunu kamery

(a) PLC ve výchozí poloze (b) PLC po pohybu kamery

Obrázek 7.4: Detekovaná oblast na PLC při posunu kamery

dokumentuje obrázek 7.5, na kterém je modrým rámečkem zvýrazněna oblast koncového senzoru vyjeté polohy aktuátoru.

Modrá oblast na všech snímcích bez větších odchylek kopíruje obrys indikační LED. Lze si všimnout, že v případě polohy podle 7.5e nebyl jeden z referenčních tagů detekován (označen červeným křížem), aplikace však i tak vyhodnotila polohu oblasti správně podle poslední úspěšně detekované polohy, kterou červený kříž označuje. K vyhodnocení barvy byl pro indikační LED koncových senzorů použit princip maxima, pro indikační LED rozvaděče byl použit medián.

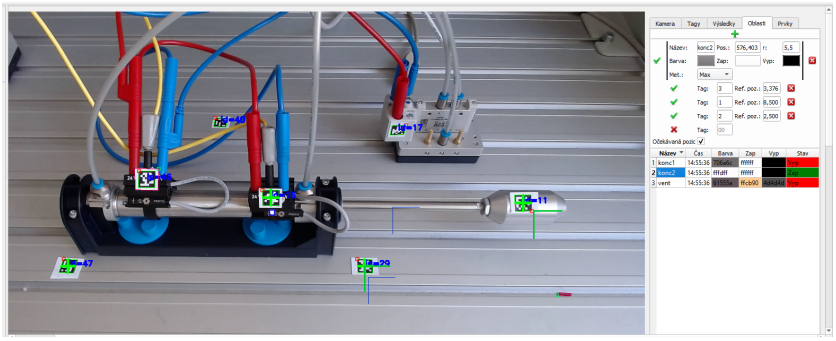
7.3 Aktuátor

Pro definování aktuátoru byly použity 2 tagy pro určení statické části a jeden tag na pohyblivé části. S touto konfigurací je model z kapitoly 6.4.1 schopen určit posuv v obou osách. Díky tomu, že je statická část definována pomocí 2 tagů, bude zjištěné posunutí nezávislé na poloze kamery (z hlediska definování směru os a jejich měřítka). Obrázek 7.6 zachycuje výsledky vyhodnocení polohy aktuátoru v obou myšlených polohách.

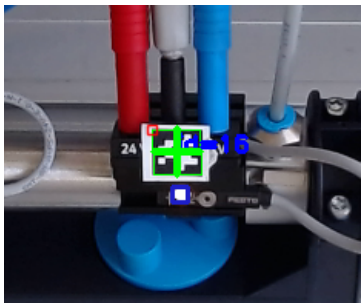
V dolní úvrati se hodnoty posuvu aktuátoru liší od výchozí polohy jen o jednotky pixelů, což může být způsobeno malou nepřesností v detekci polohy jednotlivých tagů nebo tím, že aktuátor nezajel do téže pozice, ve které byl při definování jeho výchozího stavu. V každém případě je chyba polohy natolik malá, že na další práci s údaji o poloze nebude mít vliv.

V horní úvrati je aktuátor posunut ve směru osy x o 270 jednotek, které odpovídají pixelům na snímku ve výchozí poloze kamery. O jemném pohybu kamery svědčí koeficienty zvětšení, které ukazují, že štítky statické části jsou nyní detekovány asi o 0,2 % blíže k sobě než ve výchozím stavu. Tato drobná odchylka může samozřejmě být způsobena i nepřesnou detekcí.

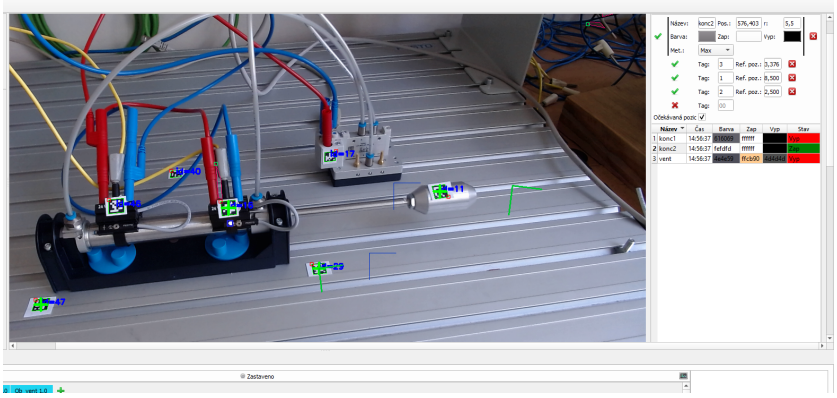
Při použití 2 štítků na statickou část a 1 na pohyblivou není možné, aby koeficient zkosení měl jinou než nulovou hodnotu. Je uvěřitelné, že tak malé číslo jako $-9 \cdot 10^{-17}$



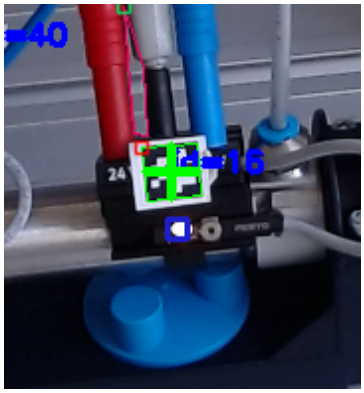
(a) Koncový senzor ve výchozí poloze



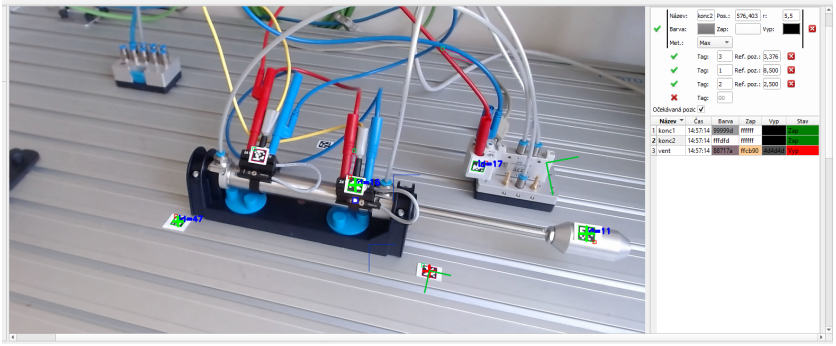
(b) Detail



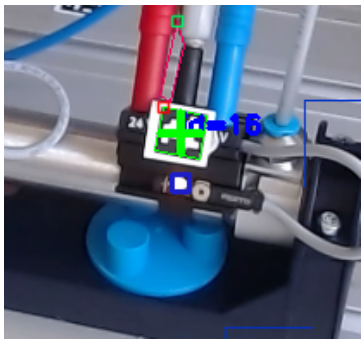
(c) Senzor po posunu kamery



(d) Detail



(e) Senzor po posunu kamery



(f) Detail

Obrázek 7.5: Detekovaná oblast LED koncového senzoru při posunu kamery

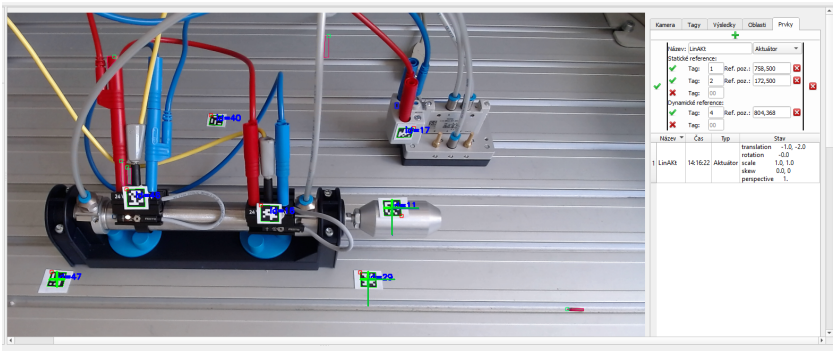
vzniklo zaokrouhlovací chybou při poměrně složitém výpočtu transformační matice a její dekompozice.

7.4 Diagnostika

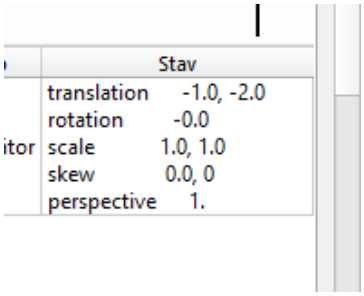
Pro ukázku schopností diagnostiky byla sestavena pravdivostní tabulka chyb, kterou lze vidět na obrázku 7.7.

Diagnostika je schopna odhalit 4 chyby:

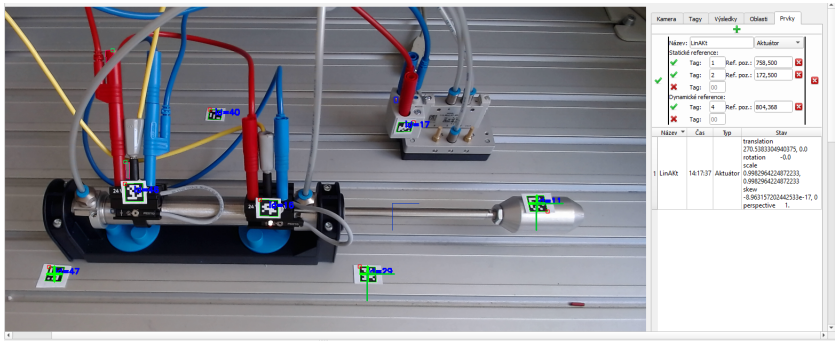
- První řádek obsahuje 2 události. První z nich je aktivní, pokud je pístnice motoru vysunuta méně než 200 jednotek. Vysunuté poloze odpovídá asi 270 jednotek, jedná



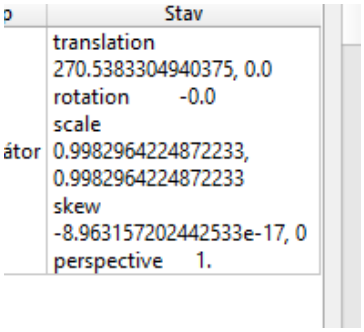
(a) Aktuátor v zajeté poloze



(b) Detail výsledků

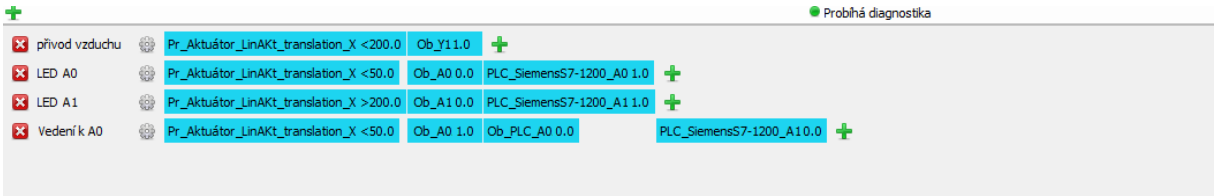


(c) Aktuátor ve vyjeté poloze



(d) Detail výsledků

Obrázek 7.6: Detekovaný aktuátor



Obrázek 7.7: Ukázka nastavení pravdivostní tabulky diagnostiky

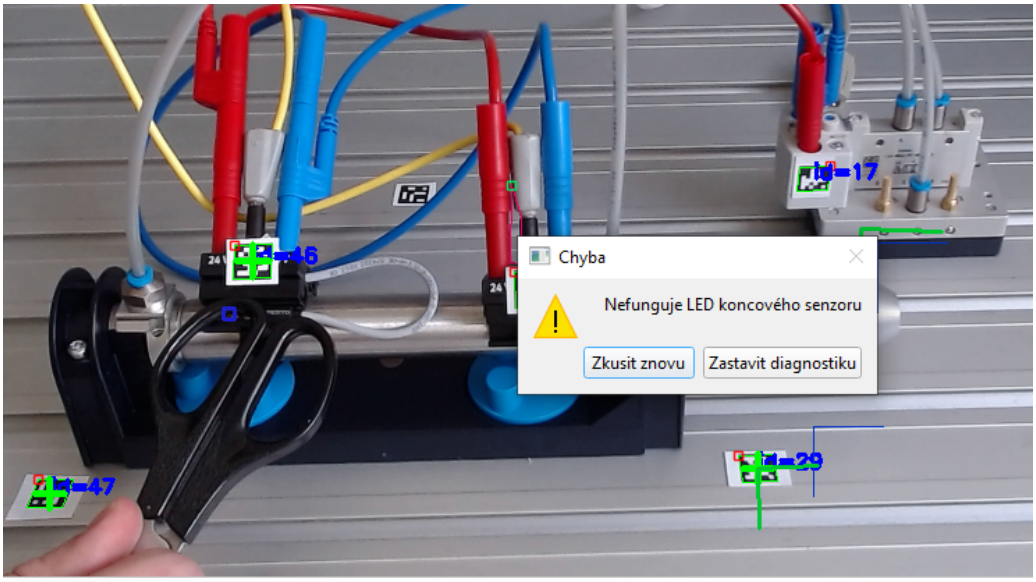
se tedy o stav, kdy motor není plně vysunutý. Druhá událost je aktivní, pokud je oblast indikační LED ventilu ve stavu zapnuto.

Tento případ odpovídá buď přerušenému přívodu vzduchu, nebo zaseknuté pístnici motoru, tak jak to popisuje příslušná chybová hláška.

- Druhý a třetí případ odpovídají vadné indikační LED na koncových senzorech motoru. Nejedná se tedy o chybu, která ovlivní zkoumaný proces, ale může zapříčinit falešné oznámení o dalších chybách, zvláště u složitějších procesů. Proto je nutné i takovéto chyby kontrolovat a odstranit.

  Příslušná indikační LED je označena za vadnou, pokud je motor v poloze odpovídající koncovému senzoru, jehož indikační LED je vypnutá, a současně PLC detekuje daný koncový senzor jako sepnutý.

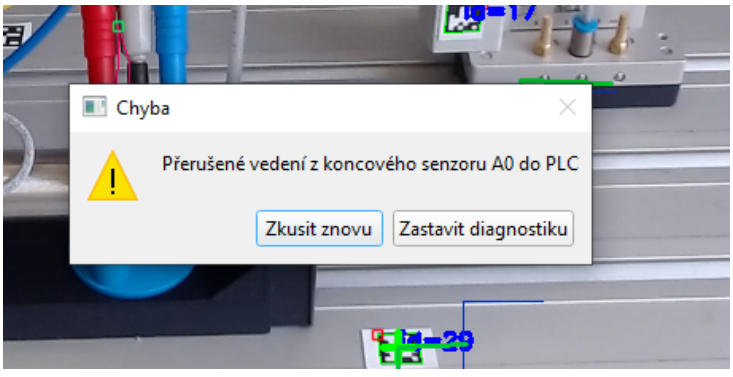
  Detekce této chyby je demonstrována na obrázku 7.8.



Obrázek 7.8: Odhalení vadné indikační LED

- Poslední řádek tabulky detekuje vadnou kabeláž mezi jedním z koncových senzorů a PLC. Případ obsahuje 4 události:
  - Pohyblivá část aktuátoru je v poloze odpovídající méně než 50 jednotkám, tedy v zajeté poloze.
  - Indikační LED koncového senzoru zajeté polohy je zapnutá.
  - Indikační LED na vstupu PLC odpovídajícím danému senzoru je vypnutá.
  - PLC nedetekuje signál z daného koncového senzoru.

  Chyba byla vyvolána vysunutím vodiče ze vstupní sběrnice PLC. Chyba byla správně detekována a její chybová hláška je k vidění na obrázku 7.9.



Obrázek 7.9: Odhalení vadného vedení mezi PLC a koncovým senzorem

Stejnými nástroji lze odhalit jakoukoli chybu, kterou lze popsat pomocí nástrojů kombinační logiky. Limitujícím faktorem je počet vstupních proměnných, které je systém schopen detekovat.

8 Závěr

8.1 Shrnutí práce

Tato práce představuje nový způsob diagnostiky chyb ve výrobních systémech. Standardní způsoby detekce většinou neodhalí příčinu chyby způsobenou vadou na senzorech nebo výrobních zařízeních. Navržený systém s označením CAFDIS (Camera Aided Fault Diagnosis and Isolation System) využívá k odhalení původu chyby nové proměnné, které jsou získány analýzou obrazu z kamer monitorujících zkoumaný proces.

V teoretické části práce bylo popsáno několik metod CV (Strojové vidění) aplikovatelných na řešený problém. Možné způsoby řešení byly posléze přehledně porovnány a pro implementaci do CAFDIS byly vybrány metody TM (Template Matching) a detekce FMS (Systémy referenčních tagů) pomocí knihovny ArUco. Obě metody detekují na snímcích z kamer sledovací tagy, kterými je výrobní zařízení označeno. Na základě polohy nalezených tagů vyhodnotí systém polohu označených akčních členů a polohy indikačních LED, kterým je přiřazen odpovídající stav. Systém taktéž dokáže komunikovat s PLC zařízeními firmy Siemens, z nichž dokáže získat hodnoty vnitřních proměnných.

Všechny tyto nové proměnné jsou použity spolu se známými proměnnými z PLC v diagnostickém modulu systému CAFDIS, kde jsou použity pro přesnější určení příčiny vzniklých chyb.

CAFDIS nastavuje pověřený uživatel prostřednictvím GUI (Grafické uživatelské rozhraní) a zadává systému další informace o zkoumaném procesu, které jsou nezbytné pro detekci a identifikaci chyby.

8.2 Splnění cílů

Zadání práce obsahovalo následující body:

1. Seznámit se s knihovnou pro strojové vidění OpenCV.
2. Vytvořit jednoduchý program pro automatickou identifikaci rozsvícené kontrolky pomocí analýzy obrazu.
3. Navrhnout jednoduchý diagnostický program pro systém logického řízení.
4. Do diagnostického programu začlenit informace z analýzy obrazu.

Systém CAFDIS je napsán v jazyce Python s použitím funkcionalit knihovny OpenCV, jak bylo zadáno. Systém dokáže určit stav indikační LED způsobem vysvětleným v kapitolách 6.2.4 a 6.4.2. Fungování této metody je demonstrováno na příkladu v kapitole 7.2. Diagnostický modul systému je popsán v kapitole 6.5. Ke svému fungování dokáže využívat data nejen z analýzy obrazu, ale i z PLC a data zadaná uživatelem. Diagnostický modul je plně funkční, jak bylo ukázáno v kapitole 7.4.

Všechny body zadání byly splněny. Systém je nad rámec zadání schopen identifikovat polohy akčních členů a přijímat informace z PLC, je vybaven rozsáhlým GUI a splňuje i další požadavky dané autorem práce v kapitole 2.1, jako například invariantnost systému při posunutí kamery.

8.3 Další rozšíření a doporučení

Návrhy na rozšíření v dalších verzích systému byly již popsány na konci příslušných kapitol.

Díky dobře členěnému zdrojovému kódu, psanému podle pravidel objektově orientovaného programování, spočívá většina návrhů na rozšíření ve vytvoření nových tříd, které dědí z tříd již existujících. Nově vytvořená třída v sobě zahrnuje nové metody i metody přepsané z rodičovské třídy. Název této třídy je poté většinou nutné umístit do příslušného seznamu (nebo slovníku) konstruktorů v sekci pracující s danými objekty tak, jak je v konkrétních kapitolách popsáno. Blíže je tento proces rozšiřování popsán v příslušných třídách ve zdrojovém kódu. Tímto způsobem lze do systému přidat další metody detekce tagů, nové typy akčních členů nebo kompatibilitu s PLC jiných firem.

Velký význam by mělo i vyzkoušení dalších metod detekce komponent výrobního procesu, například neuronových sítí.

Značnou výhodu by systému CAFDIS přinesla standardizace designu jednotlivých komponent. Proces analýzy obrazu by poté mohl být spolehlivější a celý systém by se stal jednodušším na užívání. Sjednocení designu součástí různých výrobců by mohlo umožnit i automatické nastavování diagnostického systému na zkoumaný proces a automatické odhalení celé řady chyb bez nutnosti začleňovat je do pravdivostní tabulky. Pokud by se podobné systémy jako CAFDIS rozšířily, nemusely by tyto myšlenky být tak utopické, jak se dnes může zdát.

Bibliografie

[1] V. Chauhan a B. Surgenor, "A Comparative Study of Machine Vision Based Methods for Fault Detection in an Automated Assembly Machine", *Procedia Manufacturing*, roč. 1, s. 416–428, 2015, ISSN: 23519789. DOI: 10.1016/j.promfg.2015.09.051. WWW: https://linkinghub.elsevier.com/retrieve/pii/S2351978915010513.

[2] A. Bhuvanesh a M. M. Ratnam, "Automatic detection of stamping defects in leadframes using machine vision, Overcoming translational and rotational misalignment", *The International Journal of Advanced Manufacturing Technology*, roč. 32, č. 11-12, s. 1201–1210, 2007-4-23, ISSN: 0268-3768. DOI: 10.1007/s00170-006-0449-y. WWW: http://link.springer.com/10.1007/s00170-006-0449-y.

[3] J. Köhler, A. Pagani a D. Stricker, "Detection and Identification Techniques for Markers Used in Computer Vision", roč. 19, s. 36–44, led. 2010. DOI: 10.4230/OASIcs.VLUDS.2010.36.

[4] M. Fiala, "Comparing ARTag and ARToolkit Plus fiducial marker systems", 2005.

[5] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas a M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion", *Pattern Recognition*, roč. 47, s. 2280–2292, červ. 2014. DOI: 10.1016/j.patcog.2014.01.005.

[6] J. Gao, V. Kulkarni, H. Ranavat, L. Chang a H. Mei, "A 2D Barcode-Based Mobile Payment System", in *2009 Third International Conference on Multimedia and Ubiquitous Engineering*, 2009, s. 320–329.

[7] M. Kaltenbrunner, "reacTIVision and TUIO: a tangible tabletop toolkit", s. 9–16, led. 2009. DOI: 10.1145/1731903.1731906.

[8] M. Kaltenbrunner a R. Bencina, "reacTIVision: a computer-vision framework for table-based tangible interaction", *TEI'07: First International Conference on Tangible and Embedded Interaction*, led. 2007. DOI: 10.1145/1226969.1226983.

[9] H. Nishino, "Topolo Surface: A 2D Fiducial Tracking System Based on Topological Region Adjacency and Angle Information", *JIP*, roč. 18, s. 16–25, led. 2010. DOI: 10.2197/ipsjjip.18.16.

[10] K. Shabalina, A. Sagitov, L. Sabirova, H. Li a E. Magid, "ARTag, AprilTag and CALTag Fiducial Systems Comparison in a Presence of Partial Rotation: Manual and Automated Approaches", in. led. 2020, s. 536–558, ISBN: 978-3-030-11291-2. DOI: 10.1007/978-3-030-11292-9_27.

[11] M. Brown, R. Szeliski a S. Winder, "Multi-image matching using multi-scale oriented patches", in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, sv. 1, 2005, 510–517 vol. 1.

[12] A. Mordvintsev, *Feature Matching*. WWW: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html.

[13] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal of Computer Vision*, roč. 60, č. 2, s. 91–110, 2004, ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000029664.99615.94. WWW: http://link.springer.com/10.1023/B:VISI.0000029664.99615.94.

[14] C. S. Kenney, M. Zuliani a B. S. Manjunath, "An axiomatic approach to corner detection", in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, sv. 1, 2005, s. 191–197 vol. 1.

[15] A. Mordvintsev, *Introduction to SIFT (Scale-Invariant Feature Transform)*. WWW: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html.

[16] H. Bay, T. Tuytelaars a L. Van Gool, "SURF: Speeded Up Robust Features", in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof a A. Pinz, ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, s. 404–417, ISBN: 978-3-540-33833-8.

[17] P. Viola a M. Jones, "Rapid object detection using a boosted cascade of simple features", in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, sv. 1, 2001, s. I–I.

[18] E. Rosten a T. Drummond, "Machine Learning for High-Speed Corner Detection", in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof a A. Pinz, ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, s. 430–443, ISBN: 978-3-540-33833-8.

[19] A. Mordvintsev, *FAST Algorithm for Corner Detection*. WWW: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_fast/py_fast.html#fast-feature-detector-in-opencv.

[20] M. Calonder, V. Lepetit, C. Strecha a P. Fua, "BRIEF: Binary Robust Independent Elementary Features", in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos a N. Paragios, ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, s. 778–792, ISBN: 978-3-642-15561-1.

[21] *Hamming distance*. WWW: https://encyclopediaofmath.org/wiki/Hamming_distance.

[22] E. Rublee, V. Rabaud, K. Konolige a G. Bradski, "ORB: An efficient alternative to SIFT or SURF", in *2011 International Conference on Computer Vision*, 2011, s. 2564–2571.

[23] *The OpenCV Reference Manual*, 4.3.0, Itseez, 2014.

[24] S. O'Hara a B. A. Draper, "Are you using the right approximate nearest neighbor algorithm?", in *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, 2013, s. 9–14.

[25] P. KaewTraKulPong a R. Bowden, "An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection", in *Video-Based Surveillance Systems: Computer Vision and Distributed Processing*, P. Remagnino, G. A. Jones, N. Paragios a C. S. Regazzoni, ed. Boston, MA: Springer US, 2002, s. 135–144, ISBN: 978-1-4615-0913-4. DOI: 10.1007/978-1-4615-0913-4_11. WWW: https://doi.org/10.1007/978-1-4615-0913-4_11.

[26] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction", in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, sv. 2, 2004, s. 28–31 Vol.2.

[27] A. B. Godbehere, A. Matsukawa a K. Goldberg, "Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation", in *2012 American Control Conference (ACC)*, 2012, s. 4305–4312.

[28] H. H. A. Kadouf a Y. M. Mustafah, "Colour-based Object Detection and Tracking for Autonomous Quadrotor UAV", *IOP Conference Series: Materials Science and Engineering*, roč. 53, 012–086, 2013. DOI: 10.1088/1757-899x/53/1/012086. WWW: https://doi.org/10.1088%2F1757-899x%2F53%2F1%2F012086.

[29] I. Bukovský, *Počítačové vidění a virtuální realita, Konvoluční neuronové sítě*, Výukové materiály, Praha, 2020. WWW: http://aspicc.fs.cvut.cz/jupyter/user/guest/notebooks/courseware/PocVidVR/10\%20(Konvolun\%20hlubok\%20neuronov\%20st\%20(deep\%20neural\%20networks)\%20(konvolun\%20vrstva,\%20supervizorovan\%20vrstvy),\%20YOLO,\%20SSD\%20(single\%20shot\%20detector))/Class\%20Notes/PocVidVR\%20ClassNote\%2010.ipynb?redirects=1.

[30] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard a L. D. Jackel, "Handwritten Digit Recognition with a Back-Propagation Network", in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, ed., Morgan-Kaufmann, 1990, s. 396–404. WWW: http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf.

[31] *Obrazové filtry*, 2019. WWW: ObrazovfiltryMendelUniversityinBrno[online].Brno,2019[cit.2020-07-04].Dostupnz:https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=18431.

[32] S. Sumit, "A Comprehensive Guide to Convolutional Neural Networks", *Towards data science*, roč. 2018, WWW: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[33] S. Ren, K. He, R. Girshick a J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama a R. Garnett, ed., Curran Associates, Inc., 2015, s. 91–99. WWW: http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf.

[34] R. Huang, J. Pedoeem a C. Chen, "YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers", in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, s. 2503–2510.

[35] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu a A. C. Berg, "SSD: Single Shot MultiBox Detector", in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe a M. Welling, ed., Cham: Springer International Publishing, 2016, s. 21–37, ISBN: 978-3-319-46448-0.

[36] Yizong Cheng, "Mean shift, mode seeking, and clustering", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, roč. 17, č. 8, s. 790–799, 1995.

[37] A. Mordvintsev, *Meanshift and Camshift*. WWW: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_meanshift/py_meanshift.html#meanshift.

[38] G. R. Bradski, "Real time face and object tracking as a component of a perceptual user interface", in *Proceedings Fourth IEEE Workshop on Applications of Computer Vision. WACV'98 (Cat. No.98EX201)*, 1998, s. 214–219.

[39] Z. Zhang, "A flexible new technique for camera calibration", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, roč. 22, č. 11, s. 1330–1334, 2000.

[40] J. Garcia-Rodrigez, *Advancements in Computer Vision and Image Processing*. Spain: IGI Global, 2018, ISBN: 9781522556299.

[41] J. Heikkila a O. Silven, "A four-step camera calibration procedure with implicit image correction", in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, s. 1106–1112.

[42] G. Bradski, "The OpenCV Library", *Dr. Dobb's Journal of Software Tools*, 2000.

[43] G. Van Rossum a F. L. Drake Jr, *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[44] PyQT, "PyQt Reference Guide", 2012. WWW: `http://www.riverbankcomputing.com/static/Docs/PyQt4/html/index.html`.

[45] *Qt, Qt documentation*, 2020. WWW: `https://doc.qt.io/qt-5/`.

[46] A. M. Reza, "Realization of the Contrast Limited Adaptive Histogram Equalization (CLAHE) for Real-Time Image Enhancement", *The Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology*, roč. 38, č. 1, s. 35–44, 2004, ISSN: 0922-5773. DOI: `10.1023/B:VLSI.0000028532.53893.82`. WWW: `http://link.springer.com/10.1023/B:VLSI.0000028532.53893.82`.

[47] *Detection of ArUco Markers*, 2020. WWW: `https://docs.opencv.org/trunk/d5/dae/tutorial_aruco_detection.html`.

[48] *Geometric Image Transformations, getPerspectiveTransform*, 2019. WWW: `https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html\#getperspectivetransform`.

[49] D. Nardella, S. Preeker a G. Molenaar, *Snap7*, 2013.

[50] *SIMATIC STEP7*, 2020. WWW: `https://new.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal/software/step7-tia-portal/simatic-step7-professional.html`.

[51] *Simply Automationized*, 2017. WWW: `http://simAplyautomationized.blogspot.com`.