VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

DETEKCE VÍCE OBJEKTŮ S VYUŽITÍM POČÍTAČOVÉHO VIDĚNÍ

DETECTION OF MULTIPLE OBJECTS USING COMPUTER VISION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Štěpán Maršala
AUTHOR

VEDOUCÍ PRÁCE — Ing. Roman Parák
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav automatizace a informatiky
Student:	**Štěpán Maršala**
Studijní program:	Strojírenství
Studijní obor:	Základy strojního inženýrství
Vedoucí práce:	**Ing. Roman Parák**
Akademický rok:	2019/20

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Detekce více objektů s využitím počítačového vidění

Stručná charakteristika problematiky úkolu:

Práce bude zahrnovat rešerši knihovny OpenCV, knihoven pro strojové – hluboké učení a jejich využití v oblasti detekce objektů, seznámení se s problematikou detekce objektů v průmyslových aplikacích a obecně.
Předmětem práce bude návrh řídícího programu pro zpracování obrazových dat z kamery s využitím znalostí nabytých z teoretické části práce. Praktická část práce bude také zahrnovat návrh a implementace programu pro OPC UA komunikaci s vybranou řídící jednotkou (průmyslový robot, PLC, atd.) Závěr práce bude věnován implementaci návrhu řídícího programu pro detekci objektů a ověření funkčnosti vytvořeného řešení.
Práce předpokládá aktivní přístup studenta a nutnost práce v laboratoři.

Cíle bakalářské práce:

– Nastudujte problematiku detekci objektů v průmyslových aplikacích. Zpracujte přehled aktuálního stavu v dané oblasti.
– Proveďte rešerši knihovny OpenCV, knihoven pro strojové učeni – hluboké učení a jejich využití v oblasti detekce objektů.
– Navrhněte řídící program pro zpracování obrazových dat z kamery.
– Implementujte návrh řídícího programu.
– Navrhněte a implementujte program pro OPC UA komunikaci.
– Ověřte funkčnost vytvořeného řešení.

Seznam doporučené literatury:

SONKA, M., HLAVAC, V., BOYLE, R.: Image Processing, Analysis, and Machine Vision, Third Edition, Thornson, 2008.

WEIGEL, B. Deep learning for a digital age: technology's untapped potential to enrich higher education. San Francisco: Jossey-Bass, c2002. Jossey-Bass higher and adult education series. ISBN 0-7879-5613-9.

VATSA, M., SINGH, R. a MAJUMDAR, A. Deep learning in biometrics. Boca Raton: CRC Press, 2018.

OpenCV: Open Computer Vision library. https://opencv.org/

SHAPIRO, L., STOCKMAN, G.: Computer vision. Prentice Hall, 2001.

PAJDLA, T. Global optimization in camera and robot calibration: Globální optimalizace v kalibraci kamer a robotů. V Praze: České vysoké učení technické, [2017]. ISBN 978-80-01-06114-5.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2019/20

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D.	doc. Ing. Jaroslav Katolický, Ph.D.
ředitel ústavu	děkan fakulty

ABSTRAKT

Cílem bakalářské práce je navrhnout a prakticky zrealizovat řídící algoritmus pro detekci objektů různých tvarů, barev a velikostí. Práce je rozdělena do pěti kapitol. První kapitola obsahuje rešerši dosavadního poznání v oblasti strojového vidění. Druhá kapitola popisuje funkce a využití open source knihovny OpenCV, která je použitá v praktické části práce. Praktická část začíná kapitolou shrnující použitou techniku. Následuje kapitola, která popisuje samotnou úlohu třídění objektů. Jsou zde popsány všechny technické aspekty a vysvětleny problémy, které bylo nutno překlenout. V poslední kapitole najde čtenář zprávu o zrealizování praktické části práce v laboratoři. Shrnutí výsledků laboratorního experimentu a celkového poznání je v závěru samotné práce.

ABSTRACT

The aim of the bachelor's thesis is designing and realizing an algorithm for different shapes, colors, and sizes object detection. The paper is divided into five chapters. The first chapter includes introduction to so far developed computer vision. The second chapter describes functions and uses of open source library OpenCV, which is used in practical part of this paper. Practical part begins with chapter which covers all used hardware and technology. Next chapter describes task of separating objects. Further are included all technical aspects and problem solutions. In last chapter is a report about practical work realization in laboratory. Summary of all experiment results and paper cognition is in conclusion of the paper.

KLÍČOVÁ SLOVA

počítačové vidění, detekce objektů, zpracování obrazu, Python, OpenCV

KEYWORDS

computer vision, object detection, image processing, Python, OpenCV

BIBLIOGRAFICKÁ CITACE

MARŠALA, Štěpán. Detekce více objektů s využitím počítačového vidění. Brno, 2020. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/124827. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky. Vedoucí práce Roman Parák.

PODĚKOVÁNÍ

Chtěl bych poděkovat vedoucímu této práce Ing. Romanu Parákovi za skvělou spolupráci a přátelský přístup. Dále pak za všechny rady a pomoci, které se mi v budoucnu určitě zúročí.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením Ing. Romana Paráka a s použitím literatury uvedené v seznamu literatury.

V Brně dne 1. 3. 2019 ……

Štěpán Maršala

OBSAH

1 ÚVOD

Stejně tak jako člověk vyhodnocuje prostředí kolem sebe nejen pomocí očí, ale i mozkem, tak i strojům nestačí kamery a senzory k tomu, aby mohly vyhodnocovat stavy v okolí. Člověk si tento fakt většinou neuvědomuje a když se rozhlíží po okolí, přirozeně rozdělí obraz ve svém mozku na objekty, těm pak přiřadí vlastnosti jako barvy, tvary, velikosti nebo vzdálenost. S těmito informacemi potom mozek dál pracuje.

Úkolem strojového vidění je poskytnout stejnou schopnost i strojům. Jinými slovy, nestačí jen vidět. Vyhodnocování obrazu u strojů probíhá pomocí matematických metod. Důležitým pomocníkem v této oblasti je umělá inteligence, která dokáže detekovat předměty podle podobně vypadajících objektů. Strojové vidění nachází v turbulentním technickém prostředí moderního světa mnohá využití. Bezpochyby poslouží všude tam, kde člověk rutinně plýtvá kapacitou svého mimořádně sofistikovaného mozku při činnostech, které stroje s absencí analýzy obrazu doposud nebyly schopny vykonávat. Stejně tak se strojové vidění stává skvělým pomocníkem při zvyšování bezpečnosti nás všech, například v automobilech nebo bezpečnostních systémech.

Matematické funkce a práci s obrazem v praxi pomáhá řešit knihovna OpenCV, která je volně šiřitelná a použitelná. Knihovna načítá obraz z kamery nebo fotografie a následně jej může upravovat pomocí filtrů a jiných nástrojů. Pomocí knihovny řídící program pak zpracuje upravený obraz a označí výsledné oblasti, tvary nebo body.

Cílem této bakalářské práce je sestavit a zrealizovat funkční algoritmus pro vyhodnocování obrazu z kamery. Algoritmus bude v obrazu vyhledávat objekty určitých tvarů, barev a velikostí. Algoritmus bude pracovat především s funkcemi knihovny OpenCV. Objektům přiřadí reálné souřadnice a veškeré informace odešle do PLC ke zpracování. PLC pak pomocí ABB robota roztřídí tyto objekty dle nastavených preferencí. K tomuto třídění poslouží QR kódy, které v sobě nesou informaci.

Práce je rozdělena do pěti kapitol. První kapitola popisuje stav současného poznání v oblasti strojového vidění. V další kapitole je popsána knihovna OpenCV a její nejdůležitější funkce, které byly použity v práci. Pozornost je věnována především technickým a matematickým principům těchto funkcí. Praktická část práce začíná výčtem a podrobným popisem nejdůležitější techniky, která byla použita pro zrealizování úlohy. Jedná se o kameru, robotické rameno, vakuovou přísavku, 3D tiskárnu, komunikační protokol a PLC. V další kapitole najdete popis přípravy praktické části. Je zde popsána samotná úloha, zapojení hardwaru a chod řídícího programu. Pozornost je věnována i kalibraci, která je nutná zejména pro přesné detekování objektů. Objekty k detekci byly vytištěny na 3D tiskárně. V poslední kapitole je zpráva o skutečném průběhu experimentu v laboratoři. Výsledky práce jsou pak zhodnoceny v závěru.

2 PŘEHLED SOUČASNÉHO STAVU POČÍTAČOVÉHO VIDĚNÍ

Počítačové vidění je hojně rozšířeným nástrojem pro vnímání okolního prostředí. Často se můžeme setkat i s pojmem strojové vidění, který má stejný význam. Strojové vidění umožnuje strojům vidět. Viděním pak rozumíme analýzu viděného obrazu a rozlišení objektů [1]. Základní periférií pro toto vnímaní je kamera, jejíž obraz je počítačem zpracováván a vyhodnocován. Způsobů, jak obraz z kamery vyhodnotit a získat z něj informace je více. Automatická analýza obrazu dokáže být velmi rychlá a přesná. V moderním světě nachází mnohá uplatnění.

2.1 Historický vývoj

Už v roce 1920 si lidé uvědomovali možnosti, které by strojové vidění přineslo [1]. V minulém století nebyly technické možnosti takové jako dnes, a proto se v tomto směru pracovalo spíše na vývoji matematických metod, které ve strojovém vidění analyzují obraz dodnes. Strojové neboli počítačové vidění (v zahraniční literatuře označováno pod pojmem *computer vision*) bylo při svém vývoji ovlivňováno i jinými obory jako jsou: počítačová grafika, fotogeometrie, zpracování obrazu a problematika 3D modelování. Všeobecně je za otce zakladatele považován Američan Larry Roberts [2], který působil na MIT. Z jeho disertační práce dlouho poté čerpali jeho následovníci mimo jiné i v oblasti umělé inteligence. Tehdejší algoritmy byly jen stěží použitelné. Důležitým milníkem byla práce Davida Marra. Díky němu bylo konečně možné pracovat i s vizualizací 3D objektu. Jeho program byl časem vylepšen a hojně využíván.

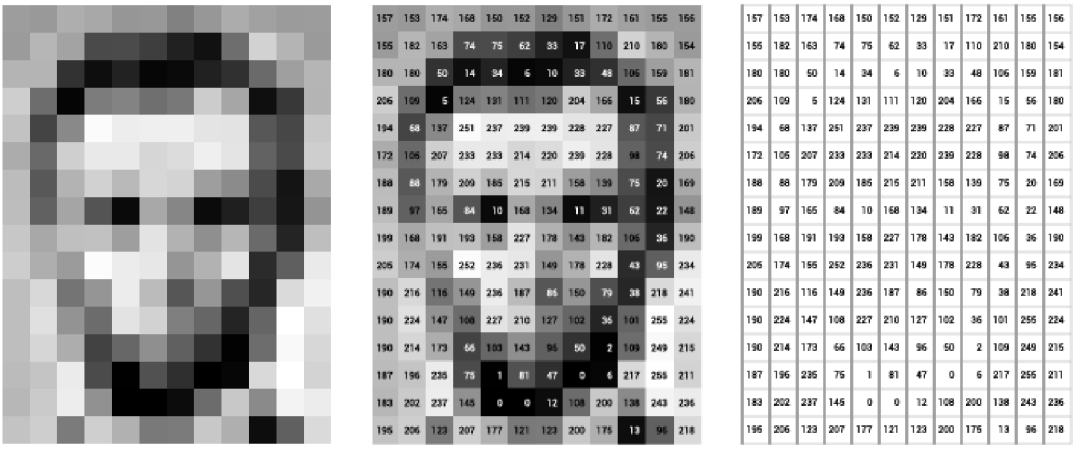
Velký vzestup využívání strojového vidění mohl nastat až v době, kdy to umožňovaly výpočetní kapacity počítačů. Tehdejší počítače byly nevýkonné a navíc obrovské. Důležité jsou i kamery. Ty byly velké a neměly takové rozlišení jako dnes. S dnešními miniaturními mikropočítači a levnými kamerami s vysokým rozlišením můžeme aplikovat strojové vidění v mnoha oblastech.

2.2 Detekce objektů

Důležitou kapitolou v počítačovém vidění je detekce objektů (v zahraniční literatuře *object detection*). Mimo detekci objektů se totiž ve strojovém vidění můžeme setkat například ještě s celkovou identifikací obrazu, detekcí pohybu, měřením vzdálenosti a další.

Obecně počítač nevidí obraz tak jako my lidé, ale načítá matici čísel z kamery [3]. Každý pixel na obrázku představuje pro počítač jeden prvek v matici. Pro zjednodušení se v počítačovém vidění často obraz převádí do černobílé podoby, kde je pak každý pixel

reprezentován číslem od 1 do 255. Jsou-li pro detekci potřeba i barvy, je matice obsáhlejší o informace o barevném složení pixelů.



Obr. 1: Matice pixelů [4]

V takovéto matici je pak pro počítač jednodušší hledat objekty. Děje se tak pomocí úpravy této matice pomocí matematických metod, které jsou popsány v kapitole 3. Knihovna OpenCV.

Detekce objektů často probíhá v tzv. *real-time* režimu (v reálném čase). Tento způsob umožňuje detekovat objekty ve videu. Za jednu sekundu tak program detekuje objekty na několika desítkách snímků. *Real-time* detekce může být velmi náročná na výpočetní kapacitu.

V poslední době se na scéně počítačového vidění objevují i 3D kamery. Tento mimořádně sofistikovaný způsob využití senzorů přidává další rozměr k analýze prostředí. Řídící algoritmus dostává mimo matice pixelů i informaci o jejich vzdálenosti. Další data samozřejmě znamenají opět vyšší nároky na výkon počítače. Výsledky ale umožňují využití počítačového vidění v mnoha dalších aplikacích. Tato technologie je momentálně ve fázi vývoje. 3D kamery fungují podobně jako lidské oči. Mají k dispozici nejméně dvě čočky, které skládají dohromady jeden pohled. Podle odchylek jednotlivých částí obrazu je pak počítána jejich vzdálenost od kamery.

2.3 QR kódy a jiné nosiče informací

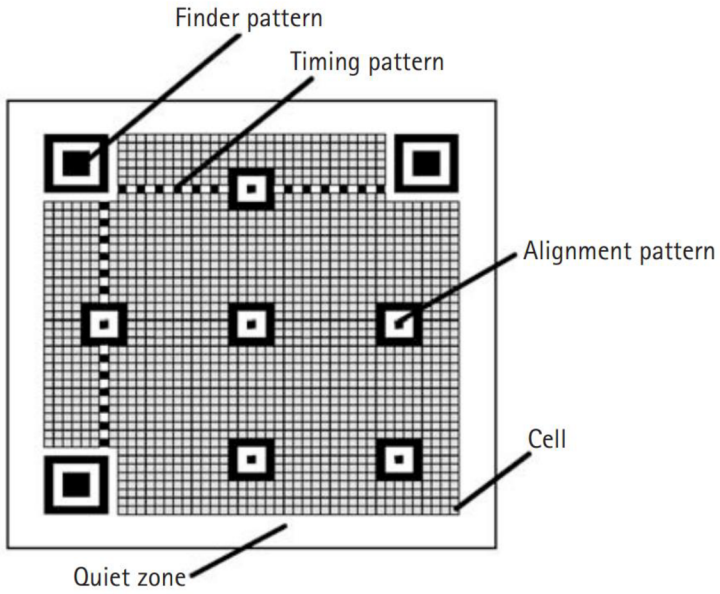
Nejrozšířenější způsob, jakým můžeme algoritmu počítačového vidění předat informaci, je použití čárových kódů. Tato metoda umožňuje velmi rychlé načítání informací a dokáže nést velké množství dat. Je jakousi alternativou prostého textu, který člověk čte z knihy. Tak jako člověk může číst knihu v několika jazycích, i počítač může rozpoznávat tištěné kódy v několika podobách. Způsoby vytváření kódů se v historii rychle vyvíjely. Nejpoužívanější z nich jsou dokonce standardizovány normou ISO. Čárové kódy slouží především k automatickému třízení. Kódy dělíme na 1D a 2D.

Mezi 1D kódy patří klasické liniové **čárové kódy**. Informace v těchto kódech je nesena v různě širokých černých čárách, které mají mezi sebou různě velké mezery. Tento

průběh je pak vyhodnocen přístrojem a dekódován podle příslušné normy. Tento způsob ukládání informací byl vyvinut pro obchodní řetězce [5], kde se čárové kódy využívají dodnes. Načítají se pomocí jednoduchých laserových čteček. V současnosti se nejvíce používají čárové kódy typu EAN-13 (European Article Number).

Obr. 2: EAN [6]

Dvourozměrné kódy, známé jako **QR kódy,** dokážou nést mnohem více informací než původní jednorozměrné. Jejich detekce už neprobíhá jen pomocí jednoduchých laserových čteček, je zapotřebí sofistikovanějších zařízení. QR kódy jsou standardizovány normou ISO/IEC 18004:2015. Existuje více druhů QR kódů. Podle vynálezce QR kódu patří mezi nejdůležitější QR kódy typ Model [7]. Mohou být různě veliké. QR kódy typu Model 2 se skládají ze čtverců *Finders* umístěných v rozích, určující orientaci natočení. Mezi nimi je pás střídajících se černých a bílých bodů pro opravu špatně načtených souřadnic. Menší čtverce *Alignment Pattern* jsou rozmístěny pravidelně napříč QR kódem a slouží pro snazší načítání kódu při různých pozorovacích úhlech. Zbytek QR kódu pak tvoří buňky, které jsou nositeli informací. [8]



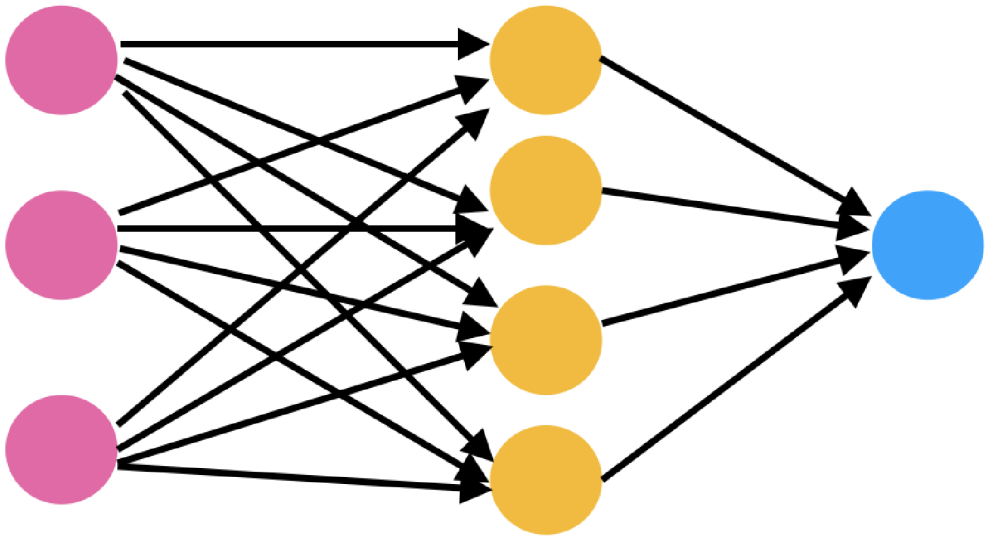
Obr. 3: Struktura QR kódu [8]

Buňky jsou díky přesně dané orientaci QR kódu vnímány jako bity, které nabývají hodnot 1 a 0 (černá a bílá). Pole těchto bitů jsou rozdělena do políček po osmi (aby dohromady dávaly jeden bajt). Tyto políčka mají napříč celým QR kódem přesně dané pořadí, aby byl zajištěn sled a informace byly zapsány a načteny stejně [9].

2.4 Umělá inteligence a neuronové sítě

Vedle klasických konvenčních způsobů detekce objektu v obrazu pomocí matematických metod a práci s maticemi se v poslední době masivně rozmáhají alternativní metody využívající umělou inteligenci. Umělá inteligence je taky matematickou funkcí. Liší se svou schopností samotvorby. Nejedná se tedy o algoritmické struktury, které by vytvářel člověk s pevně daným průběhem. Umělá inteligence vytváří program, který je často člověku neprůhledný. Na scéně detekce objektů sklízí umělá inteligence mnoho úspěchů. V praxi se jedná o hojně využívanou metodu, která má výhody především ve spolehlivém vyhledávání složitějších věcí.

Umělá inteligence se časem rozvětvila do mnoha oblastí. Nejen v detekci objektů se osvědčily neuronové sítě. Tato metoda byla inspirována chováním neuronů v lidském mozku [10]. Tyto jevy byly zjednodušeny a staly se matematickými modely, které slouží třeba pro identifikaci objektů na obrázku. Základem neuronové sítě jsou neurony.



Obr. 4: Model neuronové sítě [11]

Neurony (oranžové na obrázku 4) jsou navzájem propojeny vlákny, které mají určitou hodnotu mezi 0 a 1. Všechny tyto neurony v jedné řadě tvoří vrstvu. Neuronová síť může obsahovat více vrstev, takové síti říkáme hluboká síť a hovoříme o hlubokém učení. Růžová barva na obrázku označuje vstupní vrstvu neuronů, které nabývají taktéž hodnoty mezi nulou a jedničkou. Výstupních neuronů (na obrázku modré barvy) na konci může být více a signalizují výsledek. I tyto poslední neurony nabývají hodnoty 0 až 1.

Samotný princip neuronových sítí je překvapivě jednoduchý. Z obrázku, který je plně rozložitelný na matici čísel symbolizujících pixely (viz obrázek 1), vytvoříme řadu hodnot pixelů. Tato řada je první vrstvou neuronové sítě. Hodnoty na vstupu se znormalizují lineární aproximací na hodnoty mezi 0 až 1. Tyto hodnoty se s každým přilehlým vláknem vynásobí jeho hodnotou a výsledek se pošle do neuronu v další vrstvě tak, jak je šipkami naznačeno na obrázku 4. Hodnoty z předchozích vrstev se v každém neuronu sečtou a pomocí normalizační funkce se opět přetvoří na hodnotu mezi 0 a 1. Takto celý proces postupuje dál až do poslední výstupní vrstvy, kde se na neuronech objeví hodnoty, které symbolizují pravděpodobnost, se kterou neuronová síť přiřazuje vstup (maticový obrázek) výstupu (například člověk, pes nebo jiný předmět, který má za úkol síť rozpoznat).

Celkové vyhodnocení tedy závisí na hodnotách vláken a normalizační funkci. Zatímco normalizační funkce je pro celou neuronovou síť stejná (většinou se jedná o sigmoid [10]), hodnota vláken je různá a tvoří tak charakter i paměť celé neuronové sítě. Hodnota těchto vláken je získávána pomocí tzv. *backpropagation*, což je v podstatě zpětný chod popsaného principu [10]. Tomuto procesu říkáme trénování neuronové sítě. Neuronové síti ukazujeme obrázky i s informací o jejich zařazení a ona si zpětným procesem *backpropagation* vytváří hodnoty na vláknech mezi neurony. Takto vytrénovaná neuronová síť je pak schopna identifikovat i předměty, které nikdy předtím při tréningu neviděla.

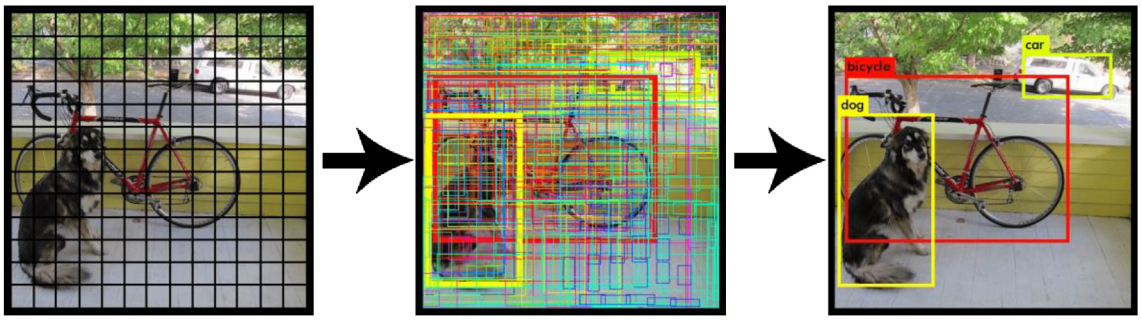
Neuronové sítě pro detekci objektů mohou být vylepšeny přidáním filtrů, které jsou blíže popsány v následující kapitole 3. Knihovna OpenCV. Tyto filtry upravují matice pixelů a mohou v nich najít například hrany, barvy nebo rohy. Filtry jsou vkládány mezi vrstvy neuronů. Neuronové sítě využívající těchto filtrů nazýváme *cellular neural network* (CNN) [10].

2.4.1 Knihovny strojového učení

Strojové učení je podoblast umělé inteligence umožňující strojům učit se ze zpětné vazby. Tyto metody patří mezi pokročilé algoritmy, proto se při jejich aplikaci využívají knihovny, které pomáhají vytvářet neuronové sítě. Knihoven existuje celá řada např. Keras, PyTorch nebo scikit-learn. V současnosti se však k těm nejpoužívanějším řadí knihovna **TensorFlow** od Google. Pomocí této knihovny můžeme snadno data upravovat (například normalizovat obrázky), stavět modely neuronových sítí a vyhodnocovat jejich účinnost. Pomocí knihovny TensorFlow může i člověk bez znalostí vyšší matematiky vytvářet neuronové sítě pro detekci objektů, analýzu textu nebo hlasu [12].

2.4.2 YOLO

K hledání a označování objektů na obrázku potřebujeme více než jen neuronovou síť. Vývojáři v současnosti své úsilí vynakládají na vyvíjení algoritmů pro detekci objektů na obrázku v reálném čase. Mezi nejpokročilejší algoritmy patří YOLO (You Only Look Once). Tato metoda rozdělí obraz na segmenty a ty pak hodnotí [13]. Proces a výsledek je znázorněn na obrázku 5.



Obr. 5: YOLO [14]

2.5 Aplikace v průmyslu a jiných odvětvích

Stroje v průmyslu jsou díky počítačovému vidění daleko více samostatné, rychlejší a nepotřebují pro svůj chod tolik lidského faktoru. Strojové vidění se v průmyslu využívá zejména při **kontrole kvality**. Jde o monotónní proces, který může kamera s algoritmem vykonávat mnohem rychleji než člověk. Počítač je schopný výrobky měřit, posuzovat tvary nebo celkový vzhled včetně barev. Největším úskalím těchto metod je světelné prostředí. Snímaný prostor musí být nasvěcován konstantním osvětlením [15]. **Čárové kódy** se v průmyslu využívají hojně zejména při třídění výrobků a logistice. Pomáhají vytvořit každému prvku v průmyslovém procesu svoje označení jednoduše čitelné pro stroje. Další rozšířenou aplikací v průmyslu je **kalibrace**. Robotická ramena využívají různých způsobů pro kalibraci. Kalibrace pomocí počítačového vidění je popsána v kapitole 3.5 Kalibrace. Používání strojového vidění zvyšuje podíl automatizace ve výrobě a posouvá průmysl do čtvrté revoluce. Ve čtvrté průmyslové revoluci se očekává vyšší samostatnost výrobních systémů a jejich vzájemná propojenost. Bez strojového vidění by tento koncept nebyl proveditelný.

Počítačové vidění se nevyužívá jen v průmyslu. V moderním světě nalézá své uplatnění stejně také tam, kde je zapotřebí lidské pozornosti a úsudku. V systému **bezpečnostních kamer** může počítačové vidění s využitím neuronových sítí vyhledávat podezřelé osoby, předměty nebo upozorňovat na nebezpečné situace. Stejně tak dobře fungují **kamery v autech**, kde upozorňují řidiče třeba na dopravní značky. K nejrozšířenějšímu využití počítačového vidění v životech lidí došlo před několika lety, kdy se v chytrých telefonech začala objevovat možnost **odemykání pomocí obličeje**. Tyto sofistikované algoritmy vytvořily z obličeje unikátní klíč. Využití této technologie se očekává i v dalších oblastech.

3 KNIHOVNA OPENCV

Open Source Computer Vision Library (OpenCV knihovna) je jedna z nejrozšířenějších knihoven pro počítačové vidění a strojové učení. Jedná se o *open source*, což znamená, že je zdarma a volně šiřitelná. Algoritmy OpenCV knihovny za uživatele řeší veškerou matematiku, která je potřebná pro úkoly počítačového vidění. Umožňuje i práci s 3D obrazem.

OpenCV knihovna je původně napsána v jazyce C++, ale dnes ji je možné použít i pro Python, Javu nebo třeba MATLAB. Nejčastěji je využívána pro *real-time* zpracování obrazu [16]. Jelikož se jedná o *open source*, kdokoliv se může stát vývojářem nebo dobrovolníkem a pomáhat při zdokonalování tohoto projektu. Celou věc dal do pohybu Gary Bradsky v roce 1999 ve společnosti Intel [17]. Knihovnu si podle odhadů stáhlo 18 milionů uživatelů mezi kterými jsou i soukromé společnosti, výzkumné skupiny a vládní organizace [16].

Tato kapitola je věnována popisu matematických a fyzikálních principů, které většina funkcí knihovny OpenCV pro svůj chod využívá. Protože jich knihovna obsahuje nepřeberné množství, byly vybrány ty nejdůležitější pro tuto práci.

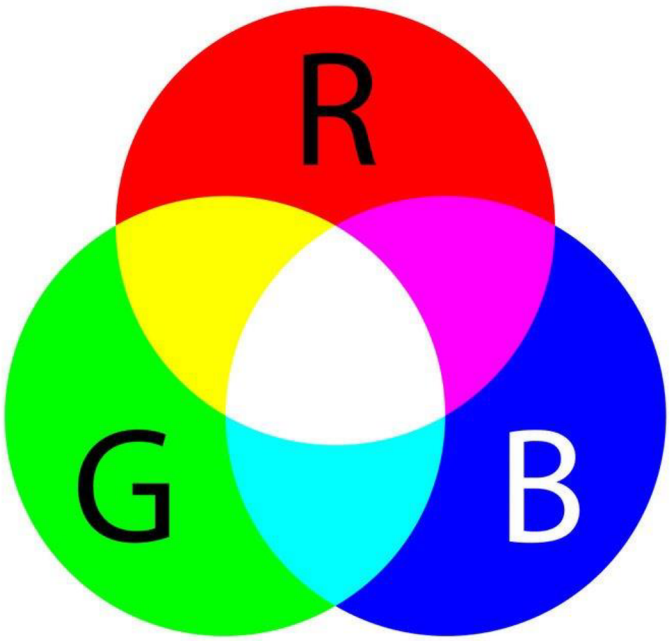
3.1 Základní funkce knihovny

Knihovna obsahuje řadu základních nástrojů pro práci s obrázky a videi. Knihovna umožňuje obrázky načítat, kreslit do nich, ořezávat a ukládat je. Lze vkládat i text s různými fonty, obrazce a geometrické tvary různých barev. Stejně tak v případě videa, které je bráno jako sled více obrázků za sebou. Každý obrázek knihovna vidí jako matici pixelů. Tyto pixely řadí v kartézském souřadném systému. Při spuštění kódu můžeme nastavit zobrazení okna například s videem nebo obrázkem. Knihovna umožňuje i zobrazení okna s posuvnými trackbary pro okamžitou změnu parametrů.

3.2 Barvy

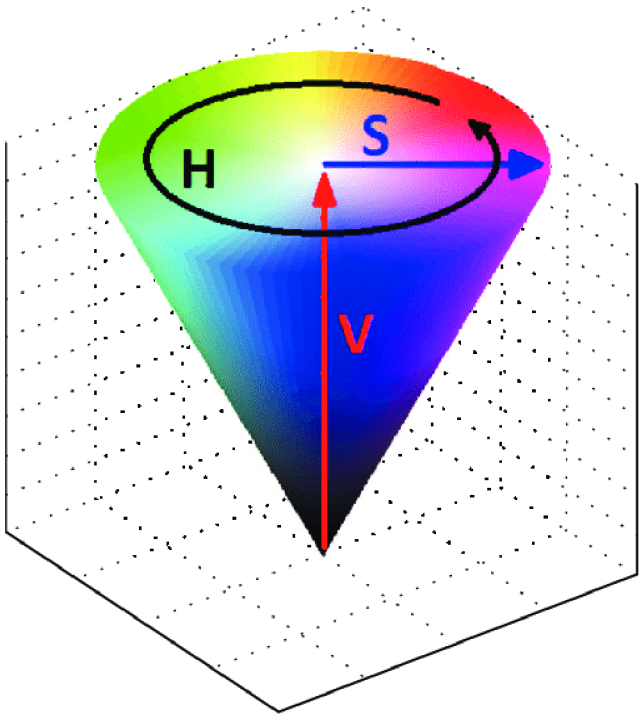
Jedním ze základních způsobů, jak v obrazu začít detekovat objekty, je odfiltrování určitých barev. Knihovna nabízí 150 metod pro definování barev [17]. Nejpoužívanější jsou RGB a HSV.

RGB chápe barvu jako mix tří základních barev – R = červená (red), G = zelená (green) a B = modrá (blue). Každý pixel obrázku je interpretován třemi hodnotami RGB. Podle procentuálního zastoupení těchto hodnot se pixel různě vybarvuje.



Obr. 6: RGB spektrum [18]

Při detekci objektů se však více pracuje s **HSV** barevným rozdělením. Do HSV si uživatel musí každý obrázek před barevným vyfiltrováním převést. HSV více odpovídá lidskému vnímání. H reprezentuje odstín (hue), S = sytost (saturation) a V = jas (value). Interpretace hodnot je zobrazena na obrázku 7.



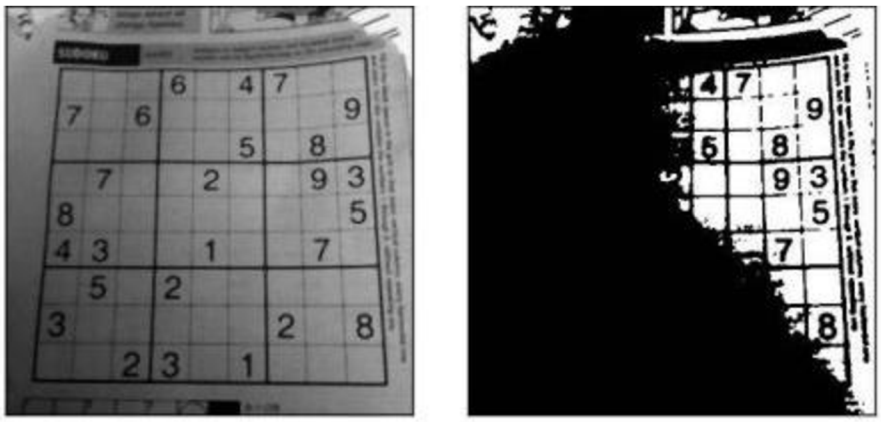
Obr. 7: HSV kužel [19]

Jak již bylo napsáno, hlavním důvodem, proč se v OpenCV zabýváme barvami, je jejich oddělení a následná detekce. To umožňuje funkce *cv2.inRange*. Vstupem funkce jsou maximální a minimální hodnoty HSV, které filtr propustí.

3.3 Filtry a příprava pro detekci

Než v obrázku bude cokoliv detekováno, je potřeba jej vhodně upravit. Mimo barevné filtrování existuje více způsobů, jak z obrázku separovat jiné části nebo obrázek k těmto účelům pozměnit. Tyto funkce matematicky upravují matici pixelů v obrázku. Využívají řadu vstupních parametrů, které řídí jejich intenzitu a můžeme je pomocí nich lépe ovládat. Existují desítky různých filtrů, následuje výčet a popis nejdůležitějších z nich.

Image Thresholding přepíše matici pixelů na nejvyšší nebo nejnižší hodnoty podle zvolené meze. V praxi to vypadá tak, že se z většiny černobílého obrázku stane obrázek, který obsahuje jen úplně černou nebo úplně bílou barvu. Tato funkce může zvýraznit objekty vysokých kontrastů (například černý text na bílém papíře), odstranit šum a objekty lépe detekovat. Pro většinu obrázků je ale tato metoda málo, protože mohou zaniknout důležité části při různém nasvícení. Matematicky je tato metoda velice jednoduchá, pouze nahrazuje pixely o určité hodnotě jinými (binárními, tedy jedničkami a nulami). Existuje více variant. Nejběžnější je vidět na obrázku:



Obr. 8: Thresholding [17]

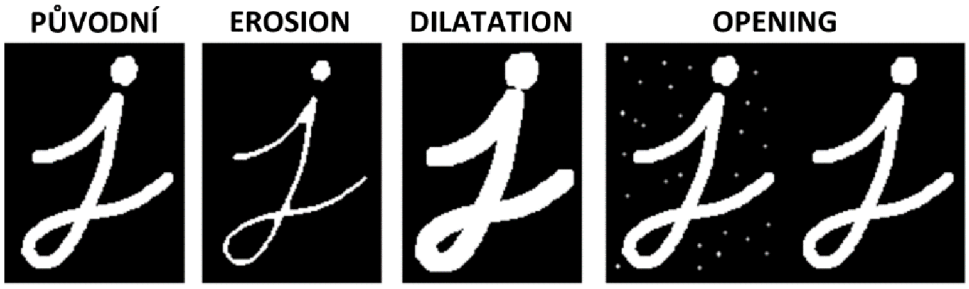
Smoothing Images rozmaže obrázek. Tato funkce může pomoci při odstraňování šumu. Kolem každého pixelu vznikne prostor, ve kterém jsou sečítány hodnoty všech pixelů. Tato hodnota je pak vydělena počtem sečtených. Jde tedy o průměrnou hodnotu okolo každého pixelu. Velikost pole určuje vstupní parametr a ovlivňuje, jak moc bude obrázek rozmazaný. Matematicky se dá funkce popsat takto:

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

Obr. 9: Smoothing [17]

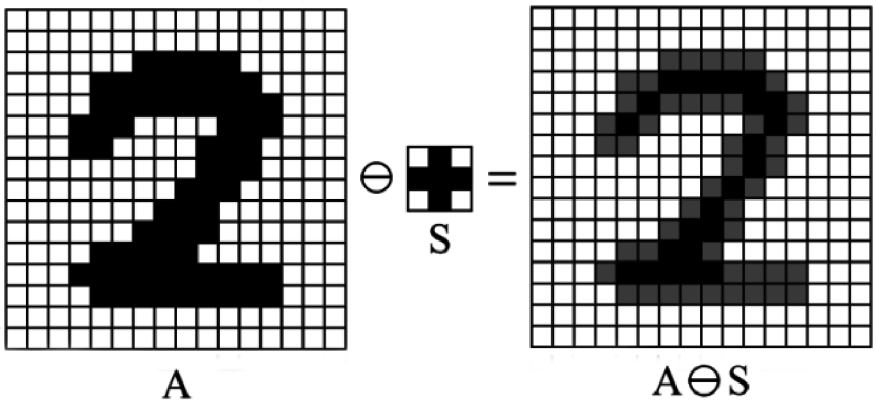
Pomocí podobné funkce *GaussianBlur* můžeme nastavit velikost oblasti v každém směru jinak. Tento nástroj je velmi efektivní v odstraňování šumu [17].

Morphological Transformations pracuje s binárním obrazem, tedy buď s černobílým nebo jiným dvoubarevným. Metoda vnímá šířku obrazců vůči okolí. Šířku je pak schopná pomocí řady funkcí upravovat. Mezi tyto funkce patří například *Erosion* (obrazec zužuje), *Dilatation* (obrazec rozšiřuje), *Opening* a *Closing* (ponechávají jen obrazce určité šířky). Funkce jsou vidět na obrázku 10:



Obr. 10: Morphological Transformations – upraveno [17]

Funkce jsou velice efektivní při odstraňování přebytečného šumu a při zvýrazňování dominantních objektů. Předtím ale musí být obrazec již upraven jinými filtry, aby byl v binárním formátu. Matematicky funkce pracují velice jednoduše – pomocí tzv. *Structuring elementu*, což je matice jedniček a nul uspořádaných většinou do kříže (velikost i tvar závisí na vstupních parametrech). Tento *structuring element* projíždí matici obrázku bod po bodu a jestliže zaznamená, že se některá část jeho matice dostala do zabarvené oblasti a jiná část zase ne, vyhodnotí, že je na okraji obrazce. Pokud pracuje jako funkce *Erosion*, tak pixely v této detekované oblasti přemění a obraz se tak po okrajích zmenší. Takto funkce projede celý obrázek a bez ohledu na to, jestli je okraj vertikální či horizontální, jej zúží nebo rozšíří. Princip je znázorněn na obrázku 11:



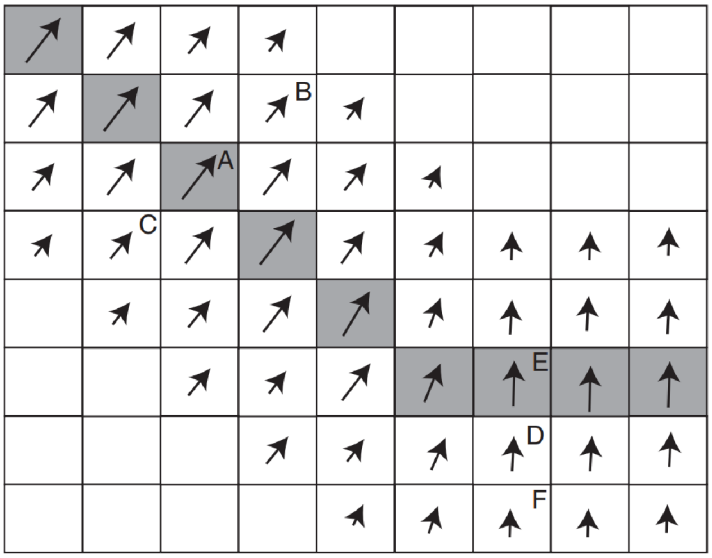
Obr. 11: Princip structuring elementu [20]

Podobného principu se může využívat i v jiných metodách. Pokud upravíme *structuring element* asymetricky, výsledkem může být úprava jen horizontálních nebo vertikálních čar. Tohoto principu využívá funkce **image gradiend.** Vstupními parametry

se upravuje citlivost na okraje v různých směrech. Používají se buď samostatně, nebo společně prostřednictvím Laplaceovy derivace [17]. *Structuring element* je matematicky chápán jako detektor změny. I v nebinárním obrázku lze pomocí této derivace najít okraje, u kterých bude detekována dostatečná změna nějakého parametru. Ve více směrech pak mluvíme o gradientu. V obrázku se v každém bodě vypočítá gradient. Matematicky je gradient v obrázku reprezentován následující rovnicí:

$$\nabla F = \left[\frac{\delta F}{\delta x}, \frac{\delta F}{\delta y}\right] \tag{1}$$

kde každá složka reprezentuje změnu hodnoty v jenom směru. Pokud by byla změna jen v jednom z nich, druhá hodnota gradientu by se rovnala nule. Gradient je tedy vektor. Na obrázku 12 jde vidět, jak po okrajích vypadá hodnota a směr gradientu. Pro počítač pak není složité zobrazovat jen ty pixely, kde hodnota nebo směr gradientu odpovídá určitým hodnotám.



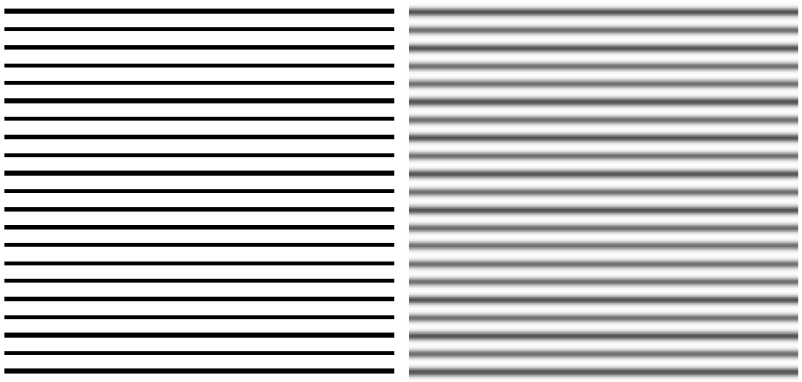
Obr. 12: Gradient [3]

V tomto případě program detekoval okraj (šedá políčka) podle velikosti gradientu (velikost v bodech A a E je hodnota odlišná od ostatních).

Tímto se dostáváme k jednomu z nejdůležitějších filtrů pro strojové vidění, detekci okrajů. Člověk detekuje okraje ve svém vizuálním světě automaticky. Právě podle okrajů pak rozlišuje objekty. Jak jsme si ukázali, pomocí gradientů a matic počítač rozpoznává ty své. Mezi nejpoužívanější funkci pro filtrování okrajů patří **Canny Edge Detection.** Tento algoritmus byl vyvinut už v roce 1986 [17]. Algoritmus nejprve vyčistí obrázek od přebytečného šumu, který by mohl mást při pozdější detekci okrajů. Následně vypočítá gradienty v každém bodě. Tento filtr vytvoří úplně nový obraz, kde ponechá jen pixely s největší hodnotou gradientu (binární obraz). Rozpětí je opět možno měnit pomocí vstupního parametru. Výsledkem je bílý obrys všech okrajů na černém pozadí. Tento filtr je obzvláště užitečný pro následnou detekci objektů v obrazu.

Obr. 13: Canny Edge Detection [21]

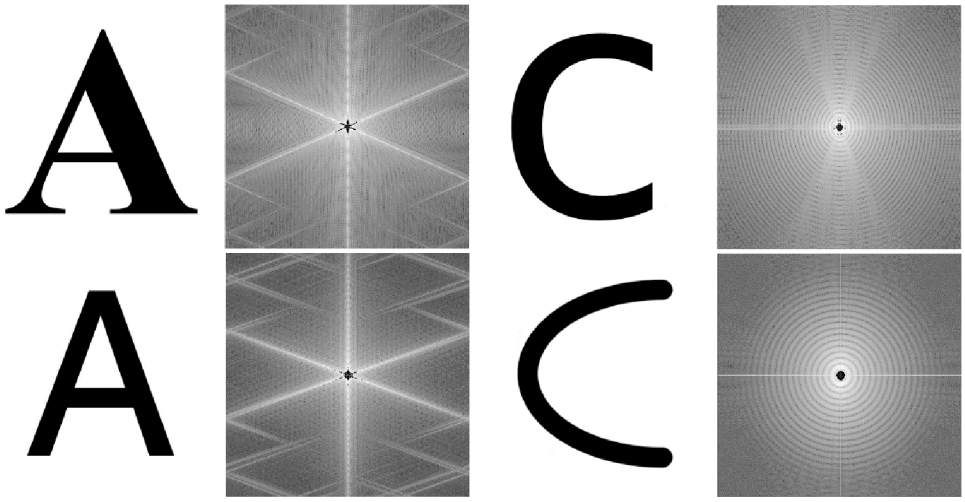
Složitějším způsobem, jak upravovat obrázky, je **Fourierova transformace**. Tento nástroj je využíván hlavně při detekci jednoduchých objektů. Fourierova transformace je matematická metoda, která se využívá i v jiných oblastech. Matematicky se dá Fourierova transformace vysvětlit jako zjišťování frekvence obrazu [22]. Obecně Fourierova transformace přeměňuje signál (funkci) s diskrétním rozložením na periodický v určitém rozmezí. Zpětnou Fourierovou transformací lze tento proces udělat i naopak. Ve strojovém vidění je pak běžné transformovat obraz ve dvou rozměrech. Výsledek se pak nazývá frekvence obrazu. Na následujícím obrázku jsou dva příklady obrazu s odlišnou frekvencí.



Obr. 14: Frekvence obrazu [22]

Obrázek vlevo ukazuje rychlé, téměř nespojité změny ve vertikálním směru. Frekvence je chápána jako velká. Na druhém obrázku jsou změny barev spojitější

a frekvence je tak menší. Pomocí Fourierovy transformace lze pracovat s touto frekvencí a zostřit tak obraz. Většinou se ale používá přesně naopak pro rozostření obrazu. Obraz se převede do frekvenčního odrazu a ten zase zpět do původního obrazu. Tímto se obraz rozostří a zmizí některé nečistoty. Fourierova transformace se dá použít i při detekci objektů [22]. Při pohledu na obrázek 15 vidíme, že vizualizace písmen má určitý frekvenční odraz.



Obr. 15: Fourierova transformace obrazu – upraveno [22]

Fourierova transformace je citlivá na tvary. V obrázku vidíme, že podobným tvarům přiřazuje podobné frekvence. Zatímco pro počítač může být složité hledat souvislosti v obrázcích s písmeny, lépe je může vidět v jejich frekvenčních odrazech. Detekce písmen se pak děje na odrazech s jejich frekvencí. Tohoto principu se využívá v každém z výše jmenovaných filtrů. Filtr nejprve obrázek vhodně upraví a až upravený odraz vstupuje do detektorů, kterým je věnována další část této kapitoly.

3.4 Detekční funkce

Jakmile máme obrázek vhodně upravený, můžeme použít funkce, které budou hledat různé podněty a vypisovat jejich souřadnice.

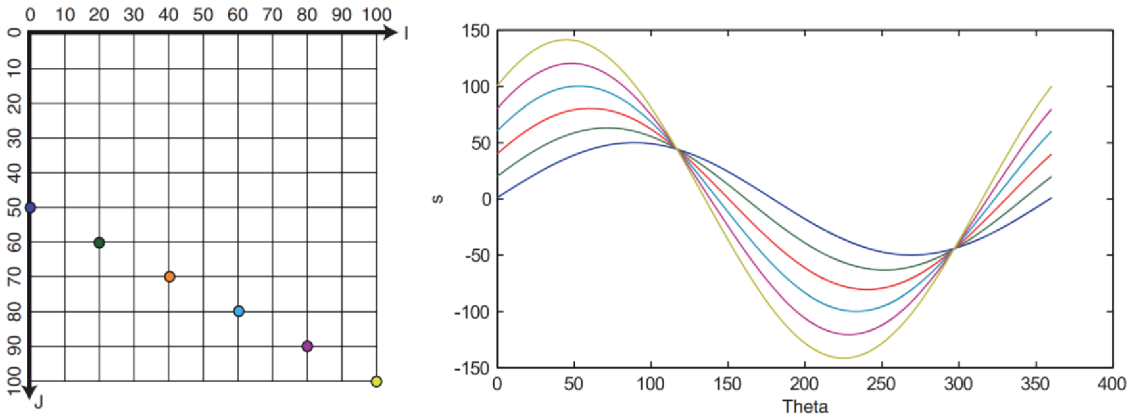
Hough Line Transform je funkce pro hledání čar. Před použitím funkce je nutné, aby byl obraz převeden do binárního odrazu krajů. Nejlépe pomocí nástrojů pro detekci okrajů. Funkce spadá do rodiny *Hough Transform*. Tyto funkce vyhledávají v obrazu jakékoliv tvary, pokud jsou definovatelné jako matematické funkce [17]. Detekce čar pracuje s následující rovnicí pro přímky:

$$s = i.cos(\vartheta) + j.sin(\theta) \tag{2}$$

kde „s“ je ortogonální vzdálenost od přímky, ϑ je úhel mezi přímkou a I osou.

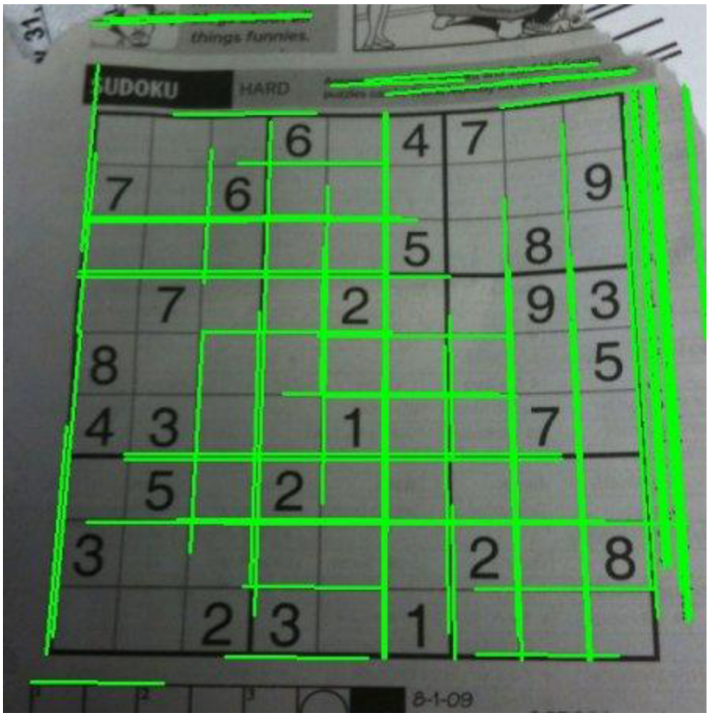
S klasickou rovnicí přímky pracovat nemůže, protože by nedokázala popsat například svislou čáru [3]. Funkce každý bod okraje v binárním obrázku prošetří a pomocí změny parametru theta vypočítá všechny hodnoty „s“ [3]. Počítá je právě tam, kde

hodnota theta (úhel) přímky je shodná s některým z dalších sousedních bodů na obrázku. Vypočtené hodnoty pak poskládá do grafu theta – s.



Obr. 16: Hough Line Transform [3]

V tomto grafu funkce vyhledá všechna lokální maxima. Pomocí těchto maxim je pak schopná dopočítat souřadnice prvních a posledních bodů na detekované přímce v obrázku. Výstupem funkce jsou tyto souřadnice. I když je tato metoda využívána, existuje ještě podobná, zjednodušená. Algoritmus popsaný výše je velice náročný na výpočty, protože propočítává bod po bodu a čáru detekuje pro každé dva body, které jsou v jedné linii. Proto byla vyvinuta metoda *Probabilistic Hough Transform.* Tato metoda funguje úplně stejně, jen vybírá náhodně určitý počet pixelů v binárním obrázku (ne všechny). Tento počet bývá dostatečný pro detekci všech čar a rychlost algoritmu je vyšší. Výsledek této metody je zobrazen na obrázku 17:



Obr. 17: Probabilistic Hough Transform [17]

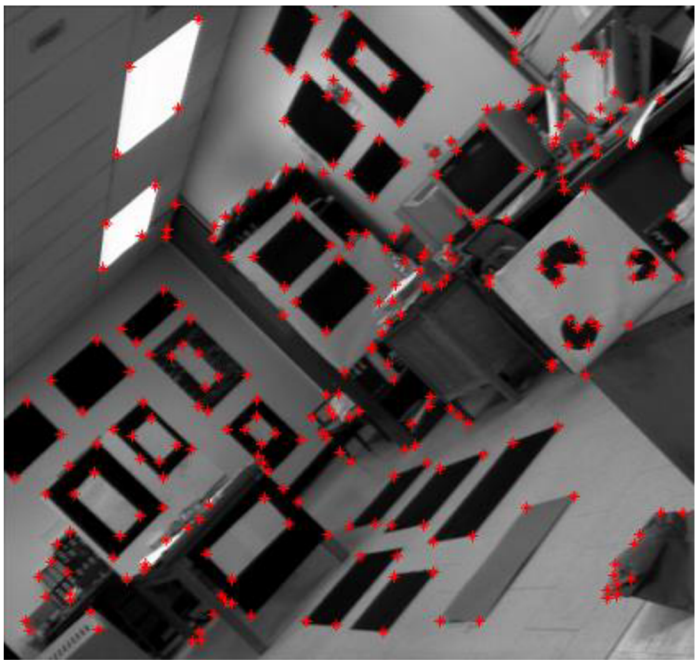
Na velice podobném principu funguje detekční funkce **Hough Circle Transform**, která detekuje kruhy. Rovnice pro vyhledávání kruhů v obrázku vypadá takto:

$$(\mathrm{i} - \mathrm{a})^2 + (\mathrm{j} - \mathrm{b})^2 = r^2 \tag{3}$$

Algoritmus je obdobný jako u detekce čar. Funkce bod po bodu propočítává hodnoty podle své rovnice a ukládá je do grafu. Nevýhodou je, že musí vracet tři parametry: dvě souřadnice středu a poloměr kruhu. Díky tomuto graf není dvou, ale tří rozměrný. Celý proces je velice výpočtově náročný. Vstupem funkce je mimo obrázku i parametr pro citlivost. Kvůli výpočtové náročnosti se pro detekci kruhů mohou použít jiné metody založené na gradientech [17]. Příklad detekce kruhů je na následujícím obrázku 18:

Obr. 18: Hough Circle Transform [23]

Jednou z nejužitečnějších funkcí pro detekci objektů je **Corner Detection** neboli detekce rohů. Jak už název napovídá, tato funkce detekuje rohy. Pomocí této funkce lze identifikovat spoustu různých objektů, protože jsou to právě rohy, co dává předmětům tvar. Rohy jsou důležitými body v obrázku, ve kterých najdeme dva silné gradienty. Příklad je zobrazen na obrázku 19.



Obr. 19: Corner Detection [24]

S matematickou formulí pro tuto metodu přišel Chris Harris a Mike Stephens v roce 1988 [17]. Jednoduše by se dalo říct, že funkce hledá ve všech místech obrázku změnu intenzity ve všech směrech. Matematický zápis vypadá takto:

$$\mathrm{E(u,v)} = \sum_{x,y} w(x,y)\,[I(x+u, y+v) - I(x,y)]^2 \tag{4}$$

kde $w(x,y)$ je funkce obrazu
$I(x+u, y+v)$ je posunutí intenzity
$I(x,y)$ je intenzita

Funkce $\mathrm{E(u,v)}$ je pro správnou detekci rohů maximalizována pomocí Taylorova rozvoje a derivacemi upravena [17]. Výsledná úprava a další potřebné rovnice:

$$\mathrm{E(u,v)} \approx [u\ v]\, M \begin{bmatrix} u \\ v \end{bmatrix} \tag{5}$$

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \tag{6}$$

kde I_x a I_y jsou derivace obrazu ve směrech x a y. Posledním krokem je rovnice:

$$R = \det(M) - k(trace(M))^2 \tag{7}$$

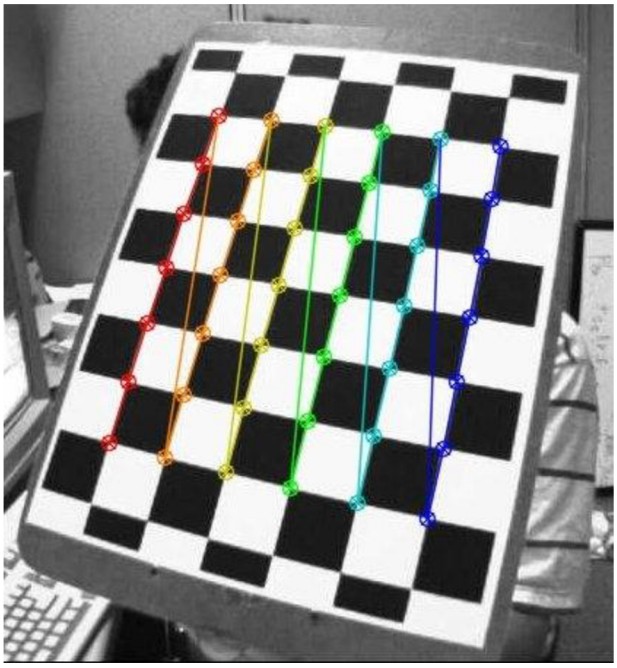
kde $\det(M) = \lambda_1 \lambda_2$, $trace(M) = \lambda_1 + \lambda_2$, lambdy jsou vlastní hodnoty M. Vypočtením a porovnáním λ_1 a λ_2 zjistíme, zda je ve vyšetřovaném místě roh. Pokud jsou lambdy velikostí srovnatelné a R je velké číslo, tak se v oblasti roh nachází [17]. Citlivost funkce se dá jako u všech ostatních metod upravit pomocí vstupních parametrů. Metod pro hledání rohů je více. Principiálně jsou si podobné. Harrisova je však jedna z prvních a stále je hojně využívána. Mezi další metody patří Moravec, FAST Corner Detection nebo SIFT.

Hojně využívanou detekční funkcí pracující s rohy je funkce pro detekci tzv. **contourů**. *Contoury* jsou sady detekovaných rohů v binárním obrázku, které jsou seskupeny podle toho, k jakému objektu patří. Díky tomu můžeme například podle počtu *contourů* detekovaných pro určitý objekt odhadnout jeho tvar. *Contoury* jsou velmi užívané pro detekci objektů.

3.5 Kalibrace

Knihovna OpenCV umožnuje kalibrovat obraz vůči kameře. Kalibrace je při *real-time* režimu velice důležitá. Podává informace o poloze kamery vůči okolnímu světu. Nejčastěji se tak děje pomocí vypočtených konstant. Díky kalibraci mohou stroje pracovat přesněji. Při velmi přesných aplikacích nemusí být metoda kalibrace pomocí strojového vidění vhodná, protože může docházet k odchylkám díky geometrii kamery [17]. Kamera může vlivem zkreslení zachytit rovné hrany jako zakřivené, to pak mění kalibrační konstanty a nedostává se přesnému kalibrování. OpenCV umí pomocí matematických nástrojů tyto odchylky spočítat. Veškerá přesnost závisí i na kvalitě obrazu z kamery.

Pro kalibraci je vhodné použít předmět, obraz nebo vzor, u kterého známe přesné rozměry a má na sobě body, které jsou pro počítač lehce zaměřitelné. Nejčastěji se používá šachovnice. Kamera vyfotí šachovnici z několika úhlů z různých výšek. Na každém obrázku si na šachovnici vykreslí následující prvky:



Obr. 20: Šachovnice [17]

S body jejich polohy a s informací o poloze při focení ve všech případech pak matematicky spočítá všechny odchylky. Do programu je třeba zadat i reálnou velikost čtverce. Uvidí-li kamera čtverec na šachovnici na jednom snímku v 10 pixelech a na

druhém, kdy se kamera vzdálí o určitou vzdálenost, ve 20 pixelech, je matematicky jednoduché dopočítat, jakou vzdálenost kamera urazila. Díky možnosti porovnání stran čtverců z různých úhlů můžeme takto kalibrovat i natočení (například robotického ramene).

V průmyslových aplikacích se využívají metody kalibrace zvané **Hand Eye Calibration**. Pomocí těchto metod se kalibrují robotická ramena například pro přemísťování různých předmětů. Metody slouží pro sled souřadného systému robota, kamery a okolního světa. Existuje jich více, na výzkumných portálech najdeme nespočet odlišných matematických modelů, které kalibrují souřadné systémy robotů *pomocí Hand Eye Calibration*.

4 PŘEHLED POUŽITÉ TECHNIKY

Kapitola obsahuje výčet nejdůležitější techniky, která byla v práci použita. V každé podkapitole je obecný popis a rešerše zařízení, představení společnosti, která produkt vyrobila, a technický popis konkrétního modelu.

4.1 Kamera Logitech

Kamery jsou zařízení pro snímání obrazu. V minulosti kamery zaznamenávaly obraz na fyzické médium jako byl například film. V současnosti používáme převážně kamery digitální, které převádí obraz na jedničky a nuly. Obraz do kamery přichází skrze objektiv, následuje soustava čoček pro usměrnění obrazu na elektronický senzor. Ten je citlivý na světlo a pomocí fotochemických reakcí v jednotlivých buňkách převede pomocí elektrického náboje obraz do digitálního souboru [25]. Čím více těchto buněk čidlo má, tím více pixelů zaznamená. Počty pixelů se pak měří rozlišení kamer podle jednotky megapixel. Tato jednotka ale nevypovídá o kvalitě fotografie, protože více megapixelů znamená, že na buňky dopadne méně světla a obraz degraduje. Proto jsou kamery často charakterizovány i parametry jako clona (propustnost světla), čas snímání jednoho snímku, citlivost buněk na světlo a velikost buněk. Kvalita snímání tedy závisí na celkovém zpracování. Nejmodernější kamery disponují i možnostmi jako je automatické zaostřování, pracují se světlem v různě osvětlených prostředích apod.

Logitech je švýcarská společnost založená v roce 1981 [26]. Zabývá se vývojem a výrobou elektronických zařízení, převážně periferiemi pro osobní počítače. Dnes má na světovém trhu velký význam a řadí se k mezinárodním společnostem. Pyšní se mnoha úspěchy, například prvenstvím ve vývoji bezdrátových myší.

Webkamera C930E je zařízení určené pro videokonference. Výrobce označuje model jako *Business Webcam*, což znamená, že je určen hlavně pro nahrazení lidského kontaktu v pracovní a obchodní sféře. Výrobce poskytuje tříletou záruku.

Tab. 1: Technické specifikace kamery [27]

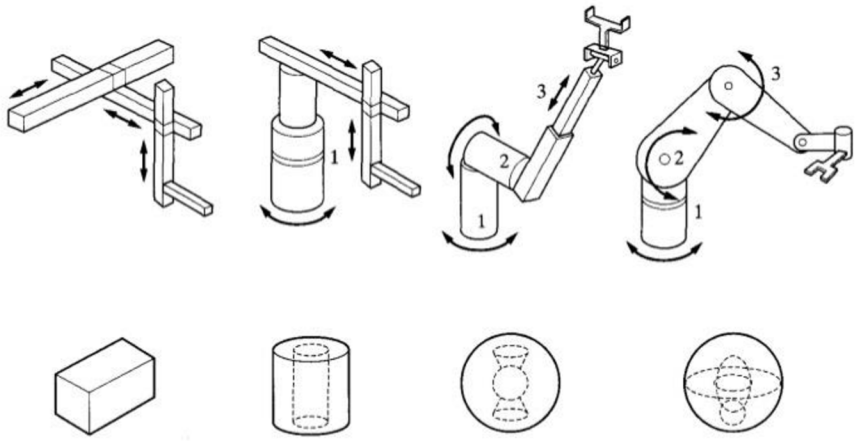
zařízení	webkamera
výrobce	Logitech
označení	C930e
uvedeno na trh	2013
cena	3 829 Kč
rozlišení	Full HD, 1080p, 1920×1080
FPS (frames-per-second)	30
úhel záběru	90°
digitální zoom	4x
kabel	USB 3.0

Obr. 21: Kamera Logitech [27]

4.2 Robotické rameno ABB

Robotická ramena se v poslední době objevují čím dál častěji v automatizovaných výrobních linkách. Díky nim se v průmyslu vyrobí více zboží, které je zároveň kvalitnější. Takto pokročilému průmyslu říkáme Průmysl 4.0 nebo čtvrtá průmyslová revoluce. Robotická ramena se nevyužívají jen v průmyslu při výrobě, ale například i v chirurgii nebo logistice. Robotické rameno nahrazuje lidskou paži. Má několik pohybových os (kloubů) pro vysoký stupeň použitelnosti. Záleží pak už jen na nástroji (efektoru), který je umístěn na konci, k čemu bude robot určen. V průmyslu se používají efektory na přemísťování objektů (přísavky nebo čelisti), svařovací hubice, měřící nástroje a další. Pojem kolaborativní robotika označuje roboty, které spolupracují s člověkem [28]. Typicky jsou to ti výrobní, kde může robot například přidržovat těžká břemena a na pokyn přemístit. Tito roboti se musí vyznačovat vysokou mírou bezpečnostních prvků. Bezpečnost je u výrobních robotů velice důležitá, protože kombinace rychlosti jejich pohybu, celkové hmotnosti ramen a síly pohonu dělají z robota nebezpečný stroj, který musí být řádně zabezpečen. Po dobu jeho činnosti se ve vytyčeném prostoru nesmí pohybovat žádná osoba (výjimkou bezpečných kolaborativních robotů, ti mají řadu senzorů a regulují rychlost pohybu).

Důležitých parametrů u výrobních robotických ramen je několik. Prvním je jejich velikost. Vyrábí se od řádu centimetrů po několik metrů. Druhým kritériem je množství pohybových členů a jejich uspořádání (konfigurace).



Obr. 22: Konfigurace robotických ramen [29]

Na obrázku 22 vidíme různé konfigurace, které udávají další důležitý parametr, a to pracovní prostor. Ten je vidět pod každým znázorněným robotem na obrázku. Důležitou roli hraje i celková únosnost robota. Další parametry jako cena, podporovaný software, komunikace apod. záleží na konkrétním výrobci.

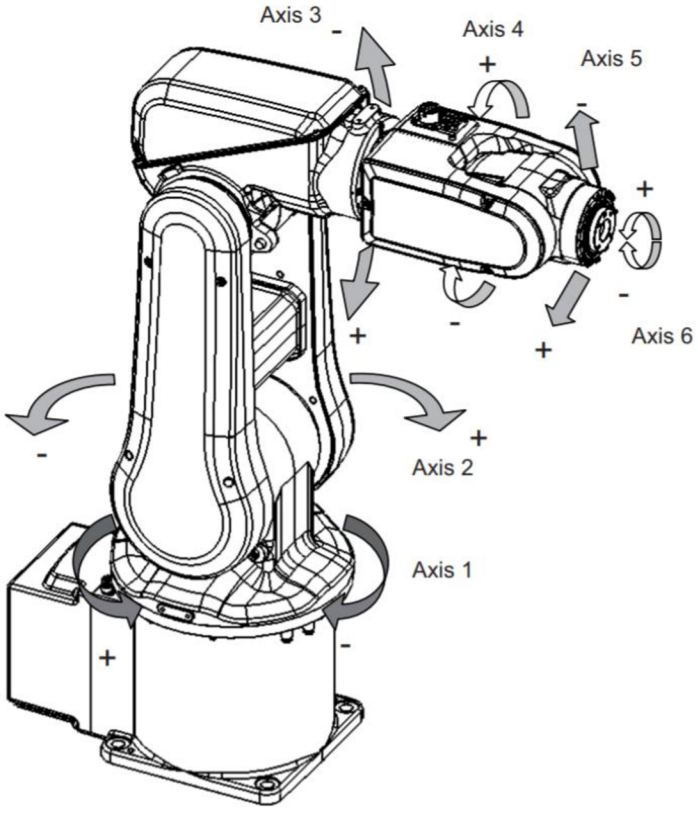
ABB je švýcarsko-švédská firma mezinárodního významu založená v roce 1988. Společnost ABB se řadí k největším konglomerátům světa. Středem jejího zájmu je výroba robotů a robotických systému pro automatizaci a energetiku. V Česku působí firma ABB od roku 1992 a má tu výzkumná i výrobní centra. Jako každý velký konglomerát je i ABB dělena do divizí podle oblasti zaměření. Divizí je pět a jsou to: **Electrification** (Elektrizace – divize se zabývá nízkonapěťovými prvky jako jsou zásuvky, kabely, ovladače apod.), **Industrial Automation** (Průmyslová automatizace – druhý největší světový hráč, poskytuje veškerá automatická řešení.), **Motion** (Pohony – divize je největším dodavatelem elektrických pohonů světa, vyrábí elektrické motory, serva, převodovky, generátory atd.), **Robotics & Discrete Automation** (Roboti a automatizace – divize se zaměřuje na chytrá řešení v průmyslu, vyvíjí umělou inteligenci a další pokroková řešení.) a **Power Grids** (Elektrické sítě – divize poskytuje automatizační prostředky a servis v energetice, dále pak transformátory, měniče a jiná vysokonapěťová řešení.) [30].

Česko patří k jedné ze sedmi zemí světa, kde divize firmy ABB Robotics provádí svůj výzkum a vývoj [30]. ABB má pro své roboty i vývojové prostředí ABB Robot Studio, ve kterém se nastavují pochody robotických ramen. Vyvíjí i jiné softwary, které slouží pro nejrůznější aplikace jako jsou např. Arc Welding Software, Plastic Software nebo 3D Printing Software. Mimo robotických ramen a softwarů vyrábí divize i kontrolery označované jako IRC. Důležitou součástí společnosti je B+R. Dříve samostatná rakouská firma byla v roce 2017 koupena ABB a připojena k divizi Robotics. Zabývá se výzkumem, vývojem a výrobou kontrolních prvků pro automatizaci.

Robotické rameno IRB 120 je malý šestiosý průmyslový manipulátor. Výrobce uvádí jeho využitelnost zejména při manipulaci s materiálem, montáži a kontrole kvality ve výrobních procesech. Je to jeden z nejmenších robotů od společnosti ABB.

Tab. 2: Technické specifikace robota [31]

zařízení	robotický manipulátor
výrobce	ABB Robotics
označení	IRB 120
uvedeno na trh	2009
cena	360 000 Kč
počet os	6
kontroler	IRC5
nosnost	3 kg
manipulační úhel	250°
maximální dosah	580 mm
napájení	200–600 V

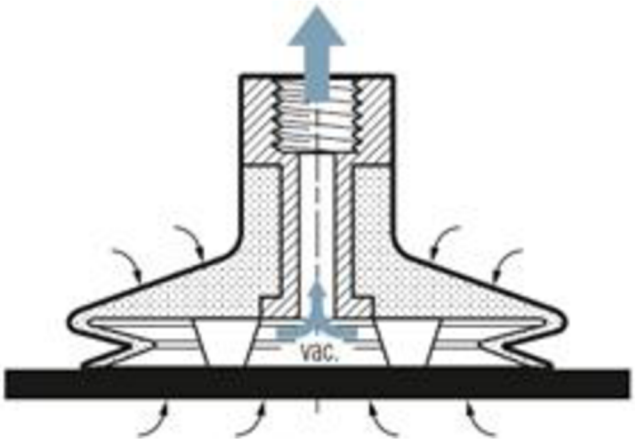


Obr. 23: Robot ABB [31]

4.3 Vakuová přísavka SMC

Nástroj, který se upíná na koncovou část průmyslového manipulátoru, se nazývá koncový efektor. Koncových efektorů máme celou řadu a dělí se na manipulační, kombinované, technologické a speciální. Manipulační efektory slouží především k přemísťování předmětů. Příkladem jsou přísavky a čelisti. Mezi méně běžné můžeme zařadit například magnetický efektor. Technologické efektory vykonávají některou z technologických operací, jako jsou například obrábění, vrtání, svařování nebo řezání. Díky preciznímu vedení jsou velice přesné. Kombinované efektory zvládají více operací. Speciální jsou pak navrhovány pro specifické výrobní operace.

Vakuové přísavky musí být napojeny na soustavu vzduchotechniky. Princip jejich činnosti je velice jednoduchý. Vývěva v soustavě vzduchotechniky vytvoří podtlak v okruhu, který je napojen na efektor. V prostoru mezi přísavkou a uchopovaným předmětem vznikne podtlak. Gumová přísavka na konci efektoru se zdeformuje a objekt se k ní přitlačí přísavnou silou, která je závislá na velikosti podtlaku, tuhosti přísavky a velikosti kontaktní plochy. Síla je způsobena působením okolního atmosférického tlaku na podtlak uvnitř přísavky [32].



Obr. 24: Vakuová přísavka [32]

Firma **SMC** (Sintered Metal Corporation) je japonská korporátní společnost založená v roce 1959. Společnost je největším dodavatelem pneumatického vybavení na světě. Společnost má téměř 20 000 zaměstnanců napříč celým světem a pokrývá 30 % světového trhu s pneumatickými a automatizačními prvky [33]. Mezi produkty patří mimo vakuové přísavky i pneumatické písty, senzory, elektrické pohony a mnoho dalšího. Firma se pyšní 12 000 produkty. Poslední středoevropskou pobočku založila firma v ČR. Česká divize SMC Industrial automation CZ s.r.o. má centrálu v Brně.

Koncový **vakuový efektor ZPR 20UNK10-06-A10** je k robotovi připevněn závitem. Vývod vzduchu je ze strany těsně nad přísavkou. Přísavka není rotační, ale výrobce vyrábí stejné přísavky, které se otáčet mohou. Zakoupit je možné ve čtyřech různých materiálových provedeních a mnoha průměrech.

Tab. 3: Technické specifikace přísavky [34]

zařízení	koncový vakuový efektor
výrobce	SMC Corporation
označení	ZPR 20UNK10-06-A10
cena	856 Kč
Průměr vakuového vstupu	6 mm
Materiál přísavky	NBR
Průměr přísavky	20 mm

Obr. 25: Přísavka SMC [34]

4.4 3D tiskárna Průša

3D tisk je moderní technologie výroby, která z 3D digitálního modelu vytváří reálnou součástku [35]. Nejpoužívanější způsob je FFF (Fused Filament Fabrication) a funguje tak, že se postupně nanáší jednotlivé vrstvy modelu, rozdělené po několika desetinách milimetru. Nanášení je prováděno tryskou. Této technologii se někdy říká aditivní výroba nebo Rapid Prototyping. V posledních deseti letech se 3D tisk neustále vyvíjí. Vývojáři se zaměřují na materiály a tiskárny samotné [35]. Díky jejich jednoduchosti je možné takovou 3D tiskárnu mít i doma. Mezi nevýhody pak řadíme hlavně rychlost tisku. 3D tisk se nevyužívá jen k tisku prototypů a jednokusových zařízení, ale i v sériové výrobě nebo tisku velice důležitých předmětů, jako je například náhrada lidského kloubu. 3D tiskem si můžeme vytisknou i náhradní díly. Modeláři využívají 3D tisku k výrobě originálních součástí, které si sami navrhnou. Návrh modelu není složitý, provádí se v běžných 3D softwarech. Nejpoužívanějšími materiály jsou nejrůznější typy plastů. Ty jsou navinuty na cívku, odkud jsou extrudovány (vytlačovány) pomocí zahřáté trysky na

podložku. Tiskne se však i z jiných materiálů, jako jsou kovy, dřevo, dokonce i čokoláda. Konstrukcí tiskáren a technologií nanášení materiálu je více.

Prusa Research je česká firma vyrábějící 3D tiskárny. Je jedním z největších výrobců světa. Založil ji v roce 2012 Josef Průša. Ve společnosti pracuje přes 400 lidí a měsíčně vyrobí více než 6 000 tiskáren [36]. Pyšní se statutem nejrychleji rostoucí technologické firmy ve střední Evropě [36]. Firma vznikla původně z nadšení zakladatele pro 3D tisk. Její nejrozšířenější produkt je stále typ Prusa i3, který je *open source*, to znamená, že jej mohou lidé napříč celým světem upravovat, vylepšovat, šířit a používat. Tiskárny Průša se staly symbolem pro hobby 3D tisk. Nejkurióznější věcí ve firmě je její Farma 3D tiskáren. Tato Farma je pojmenování pro sériovou linku na výrobu 3D tiskáren, složenou z 3D tiskáren. Na obrázku 26 je zakladatel Josef Průša v prostorách Farmy.

Obr. 26: Farma a Josef Průša [36]

Original Prusa i3 MK3S je podle výrobce nástupcem v současnosti nejoblíbenější 3D tiskárny na světě, modelu MK2 [37]. Tiskárna tiskne na nový typ magnetické podložky. Disponuje automatickou kalibrací a novým typem podávacích koleček. Zajímavostí je, že tiskárny Průša se dají za nižší cenu objednat nesložené. Zákazník si pak podle návodu tiskárnu sám doma poskládá jako stavebnici.

Tab. 4: Technické specifikace 3D tiskárny [37]

zařízení	3D tiskárna
výrobce	Prusa Research
označení	Prusa i3 MK3S
cena	26 990 Kč
pracovní prostor	25 x 21 x 21 cm
výška vrstvy	0.05 – 0.35 mm
podporované materiály	všechny termoplasty včetně polykarbonátu a nylonu

Obr. 27: 3D tiskárna Průša [37]

4.5 PLC od B+R Automatizace

PLC (programovatelný logický automat) je mozkem všech automatických linek v průmyslu. Je to počítač, na který se dají napojit vstupní (většinou senzory) a výstupní (motory, písty, světla…) periferie [40]. Periferie mohou být jak digitální (komunikovat pomocí jedniček a nul), tak analogové (komunikovat pomocí spojitého signálu). PLC počítače jsou oblíbené zejména v průmyslu díky jejich robustnosti a spolehlivosti. PLC bývají uzpůsobené prašnému prostředí a rázům. U PLC není běžné uživatelské rozhraní tak jako na osobních počítačích. Jejich hlavním úkolem je přepracovávat vstupy na výstupy podle algoritmu, který do PLC zadává programátor. Algoritmy mohou být psány v nejrůznějších jazycích, jsou dokonce i normovány. V poslední době je častým požadavkem na PLC počítače ovládání na dálku, aby programátor mohl pohodlně měnit kód, který řídí automatizovaná pracoviště. Každé PLC má jinou velikost paměti, do které se ukládají algoritmy. Předchůdci PLC byly reléové obvody [40]. Relé je elektrotechnická součástka, která spíná a rozpojuje obvod pomocí elektrického proudu. Tímto proudem se aktivuje elektromagnet a ten mechanicky obvod rozpojí. Soustavou takovýchto součástek se pak v minulosti realizovaly logické operace potřebné v automatizaci. Jejich nevýhodou byla velikost a nutnost fyzického předělávání při každé změně. Centrum moderních PLC automatů je CPU, které na bázi polovodičové techniky zpracovává všechny informace.

Společnost **B+R** je původně rakouská společnost vyrábějící automatizační prvky do průmyslových aplikací. Firma je celosvětově úspěšná. V roce 2017 ji odkoupilo ABB [41].

X20CP1584 je rozměrově menší PLC od firmy B+R využívané v automatických linkách. Má 4 komunikační rozhraní: RS232, Ethernet, POWERLINK a USB. Možno použít i OPC UA.

Tab. 5: Technické specifikace PLC [42]

zařízení	PLC
výrobce	B+R
označení	X20CP1584
cena	27 324 Kč
vstupní napětí	24 VDC
výstupní napětí	24 VDC
chlazení	nemá
paměť RAM	512 MB
rozměry	150 x 99 x 85 mm
hmotnost	400 g

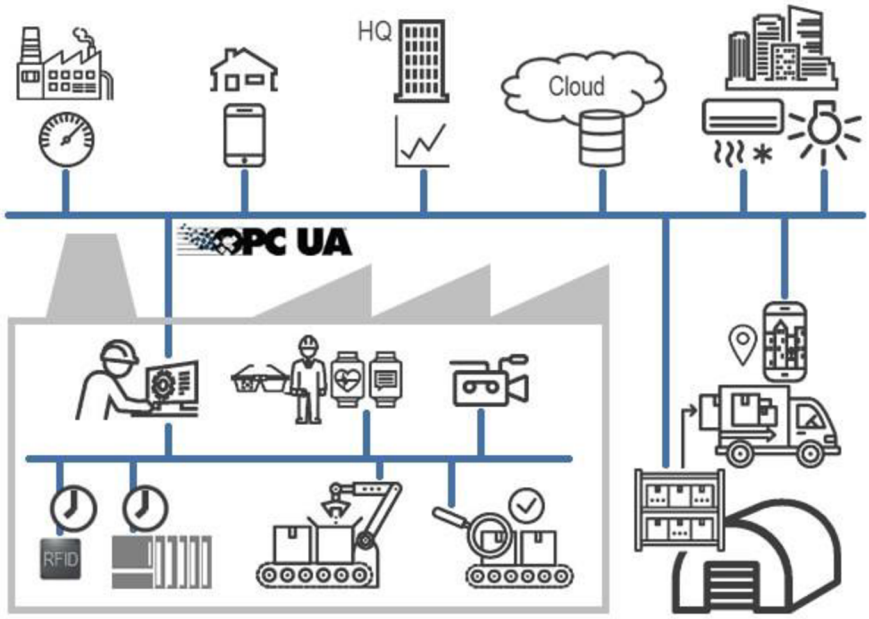
Obr. 28: PLC B+R [42]

4.6 OPC UA komunikace

V počítačovém světě najdeme mnoho způsobů, jak komunikovat mezi periferiemi. Komunikace zajišťuje sdílení informací v reálném čase a rozšiřuje možnosti využití moderních technologií. Zejména v průmyslové automatizaci je komunikace mezi jednotlivými prvky velice důležitá. Prvky mohou být napojeny na centrální počítač a komunikovat skrze něj. Tento způsob komunikace je ale považován za neefektivní, a proto se v poslední době stále více využívá tzv. *Internet of Things*, kde stroje mezi sebou komunikují decentralizovaně [38], tedy *Machine to Machine* (M2M). K tomu, aby

všechny stroje, zařízení a třeba i domácí spotřebiče komunikovaly mezi sebou bez centrálního řízení, je třeba definovat standardizovaný komunikační styl [39].

OPC Unified Architecture je velice rozšířeným způsobem komunikace v průmyslovém i neprůmyslovém světě. Před OPC UA existovalo více způsobů, jak mohla zařízení mezi sebou komunikovat. Tyto způsoby však byly nejednotné a nemohly komunikovat mezi sebou navzájem. OPC UA vyvinula společnost **OPC Foundation.** Tento komunikační standard je dostupný všem operačním systémům a je tak univerzální. OPC UA definuje způsob, jakým jsou data standardizována tak, aby je mohli používat všichni. OPC UA server je dnes součástí mnoha zařízení (například PLC). Na obrázku 29 vidíme schéma OPC UA. Jde vidět, že síť mezi prvky je plně decentralizována.



Obr. 29: OPC UA schéma [38]

5 PŘÍPRAVA VLASTNÍHO ŘEŠENÍ ÚLOHY

Cílem, kterého má být v této práci dosaženo, je návrh a realizace programu pro detekci objektů různých tvarů, barev a velikostí, tedy zpracování obrazových dat z kamery. Před samotným spuštěním programu, který byl vypracován v rámci praktické části bakalářské práce jej bylo třeba připravit. Tomu se věnuje tato kapitola.

5.1 Popis úlohy

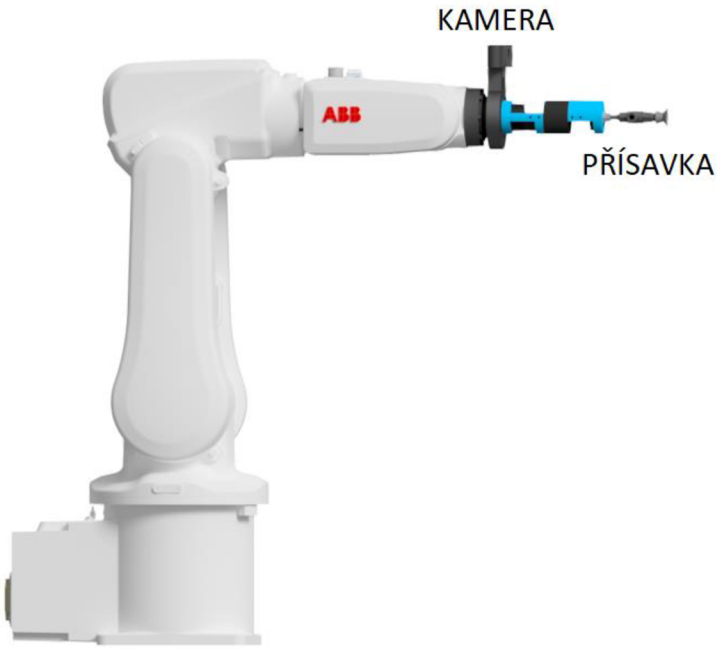
Po konzultaci s vedoucím práce byla úloha stanovena následovně: Primárním cílem úlohy je pomocí robotického ramene roztřídit vstupní objekty různých barev, velikostí a tvarů na odkladiště dle stanovených preferencí.

Jako objekty k roztřízení poslouží tělesa vytisknutá na 3D tiskárně Průša. Tělesa budou nabarvena a zapravena. Pomocí kamery připevněné na robotickém rameni ABB budou tělesa snímána. Kamera bude na robotické rameno připevněna pomocí speciálního nástroje, který se taktéž vytiskne na 3D tiskárně. Nejprve bude zapotřebí vypočítat kalibrační konstanty, díky kterým se zkoordinuje souřadný systém pixelů kamery a souřadný systém robota. Tato kalibrace proběhne pomocí QR kódu o předem známé velikosti hrany.

Pomocí počítače, který bude napojený na kameru, budou obrazová data s objekty zpracována a výstupem tak budou matice s výčtem všech objektů s informacemi o jejich poloze a vlastnostech. V dalším kroku najede robot s kamerou do prostoru, kde budou QR kódy označující odkládací místa. QR kódy v sobě ponesou informaci o tvaru, velikosti a barvě objektu. I tato obrazová data budou zpracována počítačem a výsledkem bude matice skladů s informacemi o poptávaných objektech a jejich poloze. Algoritmus v počítači pak vyhodnotí všechna získaná data a přiřadí jednotlivé objekty ke skladům. Výsledkem bude matice se souřadnicemi objektů a odpovídajících skladů. Posledním krokem bude pomocí OPC UA komunikace sdělit robotovi, aby přemístil objekty na požadovaná místa.

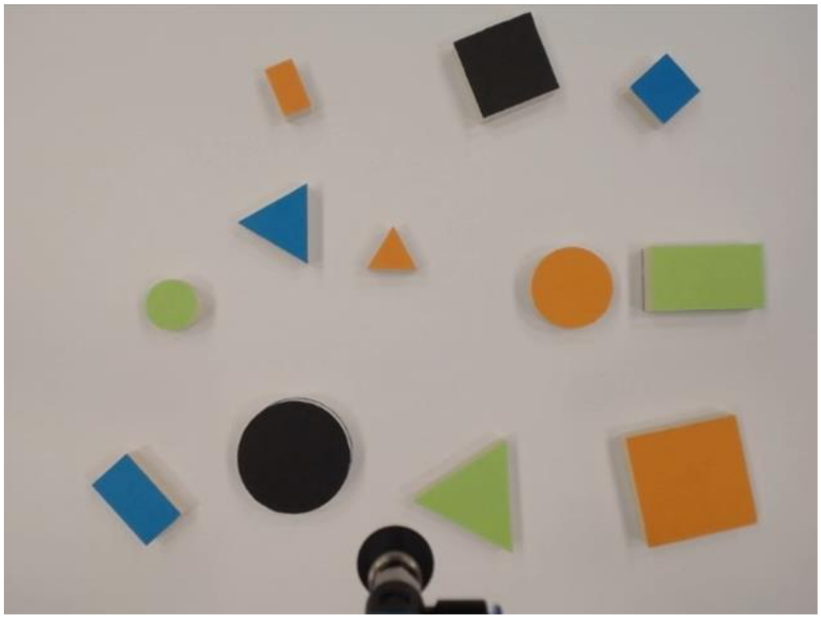
5.2 Příprava komponent

Ze všeho nejdříve je potřeba vytisknout na 3D tiskárně veškeré objekty. Prvním je **držák na kameru** Logitech. Rozměry byly zvoleny podle uchopovacích možností koncové části ramene. Po vytisknutí se držák obrousil a pomocí šroubů připevnil k robotovi a kameře.



Obr. 30: Sestavení robota

Samotné **objekty** jsou také vytisknuté na 3D tiskárně. Zvoleny jsou 4 tvary (obdélník, čtverec, trojúhelník a kruh) o 3 velikostech. Velikosti jsou voleny: 2,5 cm, 4 cm a 5,5 cm. U kruhu jde o průměr, u čtverce o stranu, u obdélníka o úhlopříčku a u trojúhelníka rozhodla výška. Objekty byly vytištěny s výškou 1 cm. Po vytisknutí byly očištěny a přelepeny barevným papírem pro barevné odlišení. Barvy jsou voleny: černá, modrá, zelená a oranžová. Objekty nejsou vytvořeny ve všech možných kombinacích (barvy, tvaru a velikosti), byly voleny tak, aby byly všechny barvy a tvary zastoupeny ve všech třech velikostech.



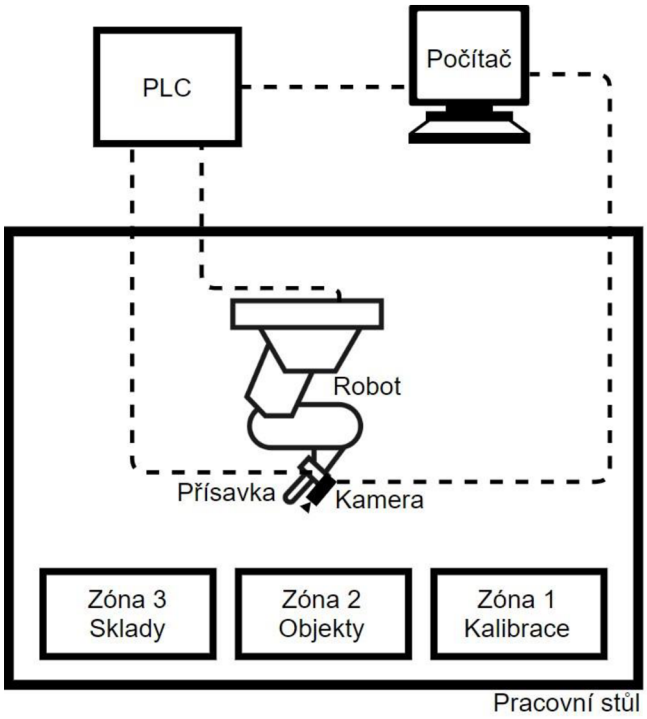
Obr. 31: Objekty

Vytisknout, tentokrát na normální tiskárně, je třeba i **QR kódy**. Jeden kalibrační QR kód byl vytisknut na fotopapír na modernější tiskárně. U kalibračního QR kódu je důležité odměřit jeho hranu a zadat ji jako parametr do řídícího algoritmu. U QR kódů pro rozlišení odkládacích prostor není třeba dbát takové kvality. QR kódy byly vygenerovány pomocí online generátoru QR kódů [43]. Objekty budou vhazovány do plastové krabičky.

Prostor pracovní plochy pod kamerou je potřeba kvůli vysoké citlivosti barevných filtrů podložit jednobarevnou podložkou. Pro jednoduchost byl zvolen tvrdý bílý rýsovací papír o velkém formátu a připevněn magnetkami k pracovnímu stolu.

5.3 Zapojení systému

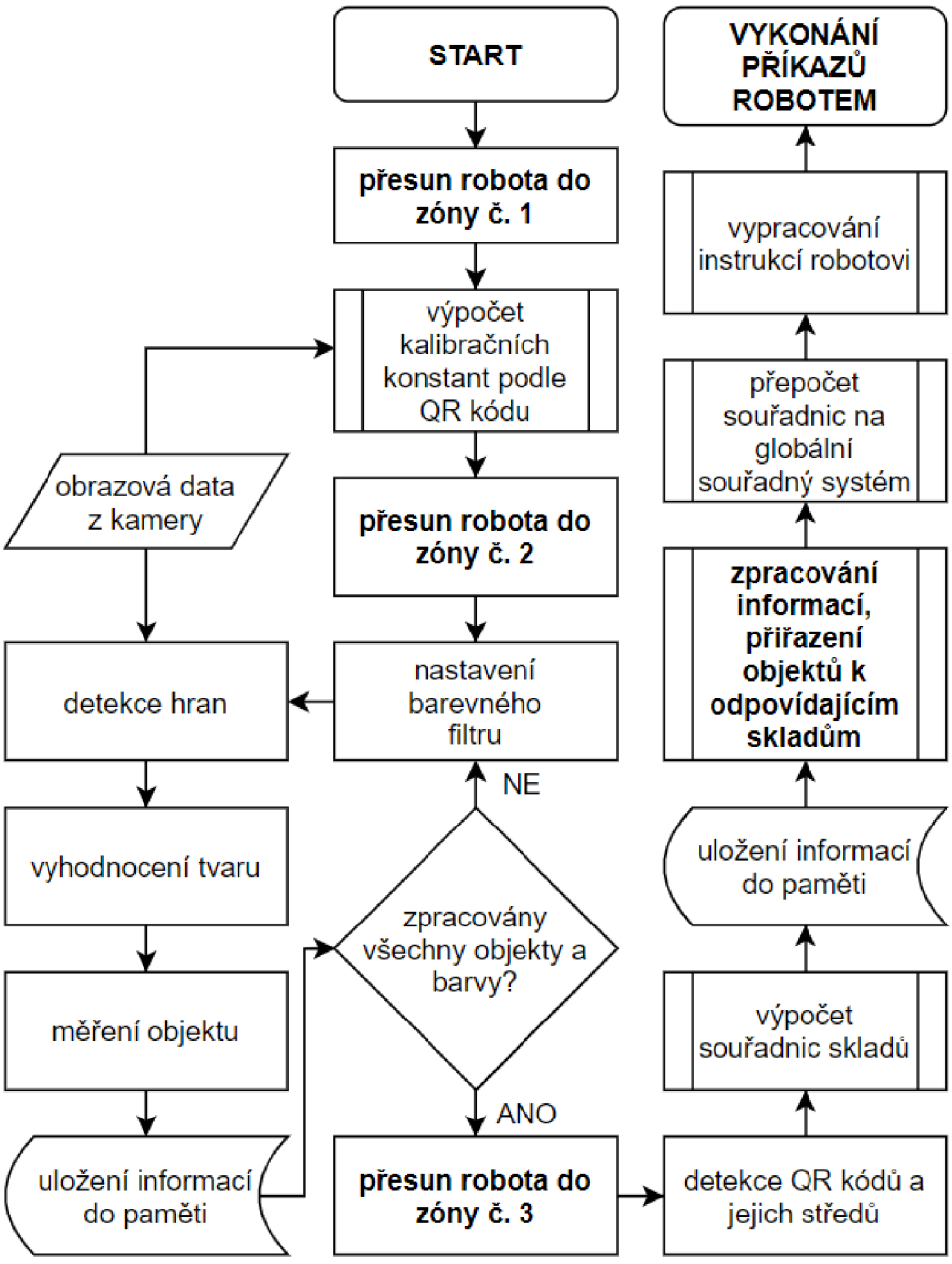
V celém systému jsou zapojeny všechny potřebné periferie a komponenty. Vše začíná u pracovního stolu, který je rozdělen do tří zón. První zóna je kalibrační, leží na ní kalibrační QR kód, robot se zde bude kalibrovat. Ve druhé zóně leží objekty k roztřízení a v poslední odkladiště označené QR kódy. Nad pracovním stolem se nachází kamera a vakuová přísavka připevněná na robotovi ABB. Ten je připevněn k pracovnímu stolu. Vakuová přísavka a robot jsou propojeny s PLC. Kamera je napojena na počítač, na kterém běží řídící algoritmus. Ten je také propojen s PLC a ovládá přes něj chod robota. Schéma systému je na následujícím obrázku:



Obr. 32: Schéma systému

5.4 Řídící algoritmus

Na základě požadavků a cílů úlohy je navržen následující algoritmus pro zpracování dat z kamery. Algoritmus je napsán v jazyce Python 3. Jazyk byl zvolen pro svou jednoduchost a přehlednost. Jsou v něm využity knihovny: OpenCV, NumPy, pyzbar a math. Celý algoritmus je v příloze A.



Obr. 33: Schéma algoritmu

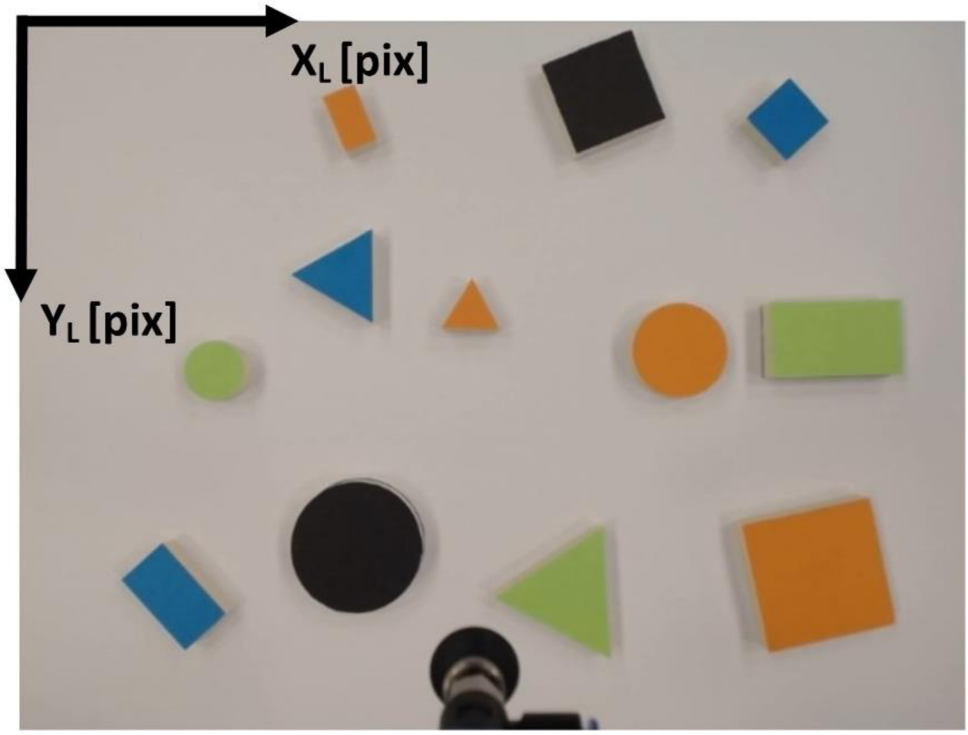
Jeho jednotlivé části jsou popsány v následujících kapitolách.

5.5 Kalibrace

Celý algoritmus začíná s výpočtem kalibračních konstant. Kalibrační konstanty slouží k:

- výpočtu velikosti detekovaných objektů
- přepočítávání lokálního souřadného systému kamery v pixelech na globální souřadný systém robotického ramene

Kamera snímá v rozlišení 640x480. Výstup pak v počítači vypadá zcela přirozeně, pracuje se s ním však v pixelech, které jsou uspořádány do dvou os lokálního souřadného systému kamery. Z leva doprava je orientovaná osa X a shora dolů osa Y.



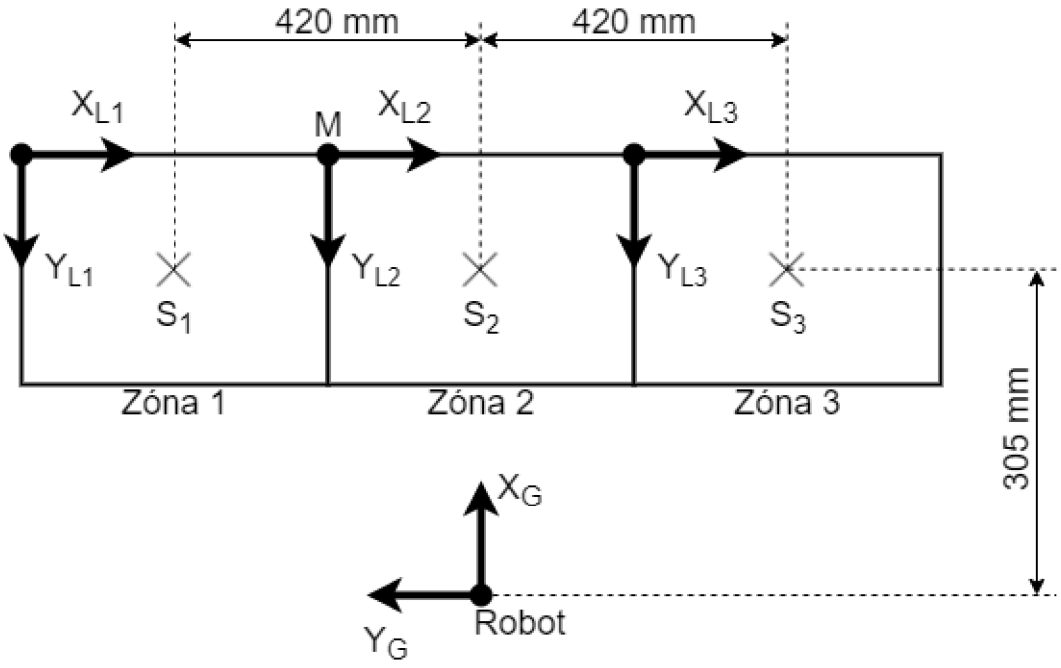
Obr. 34: Pohled kamery

Do paměti je uložena velikost hrany QR kódu. QR kód jako kalibrační médium byl zvolen proto, že je na rozdíl od barevných tvarů spolehlivě a jednoznačně identifikovatelný téměř v jakémkoliv prostředí. Nese v sobě informaci o tom, že se jedná o kalibrační QR kód a zároveň je algoritmus pro vyhledávání QR kódů schopen změřit délku jeho hrany v pixelech. Pakliže známe jeho reálnou délku hrany, jednoduchou matematikou vypočítáme konstantu (pix2mm) pro převádění vzdálenosti na kameře v pixelech na milimetry v reálném měřítku.

$$pix2mm = (hrana\ QR\ kódu\ v\ mm)/(délka\ v\ pixelech) \quad (8)$$

Pro vyhledávání QR kódů v obrazu poslouží integrovaná funkce OpenCV knihovny *cv2.QRCodeDetector()*.

Dalším úkolem kalibrace je výpočet konstant pro přepočet lokálního souřadného systému kamery do globálního souřadného systému robota. Na obrázku 35 je znázorněný pracovní prostor.



Obr. 35: Pracovní prostor

Kalibrační konstanty budou souřadnice libovolného bodu, díky kterému bude pak možno dopočítat souřadnice kteréhokoliv jiného bodu. Byl vybrán bod M. Bod M je vypočítán pomocí vstupních proměnných, které je potřeba před spuštěním programu nadefinovat a pomocí kalibrační konstanty převádějící pixely na milimetry.

$$M_X = 305 + \frac{1}{2} \cdot 480 \cdot pix2mm \tag{9}$$

$$M_Y = \frac{1}{2} \cdot 640 \cdot pix2mm \tag{10}$$

kde 305 je vstupní hodnota proměnné pro vzdálenost robota od osy středů, 640 a 480 je rozlišení kamery v pixelech.

Díky bodu M a konstantě, převádějící pixely kamery na milimetry, můžeme přiřadit kterémukoliv objektu v jakékoliv zóně globální souřadnice.

$$Objekt_{X(GLOBAL)} = M_X - \cdot Objekt_{Y(LOKAL)} \cdot pix2mm \tag{11}$$

$$Objekt_{Y(GLOBAL)} = M_Y - \cdot Objekt_{X(LOKAL)} \cdot pix2mm \tag{12}$$

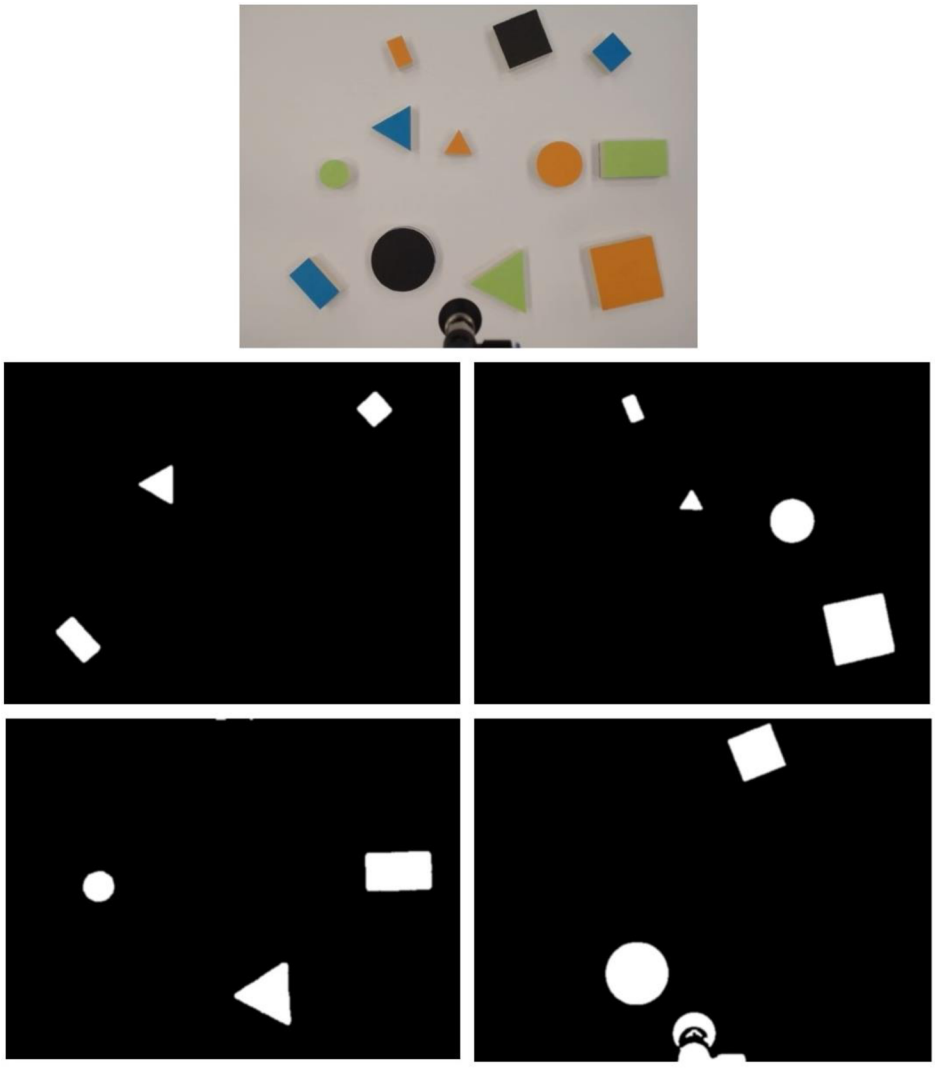
kde $Objekt_{(GLOBAL)}$ jsou globální souřadnice v milimetrech a $Objekt_{(LOKAL)}$ jsou lokální souřadnice v pixelech. Rovnice jsou ukázány pro zónu 2. Pro ostatní je princip stejný, je však třeba přičíst, nebo odečíst vzdálenost zón ve směru Y.

Poloha zón je programu předem známa. V zóně 1 leží kalibrační QR kód, v zóně 2 objekty k roztřízení a v zóně 3 pak QR kódy označující jednotlivé sklady. Robot mezi zónami přejíždí postupně po splnění úkolů. Po kalibrování tedy přejede do zóny s objekty.

5.6 Vyhledávání objektů

Páteřní částí celého algoritmu je detekce samotných objektů. Jak už bylo napsáno, detekovanými objekty jsou čtverce, obdélníky, trojúhelníky a kruhy tří velikostí a čtyř barev.

Objekty jsou detekovány na bílé podložce. Prvním kritériem je **filtr barev**. Tomuto filtru je nutné přiřadit rozmezí hodnot HSV. Tyto hodnoty byly zjištěny experimentálně. Pro následné odstínění nežádoucích hodnot HSV je použita funkce *cv2.inRange()*. Pro odstranění šumu například vlivem osvětlení a odlesku byly použity i následující filtry: *cv2.erode*, *cv2.dilate* a *cv2.GaussianBlur*. Filtrování všech čtyř barev včetně jmenovaných filtrů vidíme na následujícím obrázku.

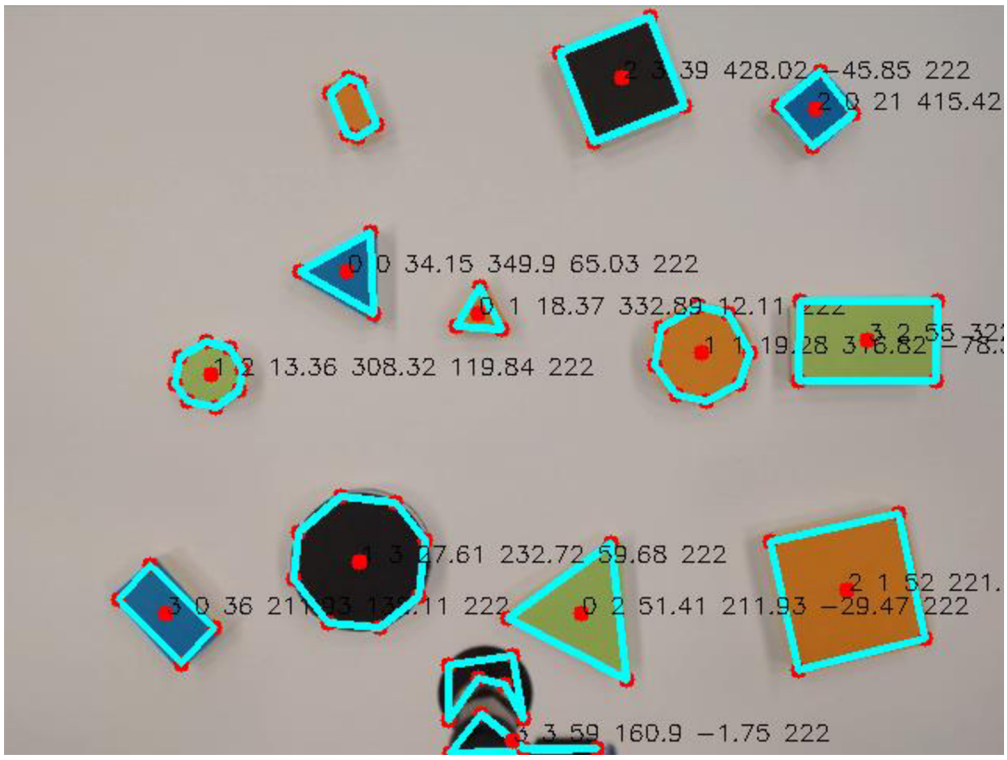


Obr. 36: Ukázka HSV filtru

Následuje funkce pro **detekci contourů** *cv2.findContours*. *Contoury* jsou pro detekci zvoleny proto, že díky jejich počtu bude snadné přiřadit k detekovaným objektům tvar. Do programu je vložena funkce *cv2.contourArea()*, díky které jsou propuštěny jen oblasti ohraničené *contoury*, které mají velikost v požadovaném rozmezí. Budou-li *contoury* 3, program označí tento objekt jako trojúhelník. Bude-li *countourů* více než 7, program to vyhodnotí jako kruh. Problém nastává u rozlišení čtverce a obdélníka, protože mají stejný počet rohů, tedy i *contourů*. Program mezi všemi detekovanými *contoury* vytvoří úsečky a objekty tak obtáhne. Poměrem délek úseček rozliší čtyřhrany na čtverce a obdélníky. Velikost objektů určí velice jednoduše, a to přepočítáním délky hran (u kruhu jde o vzdálenost protějších *contourů*) pomocí kalibrační konstanty. Výsledky si program zapíše do paměti ve formátu:

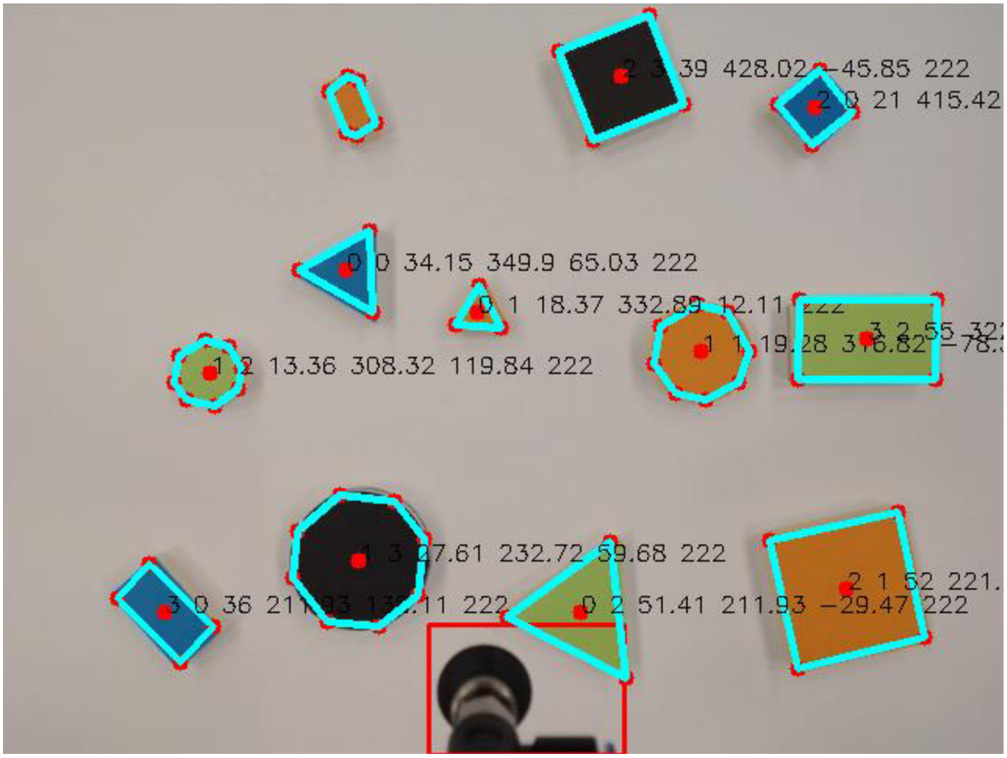
Objekty[[tvar, barva, velikost, středX, středY, středZ], [objekt2], [objekt3]...]

A vykreslí je včetně popisu do obrazu.



Obr. 37: Detekované objekty

Jak vidíte na obrázku 37, program detekuje i vakuovou přísavku, která má černou barvu a je tedy detekována jako černý kruh (protože má mnoho *contourů*). Tento problém je vyřešen vložením požadavku na ignorování části obrazu, kde se přísavka nachází. Do algoritmu tedy musí uživatel před samotným spuštěním vložit i souřadnice, které ohraničí přísavku a dále je bude program ignorovat.



Obr. 38: Selekce přísavky

5.7 Vyhledávání skladů

Následně robot přejede do poslední, třetí zóny. Z obrazových dat bude detekovat odkladiště pro detekované objekty. Tyto odkladiště budou z důvodu jednoznačnosti a citlivosti na okolní šum detekované pomocí QR kódů.

Pro knihovnu OpenCV existují dva spolehlivé algoritmy pro detekci QR kódů.[1] Prvním je již zmiňovaný *cv2.QRCodeDetector()*, který je integrovaný v knihovně OpenCV. Výhodou tohoto algoritmu je, že dokáže zachovat orientaci rohů QR kódu. Bohužel nedokáže identifikovat více než jeden QR kód v obrázku. Druhým používaným algoritmem je pyzbar 0.1.8, který byl vytvořen na MIT [44]. Tento algoritmus stejně tak jako první zmiňovaný vyhledává v obrazu QR kódy a vypisuje jejich informaci. Jeho velkou nevýhodou je, že pořadí rohů vypisuje zleva doprava bez ohledu na orientaci kódů. Není tedy možné určit orientaci QR kódu. Jedinou informací o poloze jsou jeho rohy, potažmo střed. Umí ale detekovat několik QR kódů najednou a vrací souřadnice jejich rohů a informaci v nich napsanou. Z tohoto důvodu byl pro detekci QR kódů označujících sklady vybrán algoritmus pyzbar.

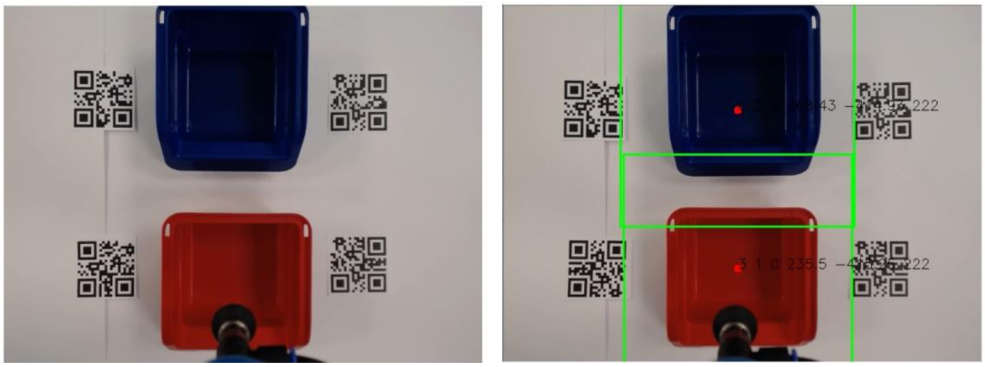
Z důvodu neschopnosti algoritmu orientovat QR kódy musí být pro označení jednoho skladu použity dva QR kódy. Poloha skladu tak bude jednoznačně identifikovatelná na středu spojnice jejich středů. Za polohu skladu nemůžeme brát střed QR kódu, protože by byl po odložení objektu na něj dále nečitelný. V každé dvojici

[1] Řeč je o dvou *open source* algoritmech, které byly zkoumány v listopadu 2019. Další spolehlivé algoritmy nebyly k datu jejich zkoumání nalezeny. Autor však nevylučuje jejich existenci.

QR kódů, označujících jeden sklad je nahrána stejná informace trojčíselným kódem. V tomto kódu je informace o barvě, tvaru a velikosti poptávaných objektů. Algoritmus vyhledá všechny QR kódy a pomocí detekce hran vypočítá souřadnice středů. Porovná QR kódy mezi sebou a mezi těmi stejnými vypočítá jejich střed. Tento střed je vykreslen do obrazu červeně (včetně zeleného ohraničení odkládací oblasti) a uložen do paměti ve formátu:

Sklady[[tvar, barva, velikost, středx, středy, středz], [sklad2], [sklad3]…]

Dvojice QR kódů je umístěna na plochu tak, aby mezi nimi byla plastová krabička pro objekty. Detekce skladů je ukázána na obrázku 39:



Obr. 39: Detekce skladů

5.8 Vyhodnocení informací

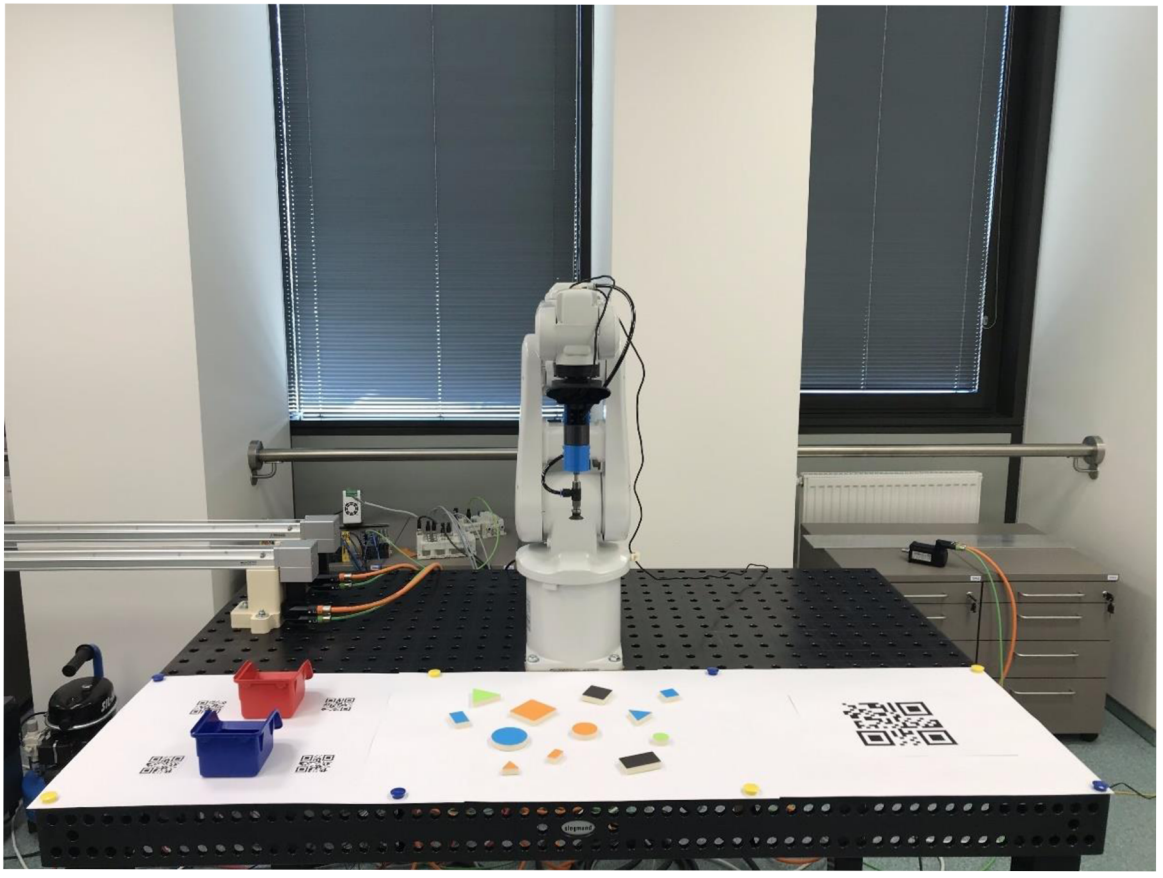
Jakmile robot nasbírá všechny potřebné informace o poloze a charakteristikách skladů a objektů, může je algoritmus začít párovat k sobě tak, aby výstupem byly jednoznačné instrukce pro robota. Porovná mezi sebou matice výstupu detekce objektů a detekce skladů a podle shodnosti proměnných, které symbolizují tvar, barvu a velikost, přiřadí jejich souřadnice k sobě. Robot pak dostane následující matici instrukcí, díky které ví, na jakých souřadnicích má uchopit a na jakých pustit objekt.

Instrukce[[objektX, objektY, objektZ, skladX, skladY, skladZ], [instrukce2], [instrukce3]...]

V maticích se objevuje i souřadnice Z. Nutno podotknout, že na výšce robotického ramene od pracovního stolu ve výpočtech nezáleží, v programu neexistuje její hodnota a ani proměnná, která by ji symbolizovala. Robot má však kvůli vakuové přísavce a výšce plastových objektů svou nulovou hladinu výš, než má nastaveno. Tuto nulovou hladinu je třeba upravit pomocí proměnných, které uživatel zadá na začátku algoritmu. Výška jeho uchopovací hladiny je tedy statická a v celém programu se zobrazuje jako jedno číslo. Hodnota Z zajišťuje, aby robot ve snaze o uchopení objektu nezničil přísavku přitlačením ke stolu.

6 REALIZACE ÚLOHY

Příprava na spuštění experimentu probíhala přesně tak, jak je pospáno v předchozí kapitole. Nejprve bylo zapotřebí všechna zařízení zapojit podle schéma na obrázku 32. Pracovní stůl zase podle obrázku 35.



Obr. 40: Připravený pracovní stůl

Vybrané objekty byly nahodile rozprostřeny v zóně 2. Do zóny 3 byly umístěny dva sklady odpovídající malému oranžovému obdélníku a středně velkému zelenému trojúhelníku. Kódové označení zapsané na QR kódech je 310 a 022.

Do programu byly zadány všechny důležité proměnné. Pro celý program viz přílohu A.

Tab. 6: Vstupní proměnné hodnoty v programu

proměnná	název v programu	hodnota
velikost hrany kalibračního QR kódu [mm]	RealSizeQR	142
vzdálenost mezi přísavkou a kamerou ve směru X [mm]	griptocam	75
výška detekovaných objektů [mm]	thickness_of_object	10
velikost přísavky [mm]	size_of_gripper	192
výška držáku na kameru [mm]	size_of_cameraholder	20
souřadnice přísavky v obrazu [pixel]	gry1	397
souřadnice přísavky v obrazu [pixel]	gry2	480
souřadnice přísavky v obrazu [pixel]	grx1	272
souřadnice přísavky v obrazu [pixel]	grx2	397
globální souřadnice X v zóně 1 [mm]	gripperxcallib	230
globální souřadnice Y v zóně 1 [mm]	gripperycallib	420
globální souřadnice X v zóně 2 [mm]	gripperxobject	230
globální souřadnice Y v zóně 2 [mm]	gripperyobject	0
globální souřadnice X v zóně 3 [mm]	gripperxwarehauses	230
globální souřadnice Y v zóně 3 [mm]	gripperywarehauses	-420
horní mez hodnot HSV modré barvy	BlueLower	[36, 161, 0]
dolní mez hodnot HSV modré barvy	BlueUpper	[255, 255, 255]
horní mez hodnot HSV oranžové barvy	OrangeLower	[0, 161, 127]
dolní mez hodnot HSV oranžové barvy	OrangeUpper	[64, 255, 255]
horní mez hodnot HSV zelené barvy	GreenLower	[27, 0, 62]
dolní mez hodnot HSV zelené barvy	GreenUpper	[64, 255, 255]
horní mez hodnot HSV černé barvy	BlackLower	[0, 0, 0]
dolní mez hodnot HSV černé barvy	BlackUpper	[255, 144, 65]

Pro osvětlení laboratoře bylo třeba do programu zadat hodnoty HSV pro barevný filtr. Tyto hodnoty byly naměřeny vlastním programem (viz přílohu A). Pomocí posuvných tlačítek byly pro všechny barvy vybrány nejlepší kombinace hodnot. Vše probíhalo v reálném čase se stejnou kamerou.

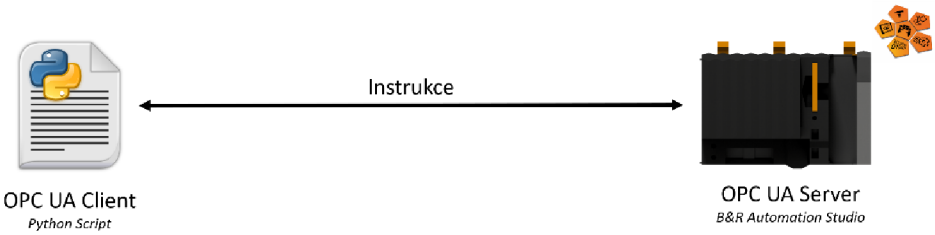
Obr. 42: Ukázka programu pro zjišťování HSV hodnot

6.1 Komunikace OPC UA

Pro komunikaci mezi řídící jednotkou PLC od firmy B+R Automation a řídícím programem pro zpracování obrazu byl zvolen komunikační protokol OPC UA (popsaný v kapitole 4.6 OPC UA komunikace) využívající modelu *Client – Server* viz obrázek 42. Komunikace mezi řídicí jednotkou a robotem ABB může být řešena mnoha různými způsoby (PROFINET, TCP/IP, atd.), nicméně tento problém nebyl předmětem této bakalářské práce.

Server na začátku vyšle příkaz pro spuštění programu na zpracování obrazu. Poté co program vyhodnotí finální instrukce (pozice jednotlivých objektů a skladů), vyšle instrukce do programovatelného automatu (PLC). V automatu se tyto instrukce zpracují a následně se předá příkaz robotu pro jeho vykonání. Program pokračuje opětovným spuštěním.

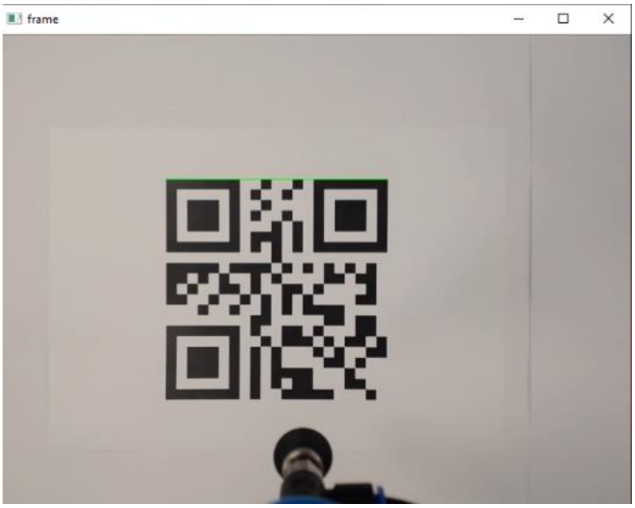
Jednoduchý skript v jazyce Python a rovněž řídící program v Automation Studiu od společnosti B&R Automation je obsažen v příloze A.



Obr. 43: Průběh obousměrné komunikace typu Client – Server

6.2 Průběh a výsledky

Po spuštění řídícího programu se robot přesunul nad zónu č. 1. Po detekci QR kódu pro kalibraci zachytil délku hrany.



Obr. 44: Kalibrace

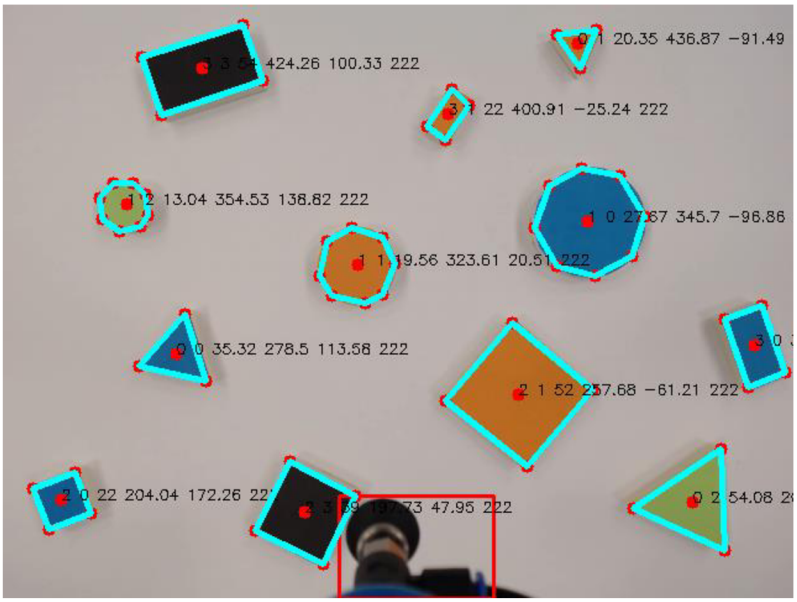
Program vypočítal následující hodnoty:

```
Pix2mm konstanta:
0.6309806837887785
Kalibracni konstanty:
456.4353641093069
201.91381881240912
```

Po přesunu do zóny 2 a zachycení obrazu zpracoval následující obraz:

Obr. 45: Vstupní obraz objektů

Program vyhodnotil vstup takto:

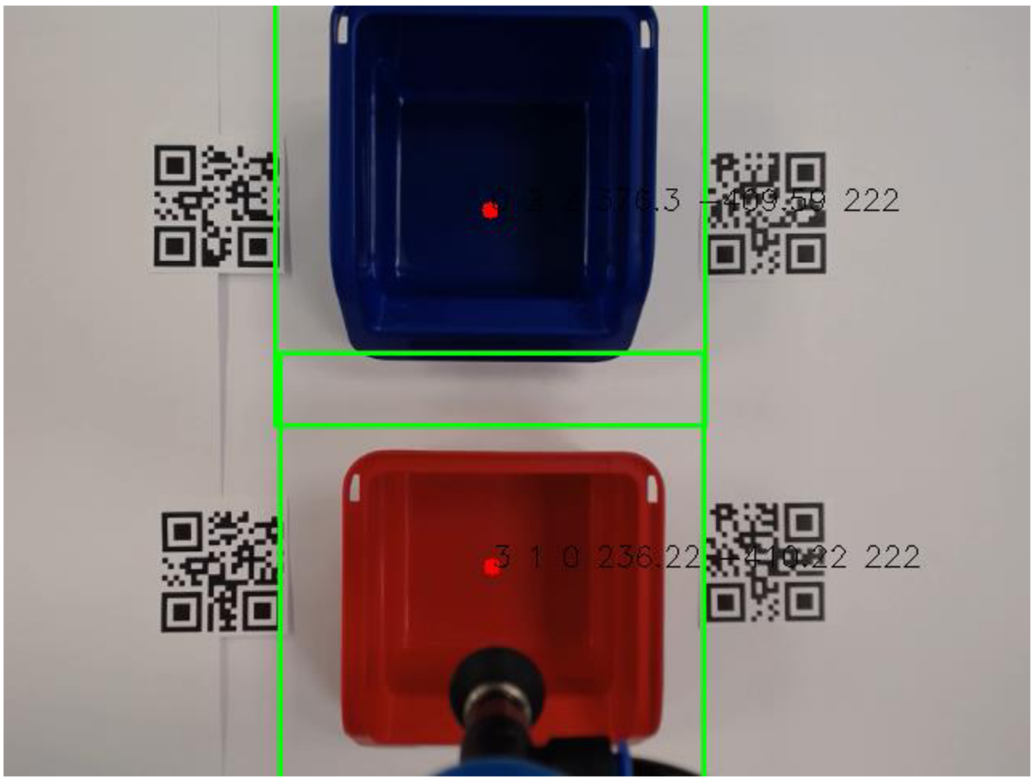


Obr. 46: Vyhodnocení vstupního obrazu objektů

Matice detekovaných objektů vypadala takto:

```
Matice objektu:
[[2, 0, 22, 204.04, 172.26, 222], [0, 0, 35.32, 278.5, 113.58, 222]
, [3, 0, 36, 282.92, 181.72, 222], [1, 0, 27.67, 345.7, 96.86, 222]
, [2, 1, 52, 257.68, 61.21, 222], [1, 1, 19.56, 323.61, 20.51, 222]
, [3, 1, 22, 400.91, 25.24, 222], [0, 1, 20.35, 436.87, 91.49, 222]
, [0, 2, 54.08, 202.78, 150.17, 222], [1, 2, 13.04, 354.53, 138.82,
 222], [2, 3, 39, 197.73, 47.95, 222], [3, 3, 54, 424.26, 100.33, 2
22]]
```

Po přechodu do zóny 3 přijal program obrazová data se sklady a vyhodnotil je následovně:



Obr. 47: Vyhodnocení vstupního obrazu skladů

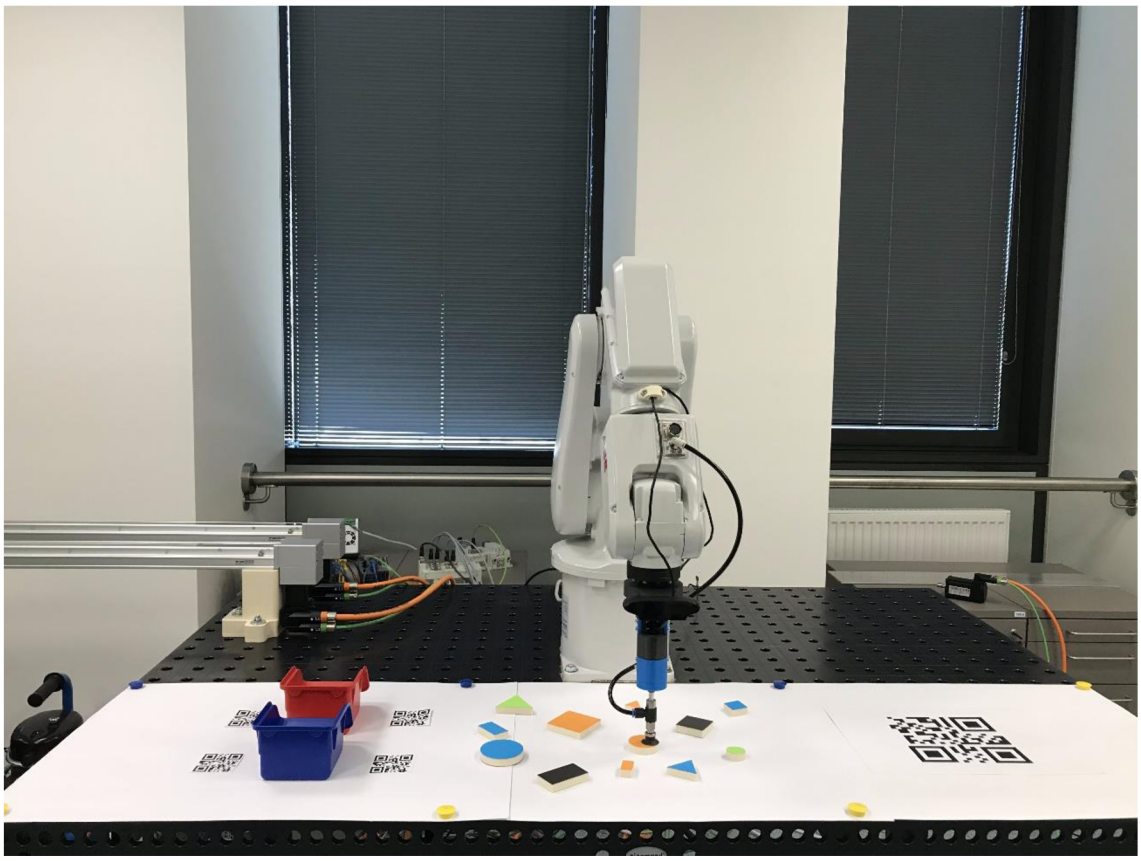
Výstupní matice skladů:

```
Matice skladu:
[[3, 1, 0, 236.22, -410.22, 222], [0, 2, 2, 376.3, -409.59, 222]]
```

Následně program vyhodnotil data a vypsal instrukce pro robota:

```
Finalni instrukce:
[[400.91, 25.24, 222, 236.22, 410.22, 222], [202.78, 150.17, 222, 3
76.3, -409.59, 222]]
```

Po vyhodnocení instrukcí robot začal přemísťovat požadované objekty do skladů.



Obr. 48: Přemísťování objektů

Pro ukázku chodu úlohy na videu (s jiným rozmístěním objektů) viz přílohu A.

7 ZÁVĚR

V rámci bakalářské práce byl vytvořen funkční program pro detekci a třídění objektů. Objekty byly vytištěny na 3D tiskárně Průša. Objekty jsou ve čtyřech barvách (černá, modrá, zelená a oranžová), třech velikostech (řádově centimetry) a čtyřech tvarech (kruh, trojúhelník, čtverec a obdélník). Tyto charakteristiky byly voleny záměrně tak, aby se jednalo o co nejrůznorodější množinu předmětů, a aby co nejlépe simulovaly předměty reálného světa. Na 3D tiskárně byl vytištěn i držák na vakuovou přísavku a kameru, který byl uchycen na koncovém článku robotického ramene ABB. Robotické rameno bylo zapojeno v systému s počítačem, který přijímal data z kamery, a na kterém běžel řídící kód a PLC průmyslovou řídící jednotkou, která ovládala robotické rameno a vakuovou přísavku. Systém komunikoval pomocí protokolu OPC UA. Řídící program v počítači a komunikační protokol byly napsány v programovacím jazyce Python.

Pro přesnou lokalizaci objektů a odkládacích míst vznikla potřeba robota zkalibrovat. Kalibrační konstanty se automaticky vypočítaly pomocí velkého QR kódu. Řídící program znal jeho reálné rozměry a automaticky si dopočítal všechny potřebné kalibrační konstanty. Díky těmto kalibračním konstantám byl pak program schopný v jakémkoliv místě obrazu kamery přepočítat lokální pixelové souřadnice na globální souřadnice robota a pracovního stolu v milimetrech. Kalibrace probíhala na prvním stanovišti v tzv. zóně 1 na úplném začátku samotné úlohy. Po kalibrování se robotické rameno přesunulo nad zónu 2, kde byly náhodně rozmístěny objekty. Řídící program pomocí funkcí knihovny OpenCV upravil obraz filtrováním barev. V takto upraveném obrazu se po několika dalších úpravách vyhledaly rohy jednotlivých objektů pomocí funkcí OpenCV pracujícími s countoury. Počet rohů, jejich vzdálenost a filtrovanou barvu přepracoval řídící program na výstupní informace o barvě, velikosti, tvaru a globálních souřadnicích všech objektů. Poslední detekční úkol čekal robota v zóně 3. V této zóně byly rozmístěny QR kódy, které symbolizovaly odkladiště pro objekty. Řídící program detekoval jejich polohu a informaci v nich obsaženou. Informace odpovídala určité barvě, tvaru a velikosti odkládaného objektu. Následovala fáze vyhodnocení všech vstupních informací. Detekovaným skladům program přiřadil poptávané objekty. Informaci o globálních souřadnicích přemísťovaných objektů a jim odpovídajících skladů poslal pomocí OPC UA komunikačního protokolu do PLC. To tlumočilo instrukce robotickému rameni, které začalo s přemísťováním objektů.

Tato úloha byla prakticky vyzkoušena v laboratoři. Výsledky byly v souladu s očekáváním. Rychlost celého procesu odpovídala testovacímu režimu, mohla by však být navýšena. Přesnost uchopování objektů byla překvapivě velmi dobrá bez nutnosti zásahu. V laboratoři bylo po celou dobu experimentu konstantní osvětlení. Na toto osvětlení byl seřízen i barevný filtr řídícího programu. V případě změny pracoviště nebo světelných podmínek v laboratoři by hodnoty musely být naměřeny znovu. Viditelná část pracovního stolu byla pokryta bílou podložkou. U velmi malých objektů program v některých případech špatně detekoval jejich tvar. Rozlišovací úroveň *coutourů* je totiž

dána vstupním parametrem, který musí vyhovovat jak velkým, tak malým objektům. Detekce QR kódu probíhala ve většině případů velmi spolehlivě. Jejich načtení bylo rychlé a přesné. Zrealizováním této úlohy byl úspěšně splněn cíl bakalářské práce.

Metody, které byly pro detekci objektů v práci použity by našly své uplatnění pro třídění objektů jednodušších tvarů. Algoritmus by mohl být využitý například při třídění různě velkých a barevných plastových uzávěrů od PET lahví, které se mnohdy separují zvlášť. Stejně tak by program posloužil v průmyslu například při kontrole kvality některých jednoduchých produktů.

Největším úskalím celého řešení je jeho úzká využitelnost. Málokteré předměty v reálném světě vykazují takové charakteristiky jako ty, které je program schopen třídit. Komplexnějšímu řešení pro detekci a třídění objektů by mohly pomoci neuronové sítě. Neuronové sítě by posloužily i v prostředí, kde není konstantní osvětlení. Budoucnost detekce objektů tedy patří umělé inteligenci.

8 CITOVANÁ LITERATURA

[1] FARINELLA, Giovanni Maria, BATTIATO, Sebastiano a CIPOLLA, Roberto. Advanced Topics in Computer Vision. London : Springer-Verlag, 2013. ISBN: 978-1-4471-5520-1.

[2] HUANG, T. S. Computer Vision: Evolution and Promise. Illinois : University of Illinois at Urbana-Champaign, 1996.

[3] DAWSON-HOWE, Kenneth. A Practical Introduction to COMPUTER VISION WITH OPENCV. Chichester : John Wiley & Sons Ltd, 2014. ISBN: 9781118848456.

[4] LEVIN, Golan. Image Processing and Computer Vision. openframeworks.cc. [Online] [Citace: 2. Březen 2020.] https://openframeworks.cc/ofBook/chapters/image_processing_computer_vision.html.

[5] barcoding.com. THE HISTORY OF BARCODES. barocding.com. [Online] [Citace: 1. březen 2020.] https://www.barcoding.com/resources/barcoding-basics/the-history-of-barcodes/.

[6] Barcode-generator. [Online] [Citace: 23. Březen 2020.] http://www.barcode-generator.org/.

[7] WAVE, Denso. QRcode.com. [Online] [Citace: 2. Březen 2020.] QRcode.com.

[8] SOON, Tan Jin. QR code. Synthesis Journal. 2008.

[9] ISO/IEC 18004:2015. ISO/IEC 18004:2015. místo neznámé : International Organization for Standardization, 2015.

[10] BEHNKE, Sven. Hierarchical Neural Networks for Image Interpretation. místo neznámé : Springer Science & Business Media, 2003. ISBN: 3-540-40722-7.

[11] DEHAL, Aditya. How to build a Neural Network from scratch. freeCodeCamp. [Online] 11. Listopad 2019. [Citace: 2. Březen 2020.] https://www.freecodecamp.org/news/building-a-neural-network-from-scratch/.

[12] tensorflow.org. Why TensorFlow. TensorFlow. [Online] [Citace: 2. Březen 2020.] https://www.tensorflow.org/about.

[13] REDMON, Joseph. You only look once: Unified, real-time object detection. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[14] pjreddie.com. YOLO: Real-Time Object Detection. pjreddie.com/. [Online] [Citace: 2. Březen 2020.] https://pjreddie.com/darknet/yolo/.

[15] SATHIYAMOORTHY, S. INDUSTRIAL APPLICATION OF MACHINE VISION. International Journal of Research in Engineering and Technology. 2014.

[16] OpenCV. About. OpenCV. [Online] [Citace: 12. Březen 2020.] https://opencv.org/about/.

[17] MORDVINTSEV, Alexander. OpenCV-Python Tutorials. OpenCV-Python Tutorials. [Online] [Citace: 12. Březen 2020.] https://opencv-python-tutroals.readthedocs.io/en/latest/index.html.

[18] hisour.com. RGB COLOR MODEL. hisour.com. [Online] [Citace: 13. Březen 2020.] https://www.hisour.com/rgb-color-model-24867/.

[19] KAU, Lih-Jen a LEE, Tien-Lin. An Efficient and Self-Adapted Approach to the Sharpening of Color Images. TheScientificWorldJournal. 2013.

[20] MARDIRLIS, Vassilios a CHATZIS, Vassilios. A Configurable Design for Morphological Erosion and Dilation Operations in Image Processing using Quantum-dot Cellular Automata. Journal of Engineering Science and Technology. 2016.

[21] SAHIR, Sofiane. Canny Edge Detection Step by Step in Python — Computer Vision. Medium. [Online] 25. Leden 2019. [Citace: 13. Březen 2020.] https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123.

[22] CANU, Sergio. Fourier Transform – OpenCV 3.4 with python 3 Tutorial 35. Pysource . [Online] 4. Srpen 2018. [Citace: 13. Březen 2020.] https://pysource.com/2018/08/04/fourier-transform-opencv-3-4-with-python-3-tutorial-35/.

[23] User: LYBER. Finding the radius and extreme coordinate points of the circles(Hough Circle Transform). Stack Overflow. [Online] 27. Září 2016. [Citace: 14. Březen 2020.] https://stackoverflow.com/questions/39721547/finding-the-radius-and-extreme-coordinate-points-of-the-circleshough-circle-tra.

[24] OLIVIERA, Saulo, GOMES, Joao Paulo a ROCHA NETO, Ajalmar. Sparse Least-Squares Support Vector Machines via Accelerated Segmented Test: a Dual Approach. Neurocomputing. 2018.

[25] MACENAUER, Andrej. Jak funguje digitální fotoaparát? FotoAparát.cz. [Online] 4. Březen 2002. [Citace: 23. Březen 2020.] https://www.fotoaparat.cz/clanek/229/jak-funguje-digitalni-fotoaparat-5017/.

[26] Abouth Logitech. Logitech. [Online] [Citace: 23. Březen 2020.] https://www.logitech.com/en-us/about.html.

[27] C930E BUSINESS WEBCAM. logitech.com. [Online] [Citace: 23. Březen 2020.] https://www.logitech.com/en-us/product/c930e-webcam.

[28] DUCHOSLAV, Petr. Co je to kolaborativní robot? 5 věcí, které byste o něm měli vědět. Factory Automation. [Online] FANUC Czech s.r.o., 5. Březen 2017. [Citace: 23. Březen 2020.] https://factoryautomation.cz/co-je-to-kolaborativni-robot-5-veci-ktere-byste-o-nem-meli-vedet/.

[29] PANDEY, Ankit. Describe robot configurations with neat sketch. (Any two) OR Explain various robot configuration with sketches. Ques10. [Online] 2019. [Citace: 23. Březen 2020.] https://www.ques10.com/p/39905/describe-robot-configurations-with-neat-sketch-any/.

[30] About ABB Robotics. ABB. [Online] [Citace: 23. Březen 2020.] https://new.abb.com/products/robotics/about-us.

[31] ABB. Product specification IRB 120. [Online] 2019. [Citace: 23. Březen 2020.] https://search-ext.abb.com/library/Download.aspx?DocumentID=3HAC035960-001&LanguageCode=en&DocumentPartId=&Action=Launch.

[32] COVAL. VACUUM HANDLING PHASES. COVAL VACUUM TECHNOLOGY INC. [Online] [Citace: 23. Březen 2020.] https://www.coval-inc.com/vacuum-technology/vacuum-handling-guide/suction-pad-operating-mode/.

[33] SMC. SMC: Continuing to pursue worldwide customer satisfaction and support automation. SMC Corporation. [Online] [Citace: 23. Březen 2020.] https://www.smc.eu/uk-ua/company.

[34] —. ZPR (U,C,D,B,UT,CT)(J,K), Vacuum Pad, Lateral Entry, w/Buffer. SMC Corporation. [Online] [Citace: 23. Březen 2020.] https://www.smcusa.com/products/?id=29330&partNumber=ZPR20UNK10-06-A10&type=Detail.

[35] PRŮŠA, Josef, STŘÍTESKÝ, Ondřej a BACH, Martin. Základy 3D tisku s Josefem Průšou. [Online] 2019. [Citace: 24. Březen 2020.] https://www.prusa3d.cz/wp-content/uploads/zaklady-3d-tisku.pdf.

[36] Prusa Research. AHOJ, MY JSME PRUSA RESEARCH! Prusa Research. [Online] [Citace: 23. Březen 2020.] https://www.prusa3d.cz/o-nas/.

[37] —. 3D tiskárna Original Prusa i3 MK3S. Prusa Research. [Online] [Citace: 24. Březen 2020.] https://shop.prusa3d.com/cs/3d-tiskarny/181-3d-tiskarna-original-prusa-i3-mk3s.html.

[38] PLC - Programovatelný logický automat. PLC-AUTOMATIZACE . [Online] [Citace: 24. Březen 2020.] http://plc-automatizace.cz/knihovna/plc.htm.

[39] B&R. O nás. B&R. [Online] [Citace: 24. Březen 2020.] https://www.br-automation.com/cs/o-nas/.

[40] —. X20CP1584. B&R. [Online] [Citace: 24. Březen 2020.] https://www.br-automation.com/cs/produkty/plc-systems/x20-system/x20-cpus/x20cp1584/?noredirect=1.

[41] Renesas Electronics Corporation. OPC Unified Architecture (OPC UA). Renesas Electronics Corporation. [Online] [Citace: 24. Březen 2020.] https://www.renesas.com/cn/zh/solutions/industrial-automation/industrial-network/industrial-ethernet-and-fieldbus/opc-ua.html.

[42] HOPPE, Stefan. OPC Unified Architecture. [Online] 2019. [Citace: 24. Březen 2020.] https://opcfoundation.org/wp-content/uploads/2017/11/OPC-UA-Interoperability-For-Industrie4-and-IoT-EN.pdf.

[43] QR Code Generator. [Online] [Citace: 31. Březen 2020.] https://www.qr-code-generator.com/.

[44] pyzbar 0.1.8. PyPI. [Online] [Citace: 31. Březen 2020.] https://pypi.org/project/pyzbar/.

9 SEZNAM PŘÍLOH

Příloha A: CD-ROM

PŘÍLOHA A.CD-ROM

Tab. 1: Obsah CD-ROM: hlavní adresář:
(„\2020_BP_MARŠALA_Štepán_200203_VUT_FSI_UAI.zip\“)

adresář	název souboru
\PDF\	2020_BP_MARŠALA_Štepán_200203_VUT_FSI_UAI.pdf
\Řídící program\	hsv_test.py ridici_program.py
\OPCUA Client\	client_opcua.py
\Hlavní Program Automation Studio\	OPCUA_Client_Vision_OpenCV.zip
\Video\	realizace_ulohy.mp4
	README.txt