ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STROJNÍ

ÚSTAV MECHANIKY, BIOMECHANIKY A MECHATRONIKY

Fúze senzorů pro strojové vidění s neuronovou sítí

Sensor fusion for machine vision with neural network

Diplomová práce

Vypracoval: Bc. Vojtěch Barnat

Obor: Průmysl 4.0

Vedoucí práce: doc. Ing. Ivo Bukovský, Ph.D.

Rok: 2021

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Barnat** Jméno: **Vojtěch** Osobní číslo: **465349**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav mechaniky, biomechaniky a mechatroniky**

Studijní program: **Průmysl 4.0**

Studijní obor: **bez oboru**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Fúze senzorů pro strojové vidění s neuronovou sítí

Název diplomové práce anglicky:

Sensor fusion for machine vision with neural network

Pokyny pro vypracování:

Proveďte rešerši metod a aplikací fůze senzorů různých spekter (RGB, DS, IR) pro rozšíření možností strojového vidění s potenciálem pro robotické aplikace založených na zpracování obrazu a kde právě obrazová informace nemusí být dostačující.
Proveďte rešerši algoritmů pro reálně-časovou detekci objektů a vyberte řešení vhodné pro implementaci na nízkonákladový HW typu mikropočítač Nvidia Jetson nano s využitím open-source SW a s možnou podporou nástrojů strojového učení.
Pro vybranou aplikaci strojového vidění s fůzí senzorů nebo její zjednodušený reálný model implemetujte vaše řešení včetně SW aplikace v open-source a řešení a grafického rozhraní.
Experimentálně vyhodnoťte vaše řešení a náležitě zdokumentujte.
Řešení řádně zdokumentujte.
Rozsah práce min. 50 stran + přílohy.
Grafický obsah max 60 %.

Seznam doporučené literatury:

[1] Šípek Vojtěch:VYUŽITÍ OBRAZOVÉ ANALÝZY A JEJÍCH ALGORITMŮ PŘI DEFEKTOSKOPII NANOVLÁKENNÝCH VRSTEV, Diplomová práce, FS ČVUT v Praze,2019-02-05, https://dspace.cvut.cz/handle/10467/80564
[2] GOHLKE, Christoph. imreg: FFT based image registration [online]. Python. Dostupné z: https://www.lfd.uci.edu/~gohlke/ https://pypi.org/project/imreg/
[3] Convolutional Neural Network (CNN) | TensorFlow Core. TensorFlow [online]. [vid. 2020-04-06]. Dostupné z: https://www.tensorflow.org/tutorials/images/cnn
[4] GIRSHICK, Ross, Jeff DONAHUE, Trevor DARRELL a Jitendra MALIK. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2014, , 580-587. ISBN 978-1-4799-5118-5. Dostupné z: doi:10.1109/CVPR.2014.81

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Ivo Bukovský, Ph.D., U12110.3

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

Datum zadání diplomové práce: **28.04.2021** Termín odevzdání diplomové práce: **13.08.2021**

Platnost zadání diplomové práce: _____

doc. Ing. Ivo Bukovský, Ph.D. podpis vedoucí(ho) práce	doc. Ing. Miroslav Španiel, CSc. podpis vedoucí(ho) ústavu/katedry	prof. Ing. Michael Valášek, DrSc. podpis děkana(ky)

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a použil pouze podklady uvedené v přiloženém seznamu literatury.

V Praze dne _____

_____

Vojtěch Barnat

Poděkování

Mé obrovské díky patří doc. Ing. Ivovi Bukovskému, Ph.D. za vedení mé diplomové práce, odborné připomínky, cenné rady, veškerý mi věnovaný čas a především za motivaci věnovat se tomuto oboru i nadále v doktorském studiu. Dále bych chtěl poděkovat svým rodičům za podporu během mého studia. Můj dík patří také mé přítelkyni za její podporu a trpělivost při zpracování této práce a po celou dobu mého studia.

Název práce:	Fúze senzorů pro strojové vidění s neuronovou sítí
Jméno autora:	Vojtěch Barnat
Druh práce:	Diplomová práce
Vedoucí bakalářské práce:	doc. Ing. Ivo Bukovský, Ph.D.

Abstrakt:

Tato práce se zabývá vytvořením prototypu fúze senzorů viditelného spektra – RGB kamery, hloubkového senzoru a infračerveného spektra – termokamery pro řešení úlohy bin picking, tedy detekce objektů v reálném čase k následné robotické manipulaci. Nejprve je zpracován obecný přehled metod strojového vidění pro detekci objektů. V praktické části je vytvořen experiment, který ve zmenšeném měřítku simuluje reálnou průmyslovou aplikaci. Pro účely vývoje a definici parametrů je vytvořeno grafické rozhraní a následně je implementován samotný modul detekce.

Klíčová slova:

Fúze senzorů, RGB kamera, hloubkový senzor, termokamera, strojové vidění, registrace obrazů, kalibrace kamery, homografie, template matching, supervizorované strojové učení, konvoluční neuronové sítě pro detekci objektů, YOLO v4

Title:	Sensor fusion for machine vision with neural network
Author:	Vojtech Barnat
Thesis type:	Diploma thesis
Supervisor:	doc. Ing. Ivo Bukovsky, Ph.D.

Abstract:

This thesis deals with the creation of a sensor fusion prototype of visible spectrum sensors - RGB camera, depth sensor and infrared spectrum - a thermal camera for solving the problem of bin picking, i.e. detection of objects in real time for subsequent robotic manipulation. First, a general overview of machine vision methods for object detection is elaborated. In the practical part, an experiment is created that simulates a real industrial application on a reduced scale. For the purposes of development and definition of parameters, a graphical interface is created and then the detection module itself is implemented.

Keywords:

Sensor fusion, RGB camera, depth sensor, thermal imager, machine vision, image registration, camera calibration, homography, template matching, supervised machine learning, convolutional neural networks for object detection, YOLO v4

Obsah

1 Úvod

Přestože žijeme v době čtvrté průmyslové revoluce a automatizace zdá se vládne průmyslu, stále se v hojném počtu vyskytují situace, kdy je výrazně ekonomičtější využít lidskou pracovní sílu pro určité fáze výroby, a v některých případech je dokonce člověk stále zcela nenahraditelný. Doba, kdy bude naprostá většina průmyslových podniků zcela robotizována, je tak v nedohlednu a nejčastěji se přistupuje k robotizaci pouze částečné. To však s sebou přináší určité problémy, které souvisejí se zcela odlišnou podstatou člověka a stroje. Pokud pomineme zřejmá bezpečnostní rizika, je zde problém přesné manipulace s materiálem. Výrobní stroje potřebují ke správnému fungování zpravidla naprosto přesné polohování vstupujících polotovarů, avšak často jsou výrobky v předcházejícím výrobním kroku skládány člověkem zcela nahodile například do přepravky. Jedním z řešení tohoto problému je technologie strojového vidění tzv. bin picking, která slouží k nalezení, uchopení a následné přesné robotické manipulaci polotovarů. Tato úloha není zdaleka triviální a pro její řešení lze s výhodou využít kombinaci senzorů, která umožňuje získat detailní informace o snímaném prostoru. Jedná se tak o aplikaci procesu tzv. fúze senzorů neboli kombinaci senzorických dat za účelem zkvalitnění získávaných informací.

Tato práce vzniká pro potřeby aktuálně řešených problémů v průmyslové praxi ve spolupráci s firmou VM Engineering s.r.o., která působí v oboru průmyslu 4.0 již řadu let a zabývá se mimo jiné průmyslovými aplikacemi strojového vidění. Jedním z řešených problémů je zmíněná technologie bin picking, pro kterou je navrhováno použití kombinace senzoru viditelného spektra (digitální kamery) a hloubkového senzoru. Dalším polem působení této firmy je kontrola rozložení teplotního pole při svařování plastů, kde se jeví žádoucí využití termografického snímače (termokamery). Vzniká tak návrh na vytvoření prototypu fúze senzorů viditelného spektra, hloubkového senzoru a infračerveného spektra.

2 Cíl práce

Cílem této práce je pomocí fúze dat z barevné kamery (RGB) a hloubkového senzoru (D) vyřešit úlohu bin picking, tedy detekci objektů v reálném čase k následné robotické manipulaci. Následně pomocí fúze dat z termografického snímacího systému (T) získat informace o teplotě detekovaných objektů. Nejprve bude provedena studie algoritmů pro detekci objektů a navrženo vhodné řešení. V praktické části bude vytvořen experiment, který bude ve zmenšeném měřítku simulovat reálnou aplikaci pro využití snímaných dat k určení polohy a teploty polotovarů. Polotovary mají tvar trubek a jsou zobrazeny na Obrázku 1 níže.

Obrázek 1 – Gitterbox s polotovary

Dílčí cíle práce lze formulovat následujícími body:

- Rešerše algoritmů pro reálně-časovou detekci objektů a řešení vhodné pro implementaci na nízkonákladový HW typu mikropočítač Nvidia Jetson nano s využitím open-source SW.

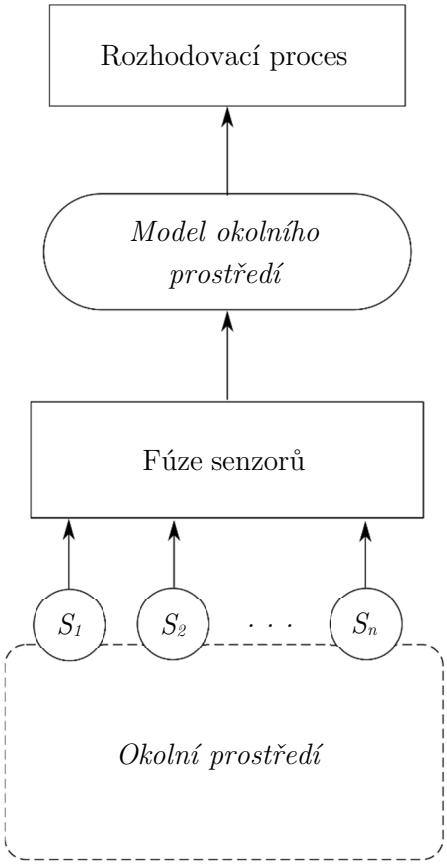
- Formulace řešení úlohy bin picking s využitím dat z barevné kamery a hloubkového senzoru pro reálnou aplikaci a jeho implementace

- Fúze RGB-D-T

3 Teoretická část

3.1 Fúze senzorů

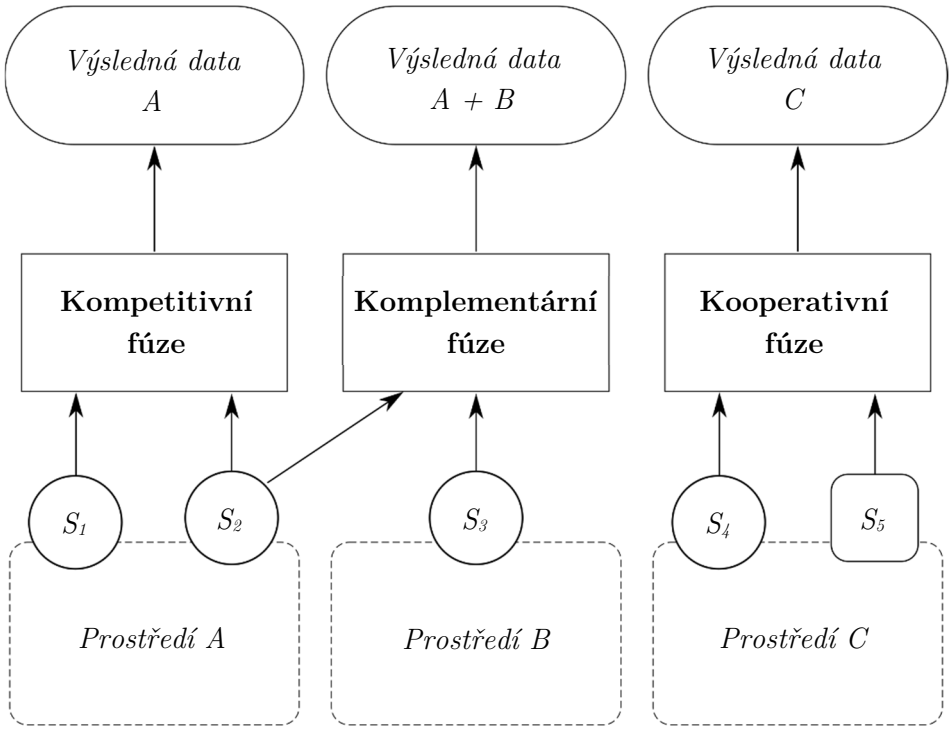
Fúze senzorů je proces integrace informací z různých zdrojů, který vede ke snížení nejistoty o pozorovaném ději na úroveň, kterou nelze dosáhnout při použití zdrojů odděleně. Tento princip lze pozorovat v přírodě, kdy evoluční úspěch je připisován druhům, kterým se vyvinuli komplexnější smyslové orgány. Lidský organismus je dobrým příkladem sytému využívajícího senzorovou fúzi. Všech pět lidských smyslů vytváří dynamický model okolního prostředí, který slouží k interakci s okolním prostředím a k procesu rozhodování pro současné a budoucí děje. Tento systém je natolik robustní, že i v případě senzorové deprivace je schopen kompenzovat chybějící informace použitím dat získaných z ostatních senzorů s překrývajícím rozsahem.



Obrázek 1 – Diagram fúze senzorů

Fúze senzorů je v dnešní době hojně využívaný koncept v technických aplikacích pro dosažení vyšší robustnosti a spolehlivosti systémů, rozšíření jejich prostorového a spektrálního pokrytí a celkové zvýšení jejich přesnosti. Moderním oborem hojně využívajícím tyto principy je strojového vidění a příkladem senzorů mohou být Radar, LIDAR či kamerové systémy různých spekter, kdy každé spektrum udává odlišnou informaci o snímaném prostředí. Typickým příkladem produktu, který je v současné době často diskutován a je založen na fúzi zmíněných senzorů je autonomní automobil. Fúze senzorů však může být s výhodou aplikována i na výrazně méně komplexní systémy průmyslové praxe, a to například úlohy typu bin picking.

Dle vzájemných vztahů jednotlivých senzorů lze rozlišovat tři základní kategorie fúze [1], které jsou znázorněny na Obrázku 3.



Obrázek 2 – Znázornění kategorií fúze senzorů, schéma vytvořeno podle [1]

Při kompetitivní fúzi snímají identické senzory stejné prostředí a udávají nezávislé měření stejné veličiny. Cílem je zde zvýšit přesnost při zajištění redundance a zvýšení spolehlivosti. Komplementární fúze zajišťuje rozšíření pokrytí pomocí vzájemně nezávislých senzorů, kdy každý snímá různou část prostředí. Kooperativní fúze je konfigurace různých senzorů snímajících stejnou scénu pro rozšíření získaných informací [1].

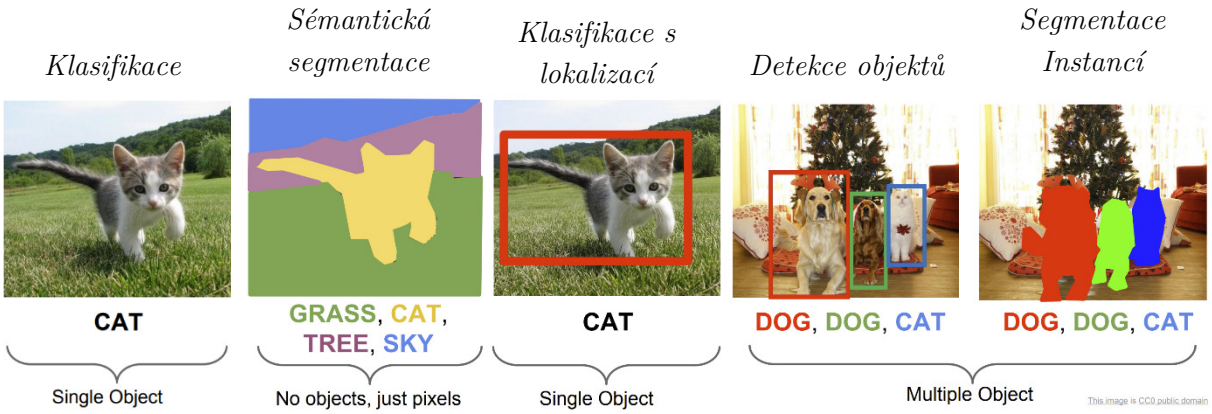
Podoba řešení fúze senzorů silně závisí na konkrétní úloze, a proto nelze hovořit o existenci univerzálního postupu pro sestavení modelu fúze, avšak existují některé metody a algoritmy, které se často pojí s tímto tématem. V oboru zpracování signálu je to především známý Kálmánův filtr [2], který je hojně využíván pro jeho jednoduchost, optimalitu a snadnou implementaci. Podle [3], kde je proveden detailní průzkum metod fúze dat, patří Kálmánův filtr k tzv. pravděpodobnostnímu přístupu fúze dat, která stojí na principech Bayesovské statistiky. Přístupů však existuje celá řada, přičemž jedním z těch moderních je například tzv. Fuzzy logika [4; 5].

3.2 Strojové vidění

Pojem strojové vidění (machine vision) je nejčastěji chápan jako specifická oblast počítačového vidění (computer vision), která se využívá v průmyslových aplikacích, především pro kontrolu procesů a navádění robotických manipulátorů. Počítačové vidění je forma zpracování obrazu za účelem extrahování informace z obrazových dat.

3.2.1 Základní úlohy

V oblasti počítačového vidění se rozlišují následující základní úlohy.



Obrázek 3 – Porovnání základních úloh zpracování obrazu, seřazeny vzestupně dle složitosti [48]

Klasifikace

Klasifikace obrazových dat je základní úlohou počítačového vidění. Úkolem je přiřadit danému obrázku příslušnou třídu. Tato úloha je typicky používána jako příklad použití neuronových sítí pro klasifikaci dat. Je zde možné použít supervizorované konvoluční neuronové sítě a aplikovat učení s učitelem, ale také nesupervizorované, samoorganizační sítě. Výsledkem je aktivace neuronu na výstupní vrstvě, který odpovídá predikované třídě. Supervizorované neuronové sítě je zapotřebí učit pomocí anotovaných dat, která jsou ve formě obrázků s označením jejich třídy. Proces učení v jednoduchosti znamená iterativní změny vah jednotlivých neuronů k dosažení minima chybové funkce. Podrobněji je tato problematika popsána v Kapitole 4.2. Dobrým a často používaným příkladem této úlohy je klasifikace ručně psaných číslic z databáze MNIST [6].

Obrázek 4 – ukázka z MNIST datasetu ručně psaných číslic [6]

Klasifikace s lokalizací

Úlohu klasifikace je možné rozšířit o lokalizaci klasifikovaného objektu. Výstupem detekčního algoritmu je zde jak třída objektu na obrázku, tak vektor jeho polohy $[x, y]$ a rozměr rámečku $[w, h]$, který daný objekt v obrázku ohraničuje. Při řešení této úlohy pomocí supervizorovaného učení je zapotřebí použít správně anotovaná data, tedy obrazová data s určením příslušné třídy a také s polohou a rozměry ohraničujícího rámečku. Pro cíle této práce není tato úloha vhodná, protože se zde nerozlišuje více objektů v jednom obrázku.

Sémantická segmentace

Sémantická neboli významová segmentace je metodou zpracování obrazu, která vede k zařazení každého pixelu na obrázku k příslušné třídě. Nerozlišují se zde od sebe různé instance stejné třídy. Pokud je na obrázku více objektů, které jsou zařazeny do jedné třídy, je výstupem oblast pixelů, kterou tyto objekty společně na obrázku vymezují. Tato metoda je tak vhodná například k rozlišení pozadí obrázku či k určení plochy zabrané zvolenou třídou objektů, nikoliv však k určení polohy konkrétních objektů v obrazových datech. Tato metoda tak není vhodná pro účely této práce.

Detekce objektů

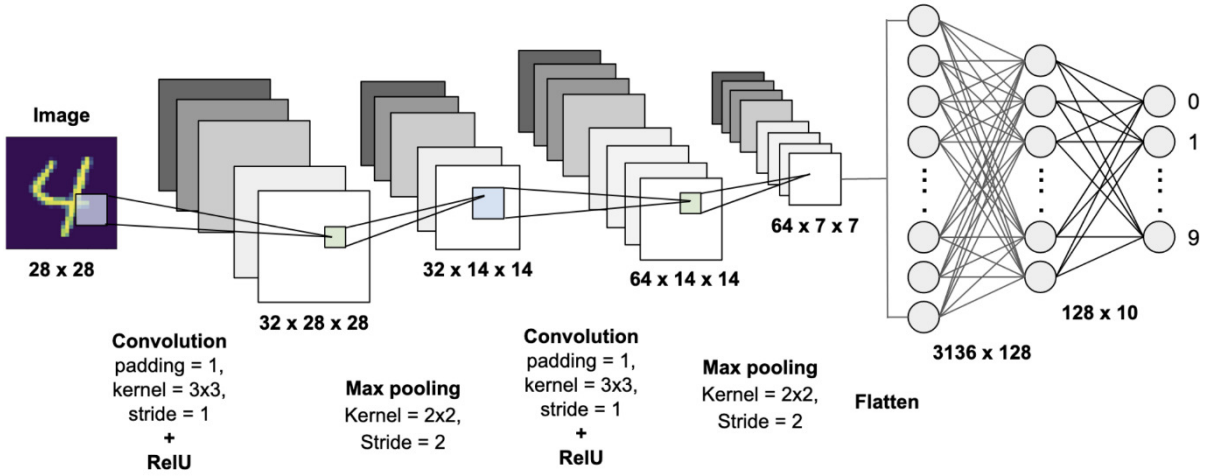
Další úlohou počítačového vidění je detekovat v obrazových datech polohu a třídu vybraných objektů. Na rozdíl od klasifikace s lokalizací je zde možný výskyt více objektů stejné nebo různé třídy na jednom obrázku. Jedná se o úlohu vhodnou pro účely této práce. Pokud jsou obrazová data snímána v nekontrolovaném venkovním prostředí a hledané objekty jsou kategorizovány poněkud abstraktně (automobil, chodec, budova apod.) je řešení této úlohy komplexním problémem, který v současné době vede k rozvoji umělé inteligence. Používány jsou zde algoritmy supervizorovaného strojového učení a architektury hlubokých konvolučních neuronových sítí, které jsou schopné extrahovat potřebné příznaky. Pokud je však zapotřebí určit polohu objektů, které mají významnou geometrickou podobnost a vstupní obrazová data jsou snímána z konzistentní scény bez rušivých vnějších vlivů, lze vyhovujících výsledků dosáhnout pomocí korelační analýzy. Při tomto přístupu k detekci objektů se porovnávají vstupní data s předem určeným vzorkem a jsou nalezena místa na obrázku s vysokou mírou shody. Výhodou tohoto přístupu je oproti algoritmům strojového učení značná jednoduchost. Není zde navíc zapotřebí provádět sběr dat a proces učení.

Segmentace instancí

Tato úloha přidává k detekci objektů navíc požadavek na označení všech pixelů, které objektu přísluší. Tato dodatečná informace by pro účely této práce přinášela řešení problému natočení polotovarů a bylo by možné jednoduše určit polohu jejich os. Tato úloha však představuje značné navýšení komplexity problému a existující architektury neuronových sítí, které umožňují segmentaci instancí jsou v porovnání s architekturami pro detekci objektů značně výpočetně náročnější a dosahují výrazně nižších rychlostí detekce. K problému natočení polotovarů je tak přistoupeno jiným řešením a tato úloha není pro účely této práce použita.

3.2.2 Konvoluční neuronové sítě

Nejmodernějším nástrojem pro řešení výše uvedených úloh strojového vidění jsou konvoluční neuronové sítě. Princip jejich fungování lze nejsnáze popsat na základní klasifikační úloze. Úkolem je zde určit třídu pro vstupní obrazová data, tedy jaká číslice od jedné do devíti se nachází na obrázku. Jako zdroj obrazových dat je použit MNIST dataset ručně psaných číslic [6], jehož ukázka je na Obrázku 5.



Obrázek 5 – Architektura konvoluční neuronové sítě pro klasifikaci ručně psaných číslic [8]

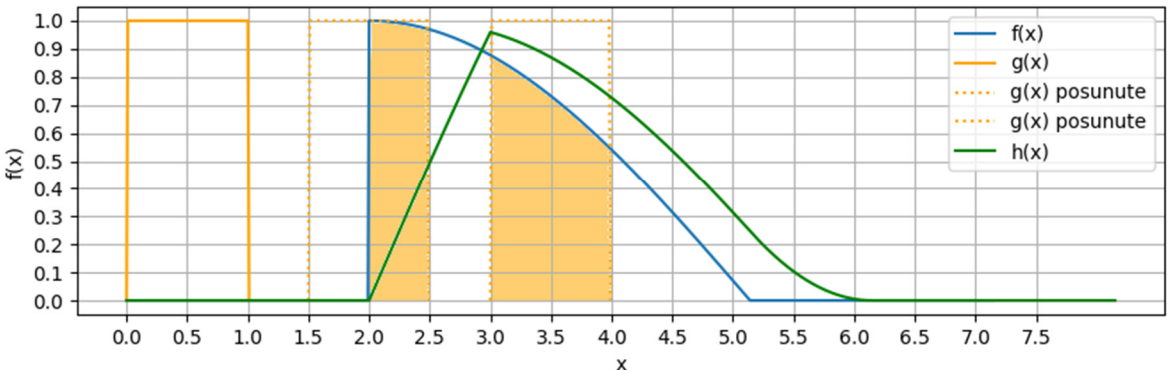
Vstupní obrazová data ve tvaru 2D matice vstupují nejprve do konvolučních vrstev, které extrahují z obrazových dat nich důležité příznaky, a pooling vrstev, které snižují rozměr dat při zachování co největšího množství informací. Poté jsou 2D obrazová data transformována v tzv. flatten vrstvě na jednorozměrný vektor a následně vstupují do vícevrstvé neuronové sítě se skrytými vrstvami (MLP). Výsledkem dopředné propagace je aktivace neuronů na výstupní vrstvě. Výstupní vrstva obsahuje 9 neuronů, kdy aktivace každého z nich koresponduje s pravděpodobností detekce jedné detekční třídy, tedy číslice 0 až 9. Při znalosti správného výsledku pro konkrétní vstupní data je možné určit chybu a její funkci, pomocí které lze principem zpětné propagace iterativně aktualizovat váhy a ostatní parametry sítě pro dosažení vyhovujících výsledků. Jednotlivé elementární operace, které se v tomto procesu odehrávají, budou dále popsány v následujících podkapitolách.

Konvoluce

Pojem konvoluce se objevuje již před rozvojem konvolučních neuronových sítí a to v oblasti zpracování signálů. V této oblasti jsou zpracovávána převážně jednorozměrná data, typicky průběh určité veličiny v čase. Ve zkratce zde jednorozměrná konvoluce znamená vyhodnocení, jak je tvar jedné funkce ovlivněn tvarem jiné funkce. Tato operace se značí symbolem $*$ a pro funkci $f(x)$ a konvoluční jádro $g(x)$ je definována dle [7] podle vztahu

$$h(x) = f(x) * g(x) = \int_{-\infty}^{\infty} f(u) \cdot g(x-u)du\,. \tag{1}$$

Graficky lze výsledek konvoluce určit tak, že konvoluční jádro $g(x)$ považujeme za jakési posuvné okénko, které posouváme ve směru osy x a hodnota $h(x)$ je rovna obsahu plochy vymezené průnikem grafů funkcí $g(x)$ a $f(x)$.

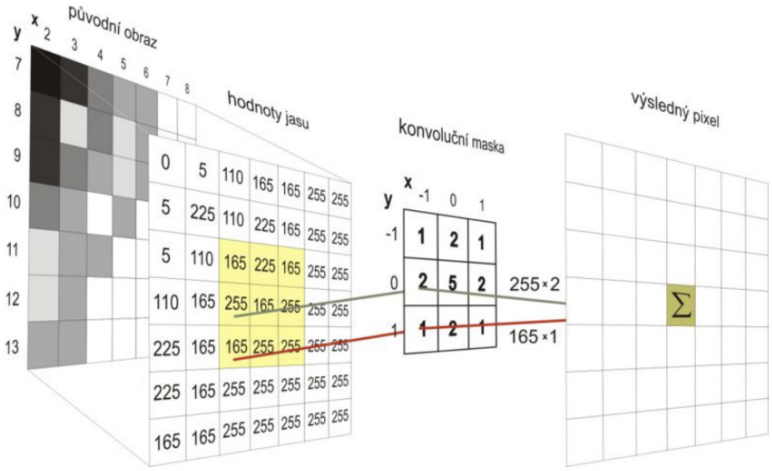


Obrázek 6 – grafické znázornění jednorozměrné konvoluce

Pro konkrétní polohu v ose x je tedy hodnota konvoluce rovna součtu všech součinů funkce s konvolučním jádrem postupně po infinitezimálních krocích pro celou šířku definičního intervalu. Nekonečně malý krok du lze nahradit krokem o reálném rozměru a lze tak definovat diskrétní konvoluci, kde integrace je nahrazena sumací podle vztahu

$$h_n = (f * g)_n = \sum_{m=-\infty}^{\infty} f_m \cdot g_{n-m} \,. \tag{2}$$

Lze dokázat, že výpočet jednorozměrné konvoluce se rovná součinu Fourierových obrazů $F(f)$ a $G(f)$ a často se takto konvoluce provádí za účelem zrychlení jejího numerického výpočtu.



Obrázek 7 – grafické znázornění 2D konvoluce [49]

V oblasti zpracování obrazu mají však data z pravidla vícedimenzionální charakter. Pro obraz ve stupních šedi lze uvažovat 2D matici, kdy hodnota každého elementu odpovídá hodnotě světlosti pixelu na obrázku.

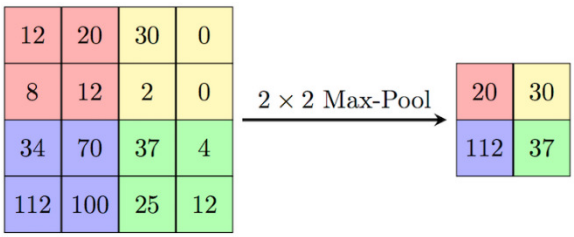
Diskrétní konvoluce dvoudimenzionálních dat je definována analogicky podle vztahu

$$h_{x,y} = (f * g)_{x,y} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f_{i,j} \cdot g_{x-i,y-j} \,. \tag{3}$$

Konvoluční jádro neboli maska je analogicky k předchozímu případu posouvána po definičním oboru funkce, která je v případě zpracování obrazu dvourozměrnou maticí. Výsledkem 2D konvoluce je dvourozměrná matice, kdy každý element je roven součtu součinů všech elementů konvoluční masky s příslušným elementem vstupní matice. Tento proces je graficky znázorněn na Obrázku 8. Proces konvoluce je často prováděn na vstupních obrazových datech několikrát s různými konvolučními maskami a je tak z jednoho vstupního obrázku vygenerováno několik příznakových map, které jsou dále zpracovávány viz Obrázek 6 – Architektura konvoluční neuronové sítě pro klasifikaci ručně psaných číslic [8]. Důležitým parametrem konvoluce je krok (z angl. stride), který určuje velikost posunutí konvoluční masky při jedné iteraci. Standartně má konvoluce krok 1 a při rozměru vstupních dat $N \times N$ tak mají výstupní data mají rozměr $(N-1) \times (N-1)$.

Max-pooling

V informatice je technika pooling (anglicky sdružování) používána k extrakci nejdůležitějších příznaků z dat při snižování jejich rozměru (rozlišení). Oba výsledky této operace jsou velmi vhodné pro zpracování obrazových dat před vstupem do neuronových sítí. Při snižování rozlišení obrazu se kvadraticky snižuje výpočetní náročnost, a je tak možné dosahovat přijatelných výpočetních časů. Neméně důležitá je však extrakce důležitých příznaků, která poskytuje do jisté míry abstraktní pohled na data a eliminuje tzv. over-fitting. Výpočet Max-pooling lze popsat na následujícím příkladu matice 4×4 při velikosti kroku (stride) 2.



Obrázek 8 –Příklad výpočtu Max-pooling [50]

Vstupní matice je rozdělena na submatice podle velikosti kroku, v tomto případě na matice 2×2. Pro každou submatici je určen maximální prvek, který definuje příslušný prvek výstupní matice. Výstup má rozměr 2×2 a při kroku 2 tak dochází ke změně rozměru dat na $N/2 \times N/2$. Ze submatic lze určit příslušný prvek výstupu i jinými způsoby, například minimem (Min-pooling) či aritmetickým průměrem (Average-pooling), avšak nejčastěji používaný je Max-pooling. Podstatným rozdílem oproti konvoluci, je absence číselných parametrů masky. Pooling vrstvy konvolučních neuronových sítí tedy vždy provádějí stejnou operaci nehledě na změnu vnitřních parametrů sítě [8].

MLP

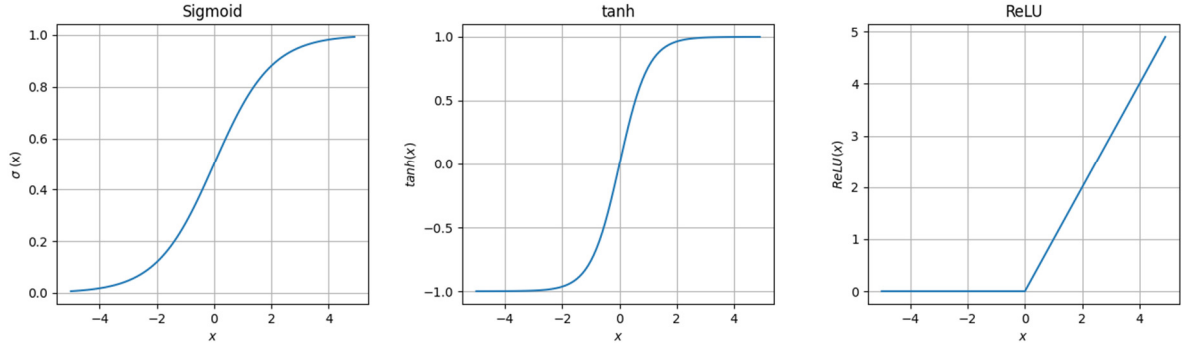
MLP (Multilayer perceptron) je označení vícevrstvou síť neuronů (perceptronů) a jedná se o základní architekturu umělých neuronových sítí. Vrstvu neuronové sítě tvoří množina neuronů, které se aktivují v jedné iteraci výpočtu společně. Jednotlivé vrstvy neuronů jsou plně propojeny. To znamená, že vstupem každého neuronu je vážený součet aktivací všech neuronů předchozí vrstvy. Aktivace každého neuronu je získána pomocí jeho aktivační funkce, jejímž vstupem je tento vážený součet. Neurony na vstupní vrstvě jsou aktivovány přímo vektorem vstupů. Schéma

této neuronové sítě společně s principem výpočtu jejího výstupu a učení je blíže popsáno na příkladu v Kapitole 4.3 [9].

Aktivační funkce

Výstup neboli aktivace neuronu není většinou definována pouze váženým součtem vstupů, ale výsledek tohoto součtu je nejprve vstupem do aktivační funkce neuronu. Důvodem může být například požadavek, aby výstup neuronu byla omezená funkce. Toho lze dosáhnout použitím aktivační funkce sigmoid (4) nebo hyperbolický tangens (5).

$\sigma(x) = \frac{1}{1+e^{-x}}$ (4)	$tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$ (5)	$ReLU(x) = max(0,x)$ (6)



Obrázek 9 – Porovnání základních aktivačních funkcí

Další aktivační funkcí, která se hojně používá v hlubokých konvolučních neuronových sítích je rektifikovaná aktivace ReLU (Rectified linear unit). Pokud je vstup záporný, vrací tato funkce nulovou hodnotu a při kladném vstupu je výstup lineární závislostí.

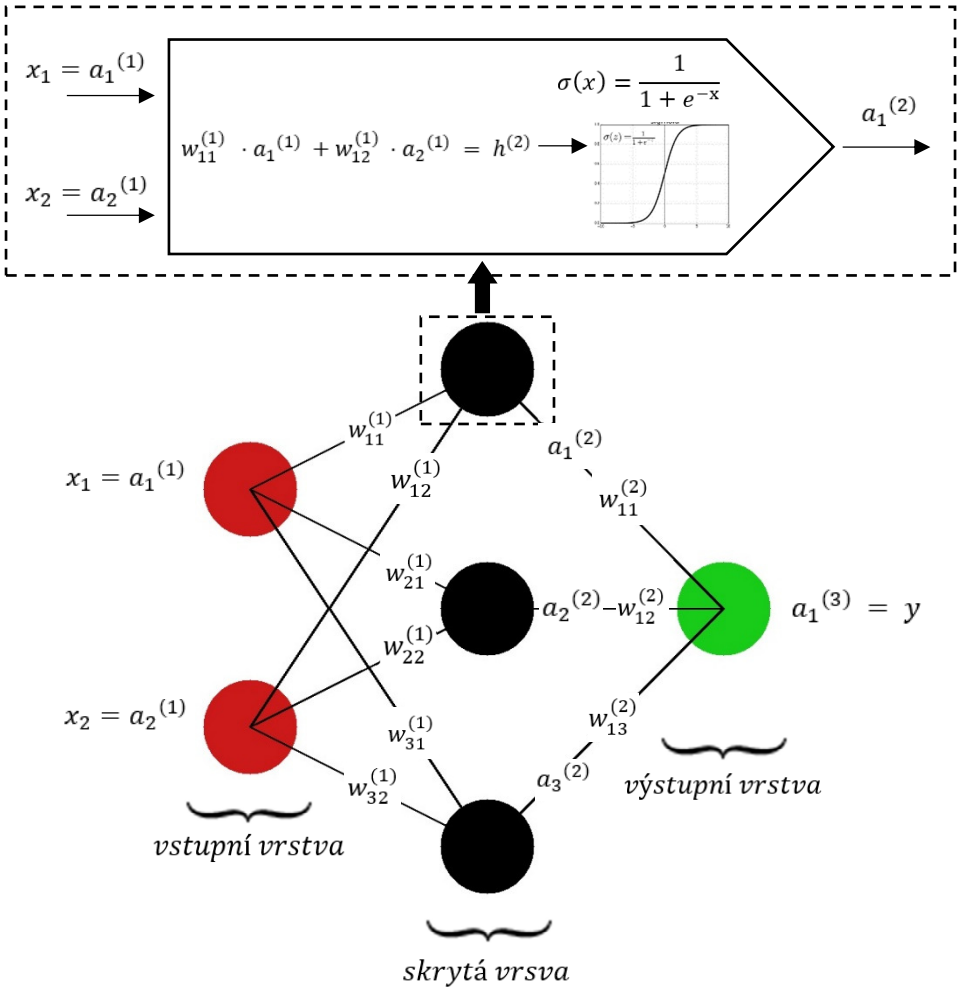
Supervizorované učení

Zvolenou architekturu neuronové sítě je pro dosažení korektních výsledků zapotřebí natrénovat na klasifikaci či detekci požadovaných objektů pomocí dostatečného množství anotovaných dat. Anotace znamená označení příslušnou třídou pro klasifikaci či označení polohy jednotlivých tříd ohraničujícím rámečkem pro detekci. Vstupní data jsou rozdělena na trénovací a testovací data ve vhodném poměru, například 85/15. Samotný proces trénování, tedy supervizorovaného učení byl již nastíněn. Váhy jednotlivých neuronů jsou iterativně aktualizovány tak, aby byl zajištěn nejvyšší pokles chybové funkce. Váhy jsou aktualizovány postupně od výstupní vrstvy pozpátku k první vrstvě tzv. zpětnou propagací. Trénink je zastaven při dosažení vyhovující průměrné přesnosti detekce na testovacích datech. Kvalitní výsledky lze dosáhnout za pomoci vhodné kombinace architektury sítě, učícího algoritmu a v neposlední řadě dostatečně velkou sadou trénovacích a testovacích dat, kterou lze navíc s výhodou uměle rozšířit tzv. augmentací. Při tomto procesu jsou vstupní obrazová data uměle zatížena šumem, zrcadlově převrácena nebo jinak transformována. Augmentace zabraňuje jevu tzv. over-fitting, tedy natrénovaní na specifika a nedokonalosti trénovacího datasetu namísto podstatných příznaků, které jsou obecně platné pro třídu detekovaných objektů. Architektur a optimalizačních algoritmů existuje celá řada. Vybrané moderní architektury pro řešení problematiky detekce objektů budou popsány dále v Kapitole 5.3. Pro optimalizaci učení jsou dnes často používány

sofistikované algoritmy jako např. ADAM [10] s proměnnou rychlostí učení (learning rate), který dosahuje minima chybové funkce velmi rychle. Základním algoritmem pro výpočet změn vah však zůstává tzv. Stochastic gradient descent (SGD), který bude pro vysvětlení základního principu supervizorovaného učení dále popsán na jednoduchém příkladu mělké neuronové sítě.

3.2.3 Příklad supervizorovaného učení mělké neuronové sítě

Princip supervizorovaného učení pomocí zpětné propagace s algoritmem gradient descent bude popsán na jednoduchém příkladu MLP s dvěma vstupy, jednou skrytou vrstvou o rozměru 3 a jedním výstupem.



Obrázek 10 – grafické znázornění mělké neuronové sítě MLP. Pro tvorbu schématu byla navržena aplikace v Pythonu s využitím knihovny OpenCV

Dopředná propagace

Výstup tedy aktivace $a_j^{(k)}$ jednotlivých neuronů k-té vrstvy je rovna výsledku funkce *sigmoid*

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad , \tag{7}$$

jejímž vstupem x je vážený součet aktivací předchozí vrstvy. Pokud označíme vektor všech aktivací v k-té vrstvě $\boldsymbol{a}^{(k)}$, platí pro jeho výpočet vztah pomocí maticového součinu

$$\boldsymbol{a}^{(k)} = \sigma\left(\boldsymbol{W}^{(\boldsymbol{k})} \cdot \boldsymbol{a}^{(k-1)}\right) \quad , \tag{8}$$

kde $\boldsymbol{W}^{(\boldsymbol{k})}$ je matice vah pro k-tou vrstvu. Takto lze postupovat od vstupní vrstvy dopředu až k výpočtu poslední, výstupní vrstvy. Jednotlivé váhy $w_{mn}^{(k)}$ jsou uspořádány do matice všech vah $\boldsymbol{W}$ následujícím způsobem. Pro každou vrstvu k existuje submatice vah $\boldsymbol{W}^{(\boldsymbol{k})}$ ve tvaru $m \cdot n$, kde m odpovídá počtu neuronů v aktuální vrstvě a n počtu neuronů v předchozí vrstvě. Matice vah má tedy následující tvar

$$\boldsymbol{W} = \left[\boldsymbol{W}^{(k=1)}\, \boldsymbol{W}^{(k=2)}\right] = \begin{bmatrix} \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \\ w_{31}^{(1)} & w_{32}^{(1)} \end{bmatrix} \begin{bmatrix} w_{12}^{(2)} \\ w_{12}^{(2)} \\ w_{13}^{(2)} \end{bmatrix} \end{bmatrix} . \tag{9}$$

Zpětná propagace

Pro učení, tedy iterativní změnu vah vedoucí k minimalizaci chybové funkce, je použit algoritmus gradient descent. Váhy jsou aktualizovány od poslední vrstvy postupně k první vrstvě. Pro poslední vrstvu lze vypočíst chybu e jako

$$e = y - \tilde{y} = a^{(3)} - \tilde{y} \quad , \tag{10}$$

kde y je výstup MLP, tedy aktivace výstupní vrstvy a $\tilde{y}$ je požadovaný výstup při zadaných vstupech. Chybová funkce je určena vztahem

$$Q = e^2 \, . \tag{11}$$

Váhy pro každou iteraci i jsou podle algoritmu SGD aktualizovány podle směru záporného gradientu chybové funkce Q

$$\boldsymbol{W_{i+1}} = \boldsymbol{W_i} - \frac{1}{2}\mu \, \nabla Q(\boldsymbol{W_i}) \, . \tag{12}$$

Změnu vah lze v poslední vrstvě vypočítat dle

$$\Delta \boldsymbol{W}^{(\boldsymbol{2})} = -\frac{1}{2}\mu \frac{\partial Q}{\partial \boldsymbol{W}^{(\boldsymbol{2})}} \tag{13}$$

a lze odvodit, že derivace chybové funkce podle vah se rovná

$$\frac{\partial Q}{\partial \boldsymbol{W}^{(\boldsymbol{2})}} = e^{(3)} \cdot \sigma'\left(h^{(3)}\right) \cdot a^{(2)} \quad , \tag{14}$$

kde $h^{(3)}$ je výstup neuronu před vstupem do funkce sigmoid, tedy

$$h^{(3)} = \boldsymbol{W}^{(\boldsymbol{2})} a^{(2)} \tag{15}$$

a σ' je derivace funkce sigmoid podle vztahu

$$\sigma'(x) = \sigma(x) - \left(1 - \sigma(x)\right) \tag{16}$$

a při znalosti

$$\sigma\left(h^{(3)}\right) = a^{(3)} \tag{17}$$

lze rovnici pro derivaci vah přepsat do tvaru

$$\frac{\partial Q}{\partial \boldsymbol{W}^{(2)}} = (a^{(3)} - \tilde{y}) \cdot (a^{(3)} - (1 - a^{(3)})) \cdot a^{(2)} . \tag{18}$$

Pro předcházející vrstvu 2 lze odvodit vztah derivaci chybové funkce podle vah následovně

$$\frac{\partial Q}{\partial \boldsymbol{W}^{(1)}} = e^{(3)} \cdot \sigma'\left(h^{(3)}\right) \cdot \boldsymbol{W}^{(2)} \cdot \sigma'\left(h^{(2)}\right) \cdot a^{(1)} \tag{19}$$

$$\frac{\partial Q}{\partial \boldsymbol{W}^{(1)}} = (a^{(3)} - \tilde{y}) \cdot (a^{(3)} - (1 - a^{(3)})) \cdot \boldsymbol{W}^{(2)} \cdot (a^{(2)} - (1 - a^{(2)})) \cdot a^{(1)} . \tag{20}$$

Pokud porovnáme rovnice pro výpočet derivací chybové funkce pro dvě po sobě jsoucí vrstvy

$$\frac{\partial Q}{\partial \boldsymbol{W}^{(k)}} = e^{(k+1)} \cdot \sigma'\left(h^{(k+1)}\right) \cdot a^{(k)} \tag{21}$$

$$\frac{\partial Q}{\partial \boldsymbol{W}^{(k-1)}} = e^{(k+1)} \cdot \sigma'\left(h^{(k+1)}\right) \cdot \boldsymbol{W}^{(k)} \cdot \sigma'\left(h^{(k)}\right) \cdot a^{(k-1)} \tag{22}$$

lze si všimnout opakujících se členů rovnice pro předchozí vrstvu. Pokud označíme

$$\delta^{(k)} = e^{(k+1)} \cdot \sigma'\left(h^{(k+1)}\right) \tag{23}$$

$$\frac{\partial Q}{\partial \boldsymbol{W}^{(k)}} = \delta^{(k)} \cdot a^{(k)} \tag{24}$$

$$\frac{\partial Q}{\partial \boldsymbol{W}^{(k-1)}} = \delta^{(k)} \cdot \boldsymbol{W}^{(k)} \cdot \sigma'\left(h^{(k)}\right) \cdot a^{(k-1)} \tag{25}$$

změny vah lze počítat iterativním postupem směrem od poslední k první vrstvě tak, že každou další iteraci nahradíme chybu

$$e^{(k+1)} \Rightarrow \delta^{(k)} \cdot \boldsymbol{W}^{(k)} . \tag{26}$$

Tímto způsobem vypočteme změny všech vah $\boldsymbol{\Delta W}$ pro i-tý datový bod (jeden obrázek při zpracování obrazu nebo časový okamžik ve zpracování signálu) a váhy jsou standartně aktualizovány podle

$$W_{i+1} = \boldsymbol{W}_i + \boldsymbol{\Delta W}_i . \tag{27}$$

Když jsou vyčerpána všechna trénovací data, je možné v učení pokračovat znovu na stejných datech v opakujících se cyklech, tzv Epochách. Proces trénování je zastaven při splnění zvoleného kritéria

$$Q_{i_{epoch}} < Q_{max} \tag{28}$$

[9] [11].

3.3 Detekce objektů

Jak již bylo naznačeno v předchozí kapitole, vyhovující řešení úlohy detekce objektů lze v určitých případech dosáhnout pomocí algoritmů korelační analýzy (Tepmlate matching), jejíž principy zde budou dále rozvedeny. Následně je provedena rešerše problematiky neuronových sítí pro detekci objektů a je vybrána vhodná architektura pro zadanou úlohu. Výhodou prvního přístupu oproti metodám strojového učení je jednoduchost jeho implementace a nižší výpočtová složitost. V současné době tyto nevýhody neuronových sítí však potlačují existující volně přístupné frameworky, které zjednodušují jejich implementaci a umožňují akceleraci výpočtů s využitím výkonných grafických procesorových jednotek.

3.3.1 Detekce hran 2D konvolucí

Základním krokem při řešení úlohy detekce objektů je detekce hran v obrazových datech. To se ve zpracování obrazu zpravidla provádí 2D konvolucí obrazových dat převedených z barevné škály RGB do stupňů šedi (grayscale). Při použití specifické konvoluční masky se zvýrazní pixely s vysokým gradientem světlosti, tedy hrany. V tomto případě byl použit Sobelův operátor. Ten používá dvě masky pro horizontální a vertikální směry gradientů. Gradienty získáme konvolucí (značeno *) následujících masek se vstupními daty

$$\boldsymbol{G_x} = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \cdot \boldsymbol{A}, \quad \boldsymbol{G_y} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ +1 & -2 & -1 \end{bmatrix} \cdot \boldsymbol{A} \tag{29}$$

a velikost gradientu pak získáme použitím

$$\boldsymbol{G} = \sqrt{\boldsymbol{G_x^2} + \boldsymbol{G_y^2}} \tag{30}$$

Výsledná hodnota velikosti gradientu pak každému pixelu obrázku přiřazuje světlost 0 až 255 (8bitový formát)

3.3.2 Template matching

Template matching je technika korelační analýzy pro nalezení oblastí ve zkoumaných datech, které mají určitou podobnost se zvoleným vzorkem. Při použití ve zpracování obrazu jsou data uspořádána do 2D matice představující jednotlivé pixely obrázku. Vzor se principem posuvného okénka posouvá po zkoumaném obrázku a každé pozici umístění vzoru na zkoumaný obrázek se přiřadí hodnota korelačního koeficientu a vytvoří se tak matice $\boldsymbol{R}$ (result). Existují různé způsoby výpočtu $\boldsymbol{R}$ a zvolena byla metoda normovaných korelačních koeficientů dle následující rovnice.

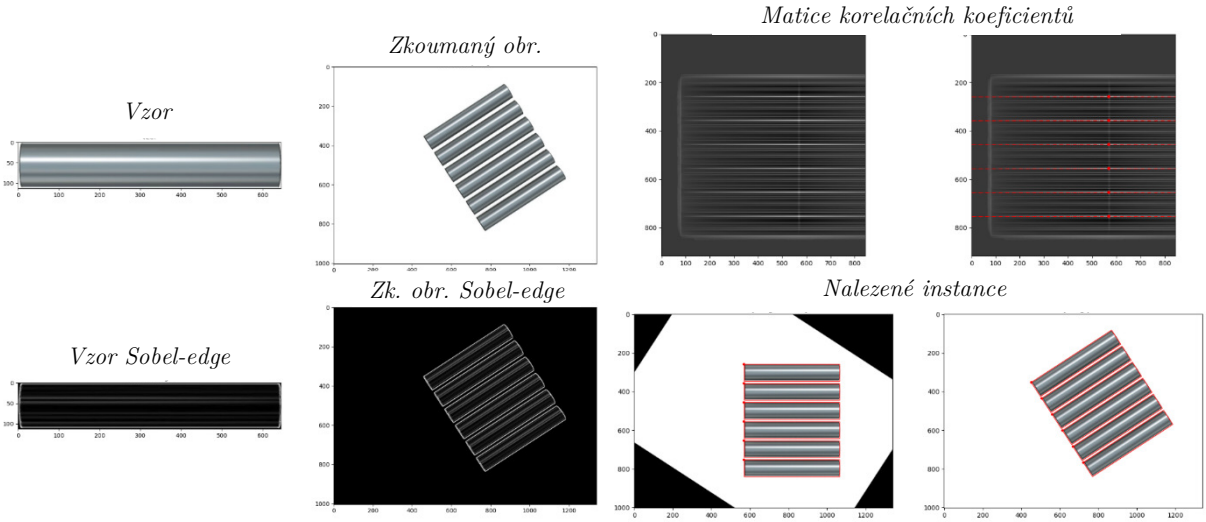
$$\boldsymbol{R}(x,y) = \frac{\sum_{x',y'}\left(\boldsymbol{T}'(x',y') \cdot \boldsymbol{I}'(x+x',y+y')\right)}{\sqrt{\sum_{x',y'} \boldsymbol{T}'(x',y')^2 \cdot \sum_{x',y'} \boldsymbol{I}'(x+x',y+y')^2}}, \tag{31}$$

kde x a y jsou pozice jednotlivých prvků 2D matice, které odpovídají příslušným pixelům, $\boldsymbol{T}$ (template) je matice detekovaných hran vzoru podle Kapitoly 3.1 a $\boldsymbol{I}$ (image) **je** obdobná matice pro zkoumaný obraz, $\boldsymbol{T'}$ **a** $\boldsymbol{I'}$ jsou standardizované matice, které jsou získány podle rovnic

$$\boldsymbol{T}'(x', y') = \boldsymbol{T}(x', y') - \frac{\sum_{x'', y''} \boldsymbol{T}(x'', y'')}{(w \cdot h)} \tag{32}$$

$$\boldsymbol{I}'(x + x', y + y') = \boldsymbol{I}(x + x', y + y') - \frac{\sum_{x'', y''} \boldsymbol{I}(x + x'', y + y'')}{(w \cdot h)}, \tag{33}$$

kde w a h jsou rozměry vzorové matice $\boldsymbol{T}$. Pro získání matice $\boldsymbol{R}$ lze použít funkci *cv.matchTemplate()* knihovny Open CV [12] a získanou matici lze vykreslit jako obrázek ve stupních šedi tak, že každé její hodnotě je přiřazena světlost tak, že čím je vyšší hodnota korelačního koeficientu tím je vykreslen světlejší pixel. Tento proces je iterativně proveden pro různá zvětšení a zmenšení vzoru a je uchovávána hodnota dosaženého maxima. Pro maximum korelačního koeficientu je matice **R** vykreslena a poloha maxima, která odpovídá skutečné poloze hledaného objektu je označena obdélníkem ve tvaru vzoru v odpovídajícím měřítku. Matici korelačních koeficientů lze dále zpracovávat pomocí prahování pro extrakci všech instancí a výsledek detekce může vypadat následovně.

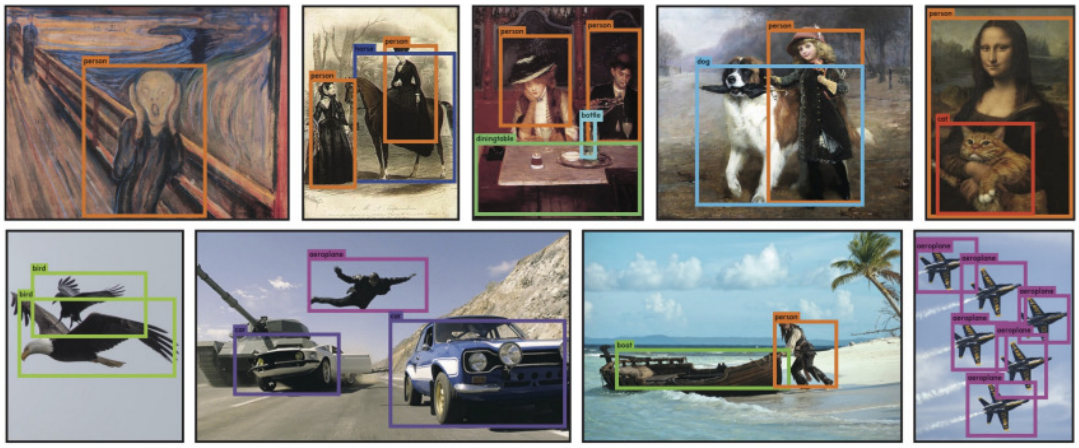


Obrázek 11 – Příklad použití metody template matching na uměle vytvořených datech

Pokud je zadáním úlohy detekce objektů s výraznou geometrickou podobností, může mít tato metoda zcela uspokojivé výsledky a není zapotřebí používat složitější metody využívající hlubokých neuronových sítí.

3.3.3 Hluboké konvoluční neuronové sítě pro detekci objektů

Moderním přístupem k řešení úlohy detekce objektů je využití supervizorovaného učení hlubokých konvolučních neuronových sítí (CNN). Tento přístup se hojně využívá v úlohách, kde je zapotřebí detekovat velké množství různých tříd objektů, může ale být s výhodou použit i pro detekci více instancí jedné třídy. Velikost vstupů a celková složitost konvolučních neuronových sítí pro detekci objektů znamená vysokou výpočetní náročnost a jejich rozvoj pro využití v reálném čase je tak možný až díky nárůstu výpočetního výkonu a vývoji efektivnějších architektur v posledních letech.



Obrázek 13 - Příklady použití CNN pro detekci objektů [13]

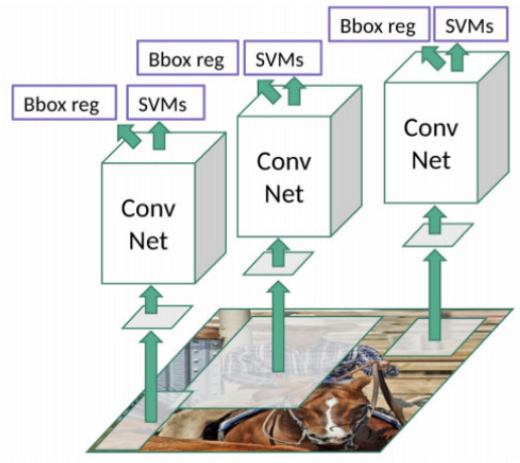
Přístup k řešení problému je zde poněkud jiný než v předchozím případě korelační analýzy a vychází z řešení klasifikační úlohy, které popsáno v Kapitole 4.2 o konvolučních neuronových sítích. Neuronové sítě pro detekci objektů jsou však oproti uvedenému příkladu složitější a obsahují mnoho desítek skrytých vrstev. Je zde zapotřebí většího množství vstupů při učení a výstupů při predikci nejen pro třídu objektů, ale také pro polohu ohraničujícího rámečku a to pro n nalezených instancí.

R-CNN

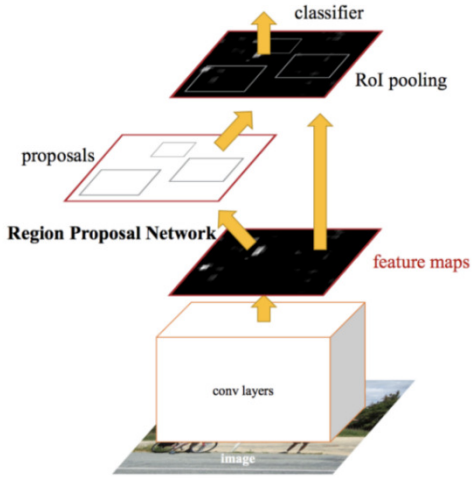
V roce 2014 bylo navrženo řešení pomocí architektury R-CNN [14], které řeší problém obrovského množství detekčních oblastí v obrazovém vstupu klasické konvoluční neuronové sítě. Je zde použito algoritmu selektivního vyhledávání pro získání přibližně 2000 regionů, ve kterých probíhá detekce, viz Obrázek 8. Tato architektura představovala obrovský posun v přesnosti a rychlosti detekce, avšak stále nelze hovořit o real-time detekci při průměrné době jedné detekce 47 sekund (na dobovém hardware) [15].

Fast R-CNN

Výrazný posun k real-time použitelnosti přinesla architektura Fast R-CNN [16], která detekci zefektivňuje pomocí mapy charakteristik, která je vytvořena konvolucí vstupního obrazu, a ze které jsou pomocí selektivního algoritmu navrhovány detekční regiony.



Obrázek 12 – Architektura R-CNN [15]



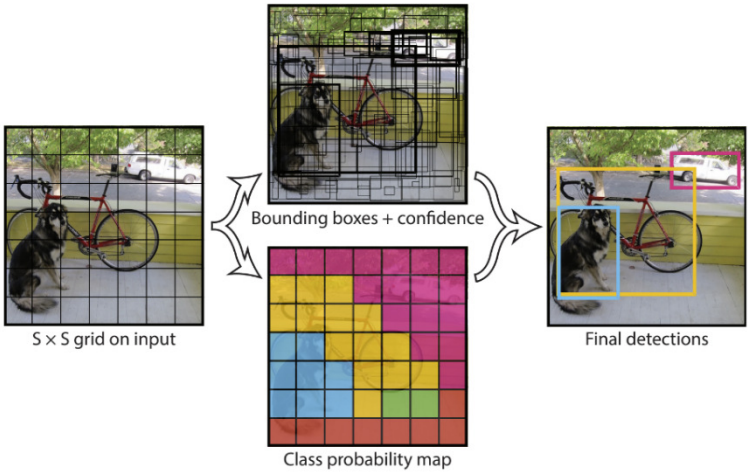
Obrázek 13 – Architektura Faster R-CNN [18]

Faster R-CNN

Dále architektura Faster R-CNN [17] zcela elimininuje selektivní algoritmus a detekční regiony jsou navrhovány separátní neuronovou sítí, viz Obrázek 7. Nejnovější iterací této architektury je Mask R-CNN [18] publikovaná v roce 2017, která predikuje navíc masku pixelů, které přísluší jednotlivým detekovaným objektům.

YOLO

Doposud popsané architektury využívají koncept regionů v obrazových datech, které s určitou pravděpodobností obsahují objekty a v nich následně probíhá samotná detekce objektů. Zcela jiným přístupem řeší problém architektura YOLO (You Only Look Once) [13] publikovaná v roce 2016. Jediná konvoluční neuronová síť zde tvoří predikce tříd a jejich rámečků.

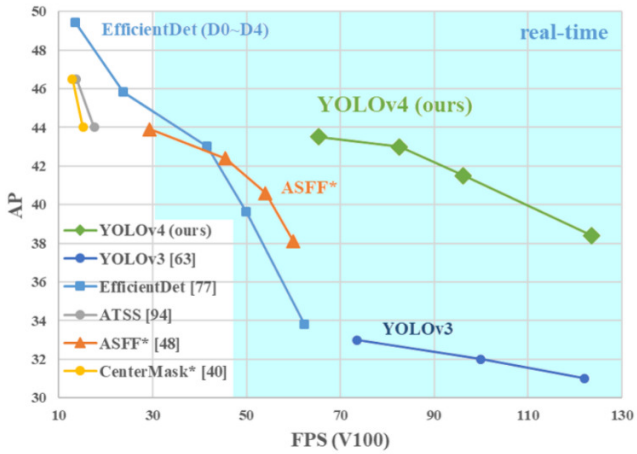


Obrázek 14 – princip detekce architektury YOLO [14]

Vstupní obraz je rozdělen na množinu SxS polí a pokud se střed objektu nalézá v daném poli, je pro tento objekt predikováno B rámečků s pravděpodobnostmi výskytu jednotlivých tříd. Finální detekce je dosažena zúžením výsledků na ty, které přesahují danou prahovou pravděpodobnost a v daném poli mají nejvyšší poměr IoU (Intersection Over Union), který vyjadřuje poměr mezi překrytím rámečku objektem a nepřekrytou oblastí.

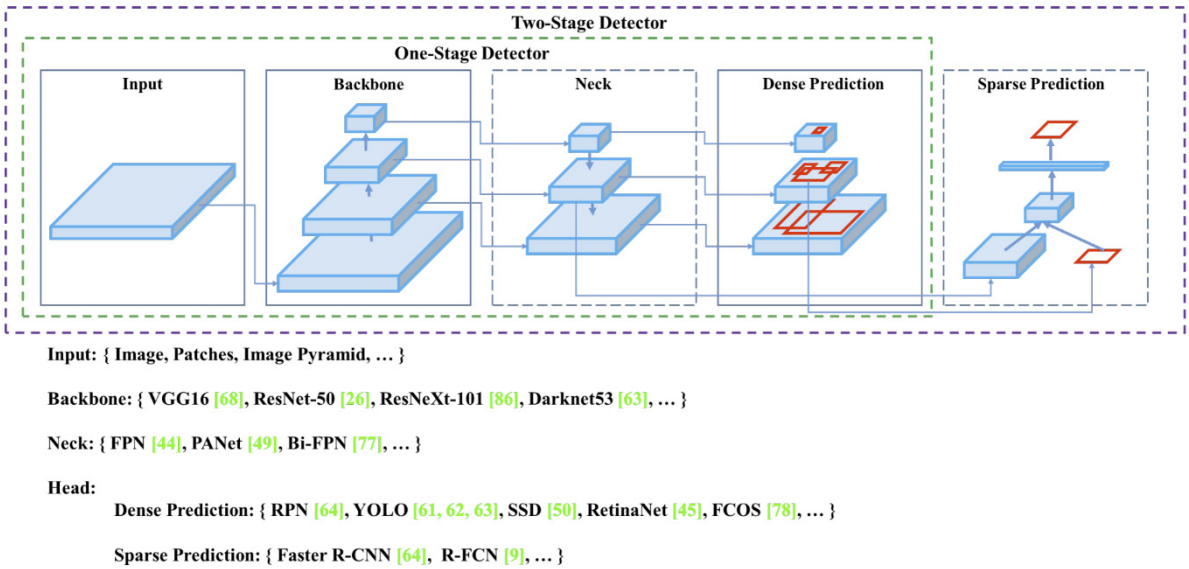
YOLO v4

Čtvrtou iterací této architektury je YOLOv4 [19] publikovaná v roce 2020, která posouvá rychlost a přesnost detekce objektů blíže k optimu. Lze zde s přehledem hovořit o real-time detekci při 65 snímcích za sekundu (FPS) na grafické kartě Tesla V100 při dosažení průměrné přesnosti (AP) 65.7% pro MS COCO dataset.



Obrázek 15 – FPS / AP porovnání nejmodernějších detektorů [14]

Tato architektura je pro její výborný poměr rychlosti detekce a průměrné přesnosti v současnosti jednou z nejpoužívanějších pro širokou škálu úloh.



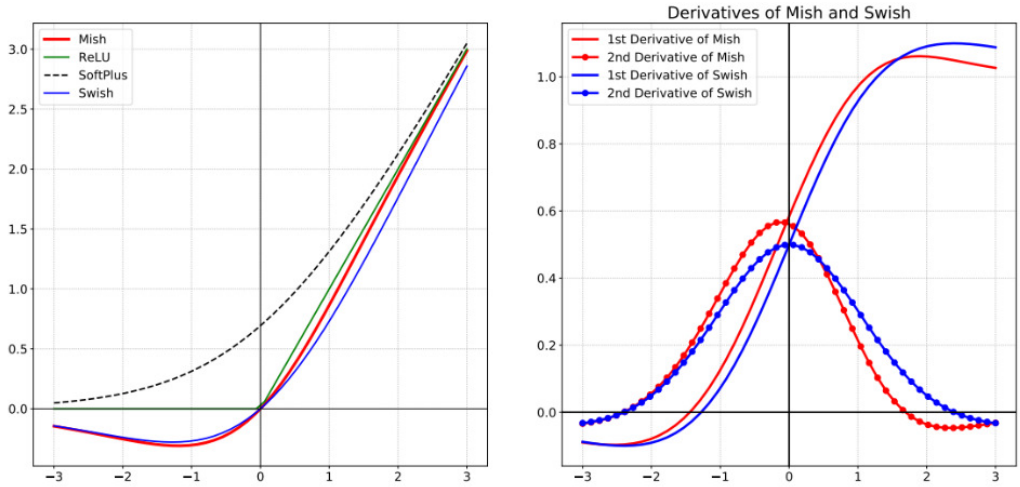
Obrázek 16 – Různé možnosti složení architektury detektoru objektů z jednotlivých bloků [20]

3.3.4 Architektura detektoru objektů YOLO v4

Při návrhu moderní architektury konvoluční neuronové sítě pro detekci objektů (detektor objektů), jakou je například YOLO v4, se nepostupovalo od základu, ale je zvoleno složení a uspořádání jednotlivých úseků z existujících otestovaných funkčních bloků. Základní rozdělení těchto bloků může být na backbone (páteř), neck (krk) a head (hlava) jak vyobrazují autoři YOLO v4 na Obrázku 18. Backobone konvoluční neuronové sítě je ta část obsahující konvoluční vrstvy, která je zodpovědná za extrakci příznakových map. Neck je potom část, kde dochází k vhodné kombinaci příznakových vrstev, a v části head dochází k samotné detekci. Detektory objektů mohou být dvoustupňové či jednostupňové. Dvoustupňové detektory (např. Faster R-CNN) oddělují úlohu lokalizace objektů, při které se predikují ohraničující rámečky, a klasifikaci třídy nalezených objektů do dvou separátních kroků. YOLO je jednostupňovým detektorem a tyto úlohy jsou vykonávány současně. Na základě experimentálně dosažených výsledků byla architektura YOLO v4 vytvořena z následujících částí.

- Backbone: CSPDarknet53 [20]
- Neck: SPP (Spatial pyramid pooling) [21], PAN (Path aggregation network) [22]
- Head: YOLOv3 [23]

K samotné detekci je použit stejný princip jako v předchozích verzích architektur YOLO, který je stručně popsán v předchozí kapitole. Dále jsou pro architekturu neuronové sítě zvoleny další klíčové parametry a vlastnosti detektoru objektů. Zaprvé se jedná o způsoby manipulace se vstupními daty pomocí augmentace dat (CutMix [24], Mosaic), tvar ztrátové funkce, regulace dat (DropBlock [25]), vyhlazování označení tříd a další specifika, která se dohromady označují jako tzv. Bag of Freebies (BoF). Dalšími specifiky jsou tvar aktivační funkce (Mish [26]), částečné propojení mezi úseky (CSP [20]) a další, které spadají pod označení a Bag of Specials (BoS) [19].



Obrázek 17 – Porovnání aktivačních funkcí Mish a Swish a její derivací [27]

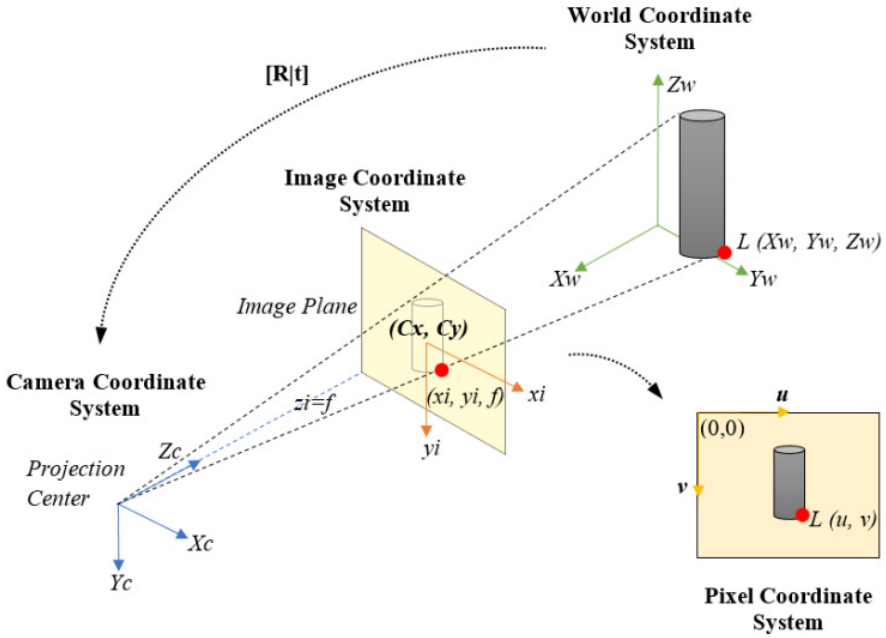
3.4 Registrace obrazů

Registrace obrazů je základní úlohou zpracování obrazu a častý problém aplikací fúze senzorů. Cílem registrace obrazů je transformovat obrazová data tak, aby se shodovala pozice odpovídajících bodů na obrázcích pořízených v různých časech, z různých snímacích systémů či systémů s různou prostorovou orientací a bylo tak dosaženo tzv. pixel-to-pixel korespondence. Běžně se k řešení této úlohy používá metoda projektivní transformace také nazývaná homografie. Tato metoda ze zadaných bodů určí transformační matici, pomocí které následně lineárně transformuje obraz tak, jako by se měnila perspektiva pozorovatele. Touto metodou však není možné dosáhnout dokonalé registrace ve všech rovinách 3D prostoru [27].

Dosažení přesné registrace obrazů v různých rovinách 3D prostoru pořízených z odlišných snímacích systémů viditelného a infračerveného spektra není triviálním problémem a výzkum ideálního řešení stále probíhá [28]. Důkladný průzkum současných metod je uveden například v práci [29]. Díky zvolnému uspořádání snímacího systému, které do jisté míry zjednodušuje problém na rovinný, lze však s přijatelnou nepřesností pro účely této práce použít metodu jednoduché homografie. Při použití této metody je však zásadní nejprve provést kalibraci snímacích systémů. Účelem kalibrace je odstranit optické zkreslení způsobené konstrukcí snímače, které způsobuje zakřivení ve skutečnosti rovných čar. Míra a charakter zkreslení jsou pro každý snímač specifické a před výpočtem matice homografie je tak nutné vycházet z již kalibrovaných snímků.

3.4.1 Kalibrace

Obraz získaný jako snímek z kamery je chápán jako reprezentace bodů snímaného prostoru na obrazovou rovinu. K matematickému popisu tohoto principu se nejčastěji používá optický



Obrázek 18 – pinhole camera model [51]

model dírkové kamery (*pinhole camera model*), kde se zavádí vztahy mezi souřadnicovým

systémem kamery O_{camera} a jeho projekcí na obrazovou rovinu O_{image}, přičemž se vychází ze souřadnicového systému snímaného prostoru O_{world} [30]

$$O_{camera} = [\boldsymbol{R}|\boldsymbol{t}] \cdot O_{world} \tag{34}$$

$$O_{image} = \boldsymbol{K} \cdot O_{camera}\,. \tag{35}$$

Pro přechod ze souřadného systému snímané scény na souřadný systém kamery je zavedena transformační matice $[R|t]$, pro kterou platí

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \boldsymbol{R}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \boldsymbol{t}\,, \qquad \boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}. \tag{36}$$

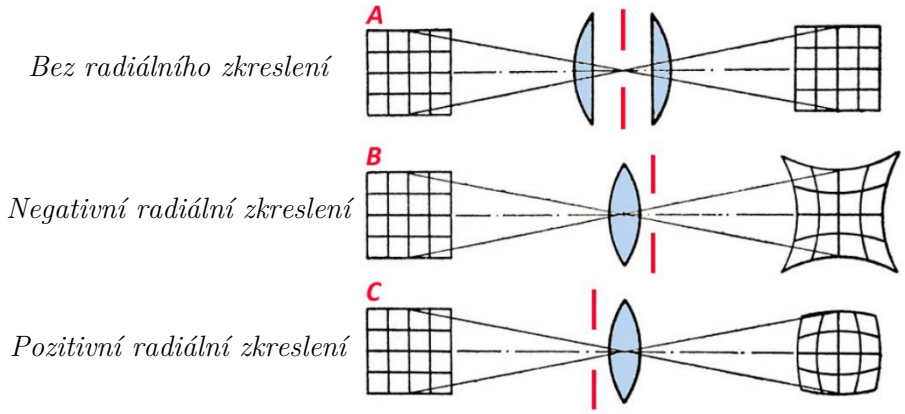
Snímač samotný je zde uvažován jako jediný bod (ohnisko) bez přídavných optických prvků. Základními parametry tohoto modelu jsou ohniskové vzdálenosti f_x, f_y a poloha středu kamery c_x, c_y. Tyto parametry se nazývají vnitřní parametry kamery a jsou soustředěny do matice kamery K, pro kterou platí

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boldsymbol{K}\begin{bmatrix} x \\ y \\ z \end{bmatrix}, \qquad \boldsymbol{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{37}$$

Obě operace lze sloučit do jedné transformace a zavést tak matici projekce P, která je výsledkem maticového součinu obou transformačních matic. Vektor souřadnic snímaného prostoru je však zapotřebí rozšířit na rozměr 4x1 za účelem splnění podmínek maticového násobení.

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boldsymbol{P}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \qquad \boldsymbol{P} = \boldsymbol{K}\,[\boldsymbol{R}|\boldsymbol{t}]\,, \tag{38}$$

kde s je parametr zvětšení. Tato matice však sama o sobě nestačí k přesnému určení transformace snímaných bodů na obrazovou rovinu, protože nezohledňuje zkreslení. To je způsobeno přítomností čočky, který má významný vliv na výsledný obraz. Pro přesnější určení modelu reprezentace se zavádí radiální a tangenciální zkreslení.



Obrázek 19 - Radiální zkreslení [52]

Pro výpočet kompenzace těchto zkreslení se zavádí následující vztahy.

$$x'' = x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x' y' + p_2(r^2 + 2x'^2) \tag{39}$$

$$y'' = y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y'^2) + 2p_2 x' y' \tag{40}$$

$$r^2 = x'^2 y'^2, \tag{41}$$

$$x' = \frac{x}{z} \tag{42}$$

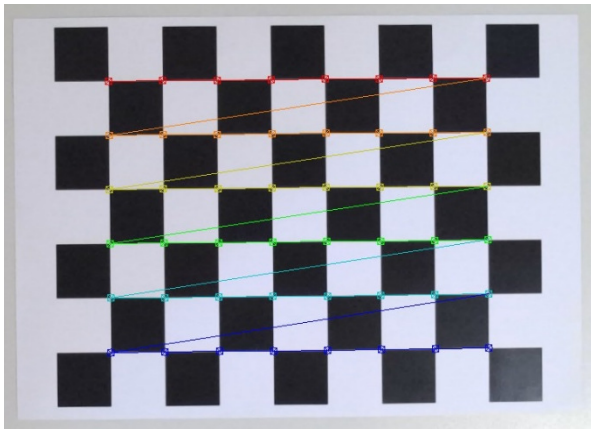
$$y' = \frac{y}{z} \tag{43}$$

kde k_1, k_2, k_3 jsou radiální koeficienty a p_1 *a* p_2 jsou tangenciální koeficienty deformace. Takto vypočtené souřadnice lze převést do souřadnic obrazu pomocí vztahů

$$u = f_x x'' + c_x \tag{44}$$

$$v = f_y y'' + c_y \,. \tag{45}$$

Kalibraci lze provést pomocí softwarových nástrojů jako *MATLAB* či *OpenCV,* které postupují při určování kalibrační matice podle metody, která byla publikována v práci [31] již v roce 1998. Základní myšlenkou je určení n dvojic odpovídajících bodů v 3D prostoru snímané scény a na obrazové rovině. Tyto tzv. korespondence lze určit manuálním zadáním těchto bodů, avšak nejčastěji se s výhodou využívají algoritmy, které jsou schopné nalézt významné body na obrázku automaticky. Typicky se pro určení korespondujících bodů využívá šachovnicového vzoru se známou konstantní délkou hran.



Obrázek 20 – fotografie šachovnicového vzoru s automaticky nalezenými body (OpenCV)

Důležitou vlastností soboru těchto bodů je, že leží v jedné rovině a jejich Z souřadnice je tak považována za nulovou. Tímto způsobem je obdržen soubor korespondujících bodů z několika snímků. Pro každý i-tý bod lze sestavit rovnici transformace pomocí hledané projekční matice

$$s\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}, \tag{46}$$

kterou lze po eliminaci parametru zvětšení s přepsat do tvaru

$$u_i(p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}) = p_{11}X_i + p_{12}Y_i + p_{13}Z_i + p_{14} \tag{47}$$

$$v_i(p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}) = p_{21}X_i + p_{22}Y_i + p_{23}Z_i + p_{24} \tag{48}$$

a je tak získána soustava dvou lineárních rovnic s dvanácti neznámými pro každý korespondující bod. Kalibrace se provádí pro několik snímků s n body a je tak zapotřebí řešit přeurčenou soustavu lineárních rovnic, kterou lze řešit například pomocí pseudoinverzní matice, kdy řešením je aproximace ve smyslu nejmenších čtverců.

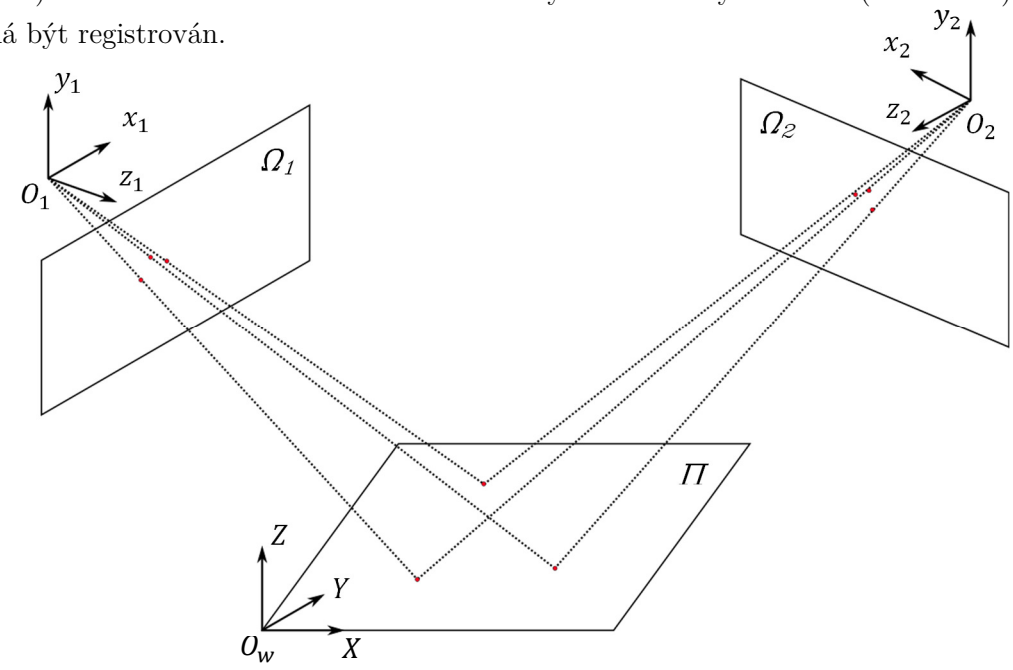
$$\boldsymbol{Ax = b}, \qquad \boldsymbol{A \in R^{m \times n} \wedge m > n} \tag{49}$$

$$\boldsymbol{x = A^T\left(AA^T\right)^{-1}b} \quad . \tag{50}$$

Získanou projekční matici lze rozložit podle uvedených vztahů a je tak možné získat vnitřní parametry kamery K, rotační a translační matici $[R|t]$ a koeficienty deformace k_1, k_2, k_3, p_1 a p_2. Tyto parametry lze zahrnout do tzv. kalibrační matice, pomocí které lze následně kalibrovat výstupní obraz [32; 33].

3.4.2 Homografie

Pojem homografie v projektivní geometrii vyjadřuje vzájemně jednoznačné zobrazení mezi dvěma projektivními prostory. Projektivní prostory mohou být snímaná rovina (na Obrázku 23 značená Π) a obrazová rovina nebo obrazové roviny dvou odlišných kamer (značené Ω), jejichž obraz má být registrován.



Obrázek 21 – Dvě kamery snímající scénu z různých perspektiv

Pro transformaci ze souřadného systému snímané scény do souřadného systému první kamery lze podle [34] psát

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \boldsymbol{H_1} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{51}$$

a nápodobně pro transformaci do souřadného systému druhé kamery

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \boldsymbol{H_2} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \tag{52}$$

Pro transformaci přímo mezi souřadnými systémy kamer lze tak psát

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \boldsymbol{H_2 H_1^{-1}} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \boldsymbol{H} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \quad , \tag{53}$$

kde $\boldsymbol{H}$ je matice homografie. Při rozepsání matice po elementech

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \tag{54}$$

a zavedení normalizovaných souřadnic, podobně jako v předchozí kapitole o kalibraci

$$x_2{}' = \frac{x_2}{z_2} \tag{55}$$

$$y_2{}' = \frac{y_2}{z_2} \tag{56}$$

lze psát

$$x_2{}' = \frac{H_{11}x_1 + H_{12}y_1 + H_{13}z_1}{H_{31}x_1 + H_{32}y_1 + H_{33}z_1} \tag{57}$$

$$y_2{}' = \frac{H_{21}x_1 + H_{22}y_1 + H_{23}z_1}{H_{31}x_1 + H_{32}y_1 + H_{33}z_1}. \tag{58}$$

Souřadnice z v tomto případě představuje faktor zvětšení a při jeho eliminaci zavedením $z_1 = z_2 = 1$ a přeskupení rovnic je získána následující soustava dvou rovnic

$$x_2{}'(H_{31}x_1 + H_{32}y_1 + H_{33}) = H_{11}x_1 + H_{12}y_1 + H_{13} \tag{59}$$

$$y_2{}'(H_{31}x_1 + H_{32}y_1 + H_{33}) = H_{21}x_1 + H_{22}y_1 + H_{23}\,, \tag{60}$$

kde neznámými jsou elementy matice homografie $\boldsymbol{h} = H_{11}, H_{12}, \ldots, H_{33}$. Tyto rovnice jsou pro neznáme koeficienty lineární a lze je přeskupit do homogenního tvaru

$$\boldsymbol{a_x h} = 0 \tag{61}$$

$$\boldsymbol{a_y h} = 0\,. \tag{62}$$

Analogicky ke kapitole o kalibraci, jsou tyto dvě rovnice jsou získány pro jeden korespondující bod ve snímcích z obou kamer. Pro sadu těchto korespondujících bodů tak lze získat přeurčenou soustava rovnic, kterou lze opět řešit například pomocí pseudoinverzní matice (50).

K získání matice homografie lze opět použít softwarové nástroje jako například *MATLAB* nebo *OpenCV* a manuálně či automaticky nalezenou sadu korespondujících bodů, která obvykle musí obsahovat minimálně 10 bodů. Moderní algoritmy pro výpočet homografie využívají

iterativní metodu RANSAC (Random sample consensus) [35], který eliminuje vliv tzv. outliers, tedy datových bodů významně odlišných od zbytku. Získanou matici homografie pak lze využít k transformaci obrazů a je tak dosažena jejich vzájemná registrace [36; 37; 38].

3.5 Použitý hardware a software

Nvidia Jetson Nano

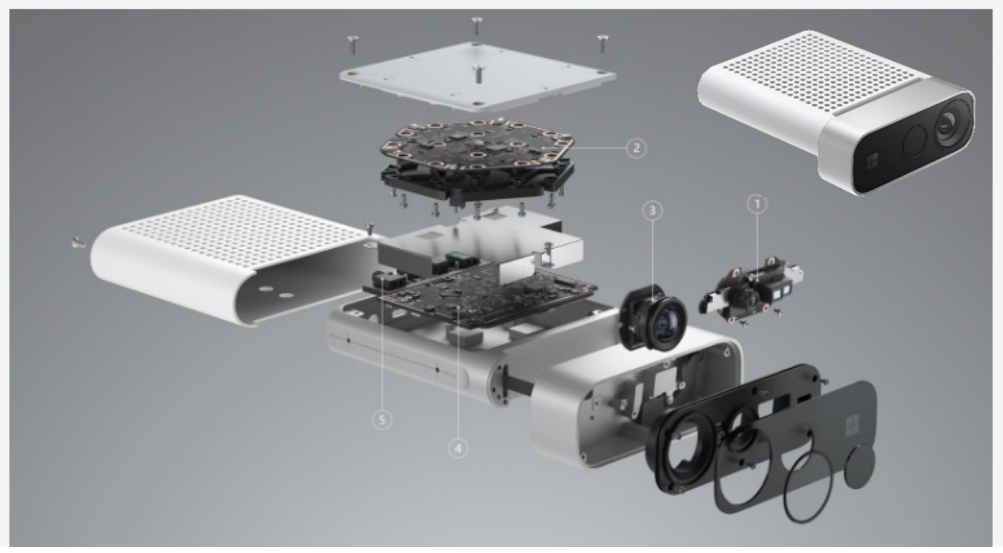
Jako nízkonákladový výpočetní hardware pro implementaci vytvořených softwarových modulů byl zvolen Nvidia Jetson Nano, který disponuje čtyřjádrovým procesorem architektury ARM o taktu 1.43 GHz a 128 jádrovým grafickým čipem MAXWELL s pamětí 4 GB.

Obrázek 22 - Nvidia Jetson Nano [53]

Výhodou této vývojové platformy je podpora architektury Nvidia CUDA (Compute Unified Device Architecture) pro paralelní akceleraci podporovaných knihoven a softwarových modulů strojového učení. Pro účely získání modelu strojového učení je však z důvodu urychlení použit stolní PC s výkonnou GPU.

Microsoft Azure Kinect

Tento komplexní senzor je vybaven gyroskopem a akcelerometrem (4), polem sedmi stereo-mikrofonů (2), ale pro tuto aplikaci především 12megapixelovou RGB kamerou (3) a



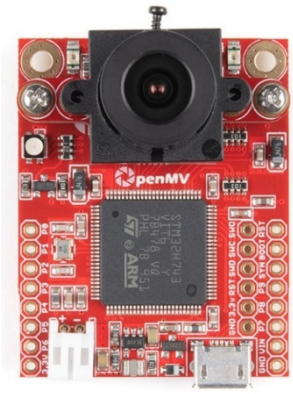
Obrázek 23 - Microsoft Azure Kinect [40]

1megapixelovým hloubkovým senzorem (1), který využívá vysílač a kameru ve spektru infračerveného záření (IR) a funguje na principu Time-of-Flight.

Hloubkový senzor lze provozovat v běžném režimu NFOW (Narrow Field of View) pro vzdálenosti 0.5 - 3.86 m a širokoúhlém režimu WFOW (Wide Field of View) pro vzdálenosti 0.25 - 2.21 m. Avšak výrobce udává, že operační vzdálenosti se mohou měnit pro předměty s různou odrazivostí. Výrobce dále udává systematickou chybu měření < 11 mm + 0.1% vzdálenosti [39].

OpenMV H7 camera

OpenMV je start-up projekt z roku 2015, který se zabývá výrobou cenově dostupných modulů pro aplikace strojového vidění a bývá nazýván „Arduino pro strojové vidění“. Jeho výhodou je operačním systémem na bázi MicroPython a je tak možné přímo na mikrokontroleru programovat aplikace v tomto vysokoúrovňovém jazyce. V této práci je konkrétně využit mikrokontroler OpenMV H7.

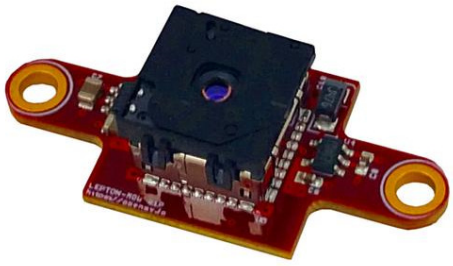


Obrázek 24 – OpenMV H7 Camera [47]

Pro účely této práce slouží OpenMV mikrokontroler pouze k získávání a odesílání dat do PC, respektive do mikropočítače NVIDIA Jetson. K tomuto účelu je použita knihovna *RPC* pro MicroPython vyvinutá týmem OpenMV. OpenMV mikrokontroler je provozován v režimu *slave* a pomocí třídy *master* na PC jsou na něm volány funkce a následně získávána data.

FLIR Lapton modul

Na mikrokontroleru OpenMV je nainstalován modul s termografickým snímačem FLIR Lepton 3.5. Tento snímač pracuje na vlnových délkách 8–14 mikrometrů (LWIR) a snímá



Obrázek 25 – FLIR Leptobn modul [47]

s maximálním rozlišením 160x120 pixelů. Senzor disponuje automatickou závěrkou a kalibrací

pomocí FFC (flat-field correction). Výrobce udává teplotní citlivost do 50 mK. Celou specifikaci je možné dohledat v [40].

3.6 Použitý software

Pro účely této práce je použit vysokoúrovňový programovací jazyk Python, a to ve verzi 3.8. Vývoj zpočátku probíhal na PC a operačním systému Windows 10, avšak následně byl software implementován na nízkonákladový hardware Nvidia Jetson Nano s procesorem architektury ARM a operačním systémem Linux distribuce Ubuntu 18.04.5.

Snímací systém

V rámci sady Microsoft Azure Kinect Development Kit existuje knihovna Sensor SDK [39] pro tvorbu vlastních aplikací v jazyce *C++* a *C#.* Mezi důležité funkce této knihovny patří především synchronizace streamů RGB a D a jejich vzájemná registrace. Pro využití funkcí z knihovny Sensor SDK v jazyce Python bude v této práci použita knihovna *PyKinectAzure* [41], která zajišťuje port funkcí z *C++* do jazyka Python. Mikrokontroler OpenMV operuje pomocí software MicroPython, který je napsaný v jazyce *C* a pro získání odesílání dat z mikrokontroleru OpenMV je využita knihovna *RPC (Remote Python Call)* [42]. OpenMV mikrokontroler je provozován v režimu *slave* a pomocí funkcí třídy *master,* která je volána na přijímacím zařízení (PC / Jetson Nano) v jazyce Python, jsou volány funkce softwaru MicroPython a následně přijímána data.

Použité knihovny

Knihovnou, jejíž funkce pro práci s obrazem jsou v této práci hojně využívány, je *OpenCV* [43]. Instalace knihovny *OpenCV* byla provedena s podporou CUDA architektury pro akceleraci aplikací strojového učení. Dále jsou použity standartní open-source knihovny jako například *Numpy, Scipy, Pillow, Matplotlib* a další. Celý soubor použitých knihoven je uveden v přiloženém souboru *requirements.txt.* Pro tvorbu jednoduchého grafického rozhraní (GUI) je použita knihovna *Tkinter.*

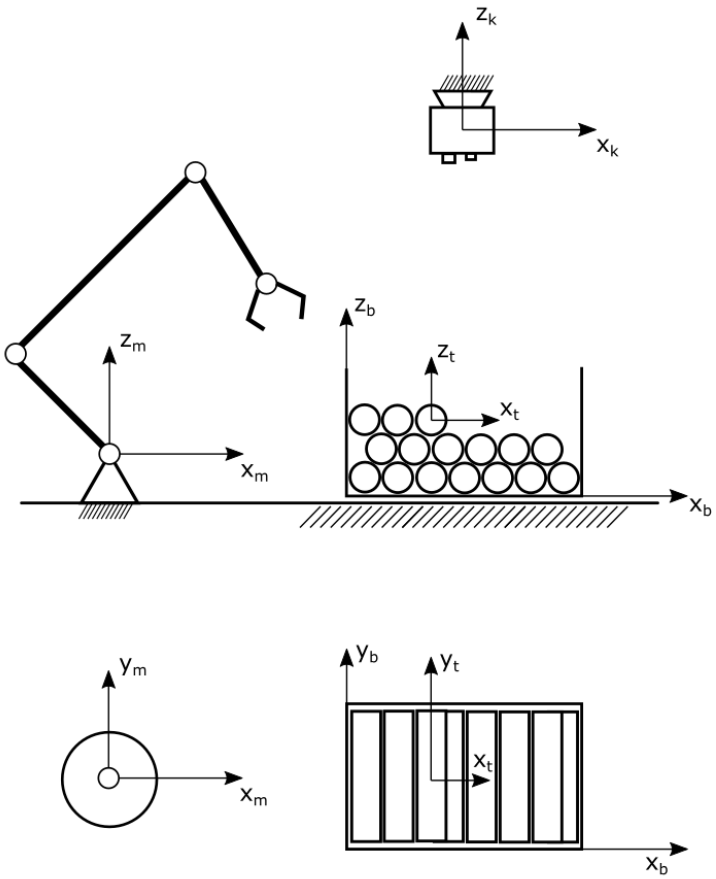
3.7 Shrnutí teoretické části a vybrané metody

Pro praktickou část této práce budou použity následující metody a postupy vyplývající z provedené rešerše. Jako schéma fúze senzorů je vybrána kooperativní fúze, tedy kombinace různých senzorů snímajících jednu scénu pro rozšíření spektra získávaných informací, přičemž zvolenými senzory jsou barevná kamera, hloubkový senzor a termokamera. Z důvodu odlišného zorného pole a fyzické orientace těchto snímačů je nejprve nutné provést registraci obrazů. K tomu je po kalibraci kamer přistoupeno metodou lineární transformace pomocí homografie. Zadanou úlohu bin picking lze zařadit podle úloh strojového vidění do kategorie detekce objektů. Jako první přístup k jejímu řešení je zvolena korelační analýza obrazu, na kterém jsou detekovány hrany. Druhým přístupem je zvoleno supervizorované učení hlubokých konvolučních neuronových sítí. Pro toto řešení je vybrán detektor objektů architektury YOLO v4.

4 Praktická část

4.1 Schéma experimentu

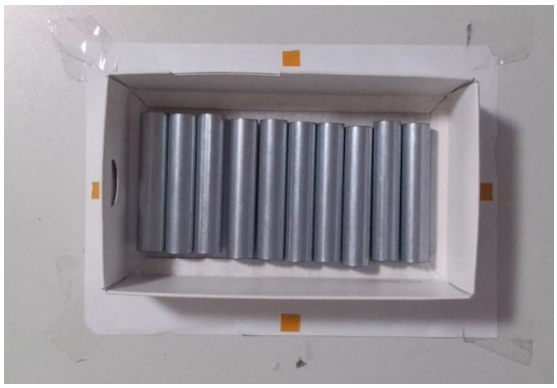
Cílem reálné aplikace je určit polohu polotovarů ve formě trubek, které se nacházejí uvnitř gitterboxu. Pro řešení je zvoleno schéma dle Obrázku 28, kde snímací systém je umístěn nad gitterboxem a je natočen kolmo k podlaze. Výsledný záběr kamer je pak půdorysem scény, který je na Obrázku 2 dole znázorněn.



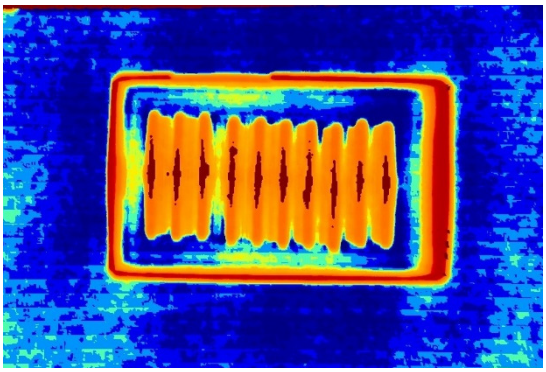
Obrázek 28 – Schéma úlohy bin picking

Předpokládáme, že poloha manipulátoru je vůči kameře s hloubkovým senzorem neměnná a známá. Naopak poloha gitterboxu se může měnit a je třeba ji zjišťovat.

Pro účely této práce bude místo reálných polotovarů použita náhrada ve zmenšeném měřítku o průměru 20 mm. Je zde kladen důraz na fakt, že při řešení této úlohy je klíčovým faktorem odrazivost povrchu polotovarů, a povrch náhrady je tedy volen s podobnými vlastnostmi.



Obrázek 26 – scéna z RGB kamery

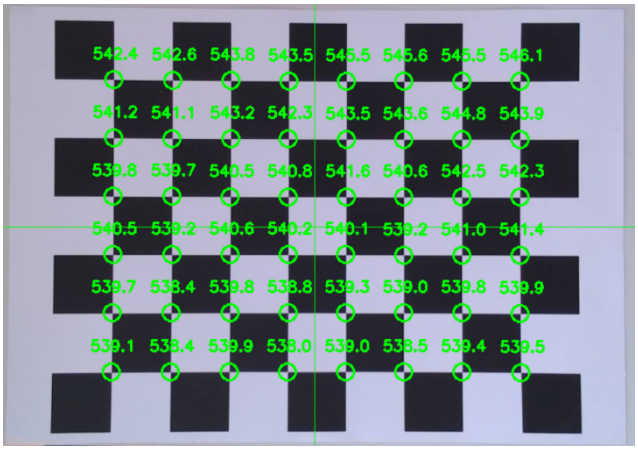


Obrázek 27 – scéna z hloubkového senzoru

Obrázek 28 – scéna experimentu

4.2 Vyrovnání kamery

Pro korektnost celého experimentu a jeho výsledků je nutné zajistit kolmost kamery vůči snímané rovině. Na následujícím obrázku je zobrazen výstup vytvořené funkce, která nalezne body šachovnice a znázorňuje jejich vzdálenosti získané hloubkovým senzorem. Šachovnice leží v rovině, kterou považujeme za vodorovnou.



Obrázek 29 – funkce pro manuální vyrovnání kamery

Pomocí manuálního natáčení kamery je možné dosáhnout přibližného vyrovnání, je však téměř nemožné kameru vyrovnat dokonale. Pro tento účel byla vytvořena funkce, která proloží nasnímanými body rovinu a následně koriguje hloubková data pomocí této roviny. Rovina je v prostoru jednoznačně definována pomocí trojice bodů dle

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \tag{63}$$

$$\boldsymbol{A\,p} \;=\; \boldsymbol{z} \tag{64}$$

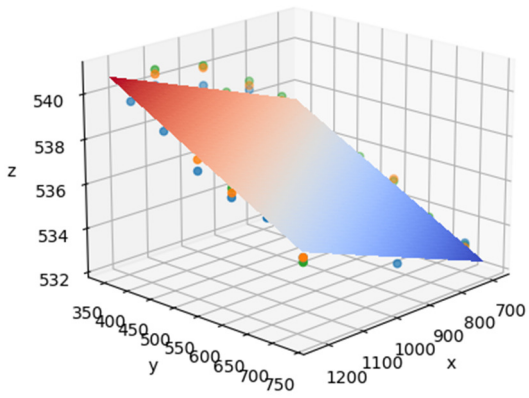
$$\boldsymbol{p} \;=\; \boldsymbol{A^{-1}z} \tag{65}$$

V našem případě, je rovina je počítána z průměru 3 snímků, které obsahují 48 bodů šachovnice, celkově tedy 144 bodů. Matice A tak není čtvercová a nelze provést jednoduchou inverzi. Pro získání řešení ve smyslu nejmenších čtverců, lze použít řešení pomocí pseudoinverzní matice

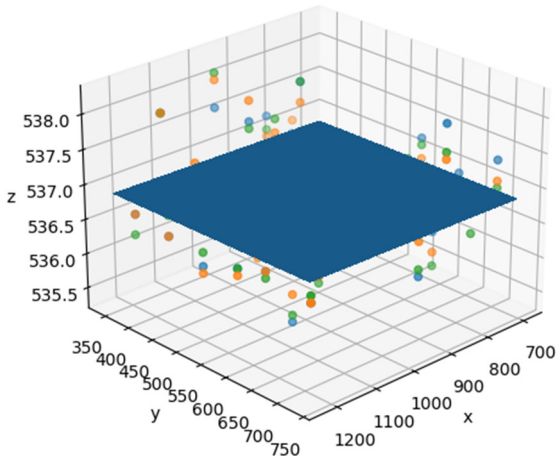
$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ : & : & : \\ x_{144} & y_{144} & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ : \\ z_{144} \end{bmatrix} \tag{66}$$

$$\boldsymbol{A\,p} \;=\; \boldsymbol{z} \tag{67}$$

$$\boldsymbol{p} \;=\; \boldsymbol{(A^T A)^{-1} A^T\, z}\,. \tag{68}$$



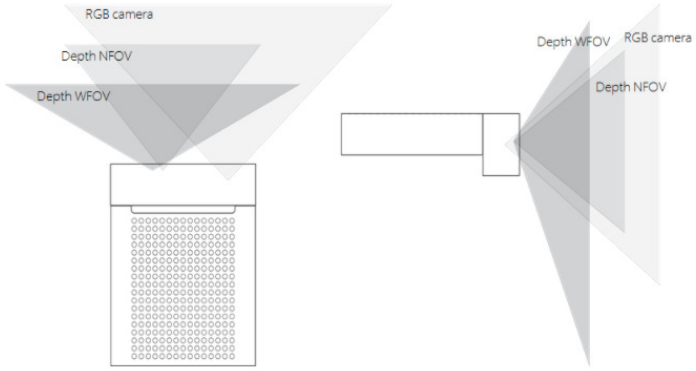
Obrázek 30 – Body šachovnice proložené rovinou ve smyslu nejmenších čtverců



Obrázek 31 - – Body šachovnice proložené korigovanou rovinou

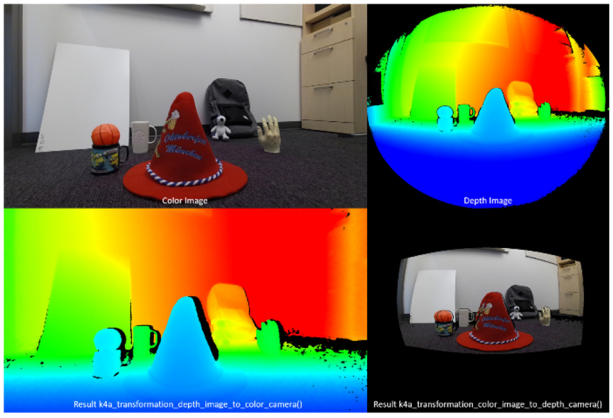
4.3 Registrace RGB-D

Na Obrázku 15 z dokumentace [39] můžeme vidět, že RGB kamera a hloubkový senzor mají rozdílné zorné pole, pozici snímače a jeho natočení.

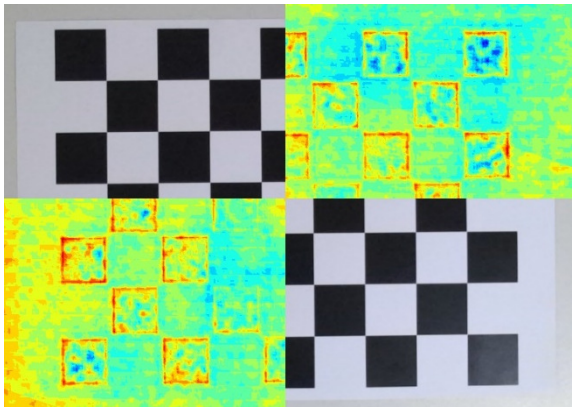


Obrázek 32 - Zorné pole snímače Microsoft Azure Kinect

Aby bylo možno určit vzdálenost pro každý pixel obrazu kamery, tedy souřadnici z v našem experimentu, je zapotřebí provést registraci obrazů RGB (barevná kamera) a D (hloubkový senzor). Registrace obrazů znamená dosažení tzv. pixel-to-pixel korespondence. Tuto úlohu lze řešit po kalibraci jednotlivých snímačů pomocí označení odpovídajících si bodů obrázku a následného určení matice homografie. V tomto případě je však možné využít funkce ze sady Microsoft Azure Kinect Sensor SDK, která tyto procesy činí s využitím znalostí o fyzické konstrukci snímačů.



Obrázek 33 – horní řádek: Původní obrazy z RGB kamery a hloubkového senzoru, spodní řádek: obrazy po registraci

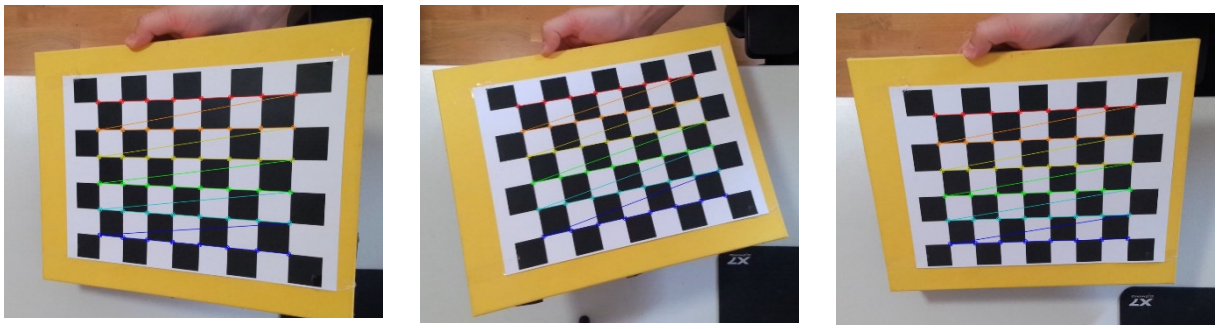


Obrázek 34 - registrovaný obraz RGB-D

Pro účely této práce je považován obraz RGB kamery jako výchozí a obraz hloubkového senzoru je transformován pomocí funkce *k4a_transformation_depth_image_to_color_camera* (Obrázek 36 vlevo dole). Přesnost této transformace je možné zhodnotit jednoduchým překrytím obrazů RGB a D na Obrázku 37. Protože černá pole mají jinou odrazivost, způsobují zkreslení měřené hloubky a je zde po ekvalizaci histogramu šachovnice viditelná. Je evidentní, že registrace obrazů RGB-D není dokonalá, avšak pro účely této práce je považována za vyhovující.

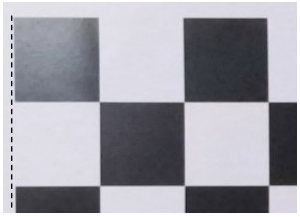
4.4 Kalibrace RGB kamery a určení měřítka

Kalibrace RGB kamery je provedena pro eliminaci mírného zkreslení, které však není zásadní pro registraci obrazů RGB-D. To je rozdíl oproti registraci RGB-D-T v následující kapitole, kde je zkreslení výrazné a je pro korektní registraci nutné nejprve kalibraci provést. Pro kalibraci kamery je použita funkce z knihovny OpenCV. Je vytvořeno interaktivní GUI, pomocí kterého uživatel pořídí alespoň 3 snímky šachovnicového vzoru, který musí být v jedné rovině, při různém natočení této roviny.

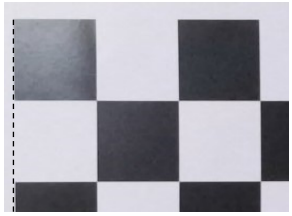


Obrázek 35 – Vstupní data do kalibrační funkce s nalezenými body šachovnice

Výsledek kalibrace není na první pohled zřejmý, avšak zkreslení kamery je nezanedbatelné a je vyčísleno v Tabulce 1 v následujícím kroku při určování měřítka.



Obrázek 36 – snímek před kalibrací



Obrázek 37 – snímek po kalibraci

Dále je určeno měřítko tak, aby bylo možné v pracovní rovině určit vzdálenosti objektů v jednotkách vzdálenosti (mm). K tomu je využito funkcionality kalibrační funkce, která nalezne rohy šachovnice a znalosti skutečné rozteče polí, která činí 40 mm. Vzdálenost poloh jednotlivých bodů na obrázku v pixelech je zaznamenána pro směry x a y a výsledný průměr je vydělen skutečnou vzdáleností. Následně je pro každou hranu zpětně vypočtena odchylka.

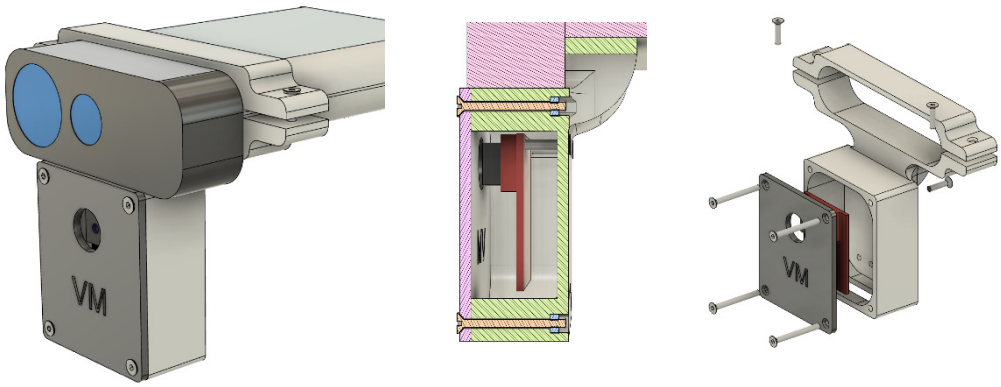
Před kalibrací		Po kalibraci	
$\overline{d_x} = 40.082\,mm$	$\sigma_x = 0.478\,mm$	$\overline{d_x} = 40.006\,mm$	$\sigma_x = 0.130\,mm$
$\overline{d_y} = 39.914\,mm$	$\sigma_x = 0.294\,mm$	$\overline{d_y} = 39.994\,mm$	$\sigma_x = 0.082\,mm$
$\bar{d} = 40.000\,mm$	$\boldsymbol{\sigma_d = 0.408\,mm}$	$\bar{d} = 40.000\,mm$	$\boldsymbol{\sigma_d = 0.109\,mm}$

Tabulka 1 – srovnání odchylek délek hran před a po kalibraci
(d – délka hrany, r – vypočtené měřítko)

4.5 Registrace RGB-D-T

V této kapitole je zadaná úloha využívající fúzi dat z RGB kamery a hloubkového senzoru dále rozšířena o fúzi s daty získanými termografickým měřicím systémem (T). Teplotní data mohou sloužit k určení teploty hledaných objektů, ale také zefektivnit samotnou detekci, a to v případě, že se teplota hledaných objektů liší od teploty okolního prostředí. Při řešení konkrétní úlohy bin pickingu zadané v této práci bylo dosaženo uspokojivých výsledků za pomocí fúze dat RGB a D. Tato kapitola je věnována fúzi dat z termokamery se stávajícím RGB-D systémem pro získání teploty hledaných objektů

Aby bylo možné registrovat obraz termokamery, je nejprve nutné zajistit pevnou vzájemnou polohu snímacích systémů. K tomuto účelu byl navržen adaptér, který slouží zároveň jako kryt mikrokontroleru a snímače OpenMV.

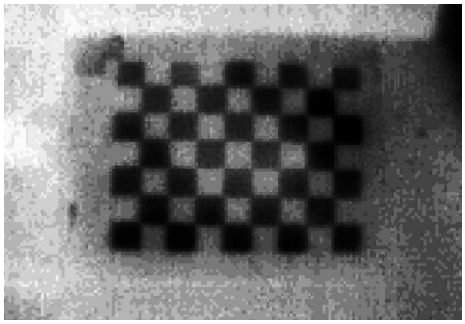


Obrázek 38 – CAD model návrhu adaptéru pro uchycení termokamery

Obrázek 39 - adaptér pro uchycení termokamery vyrobený technologií 3D tisku z materiálu PLA

4.5.1 Kalibrace termokamery

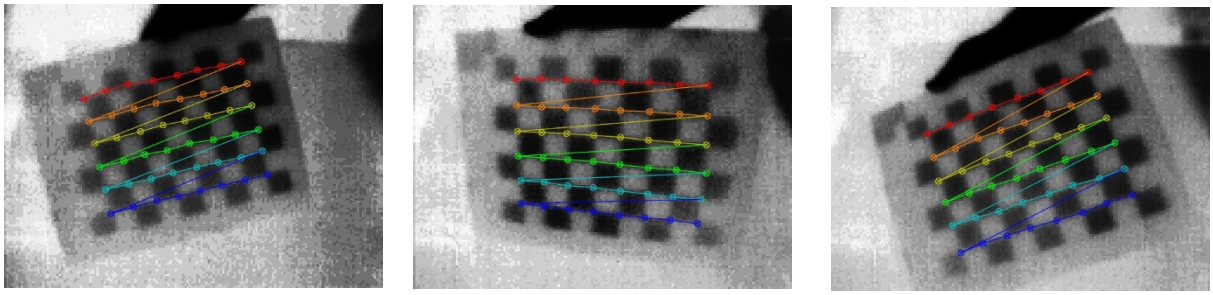
Před samotným určením matice homografie je zapotřebí provést kalibraci zkreslení obrazu termokamery. Postup kalibrace je obdobný jako při kalibraci RGB kamery v Kapitole 7.6. Pro dosažení kontrastu tmavých a světlých polí pomocí termokamery, je šachovnice zahřívána radiačním zdrojem tepla – infralampou. Díky vyšší rychlosti zahřívání tmavých polí šachovnice je dosažen postačující kontrast.



Obrázek 40 – Dosažený kontrast šachovnicového vzoru

Obraz termokamery je dvourozměrná matice hodnot, kterou lze vyobrazit jako černobílý obrázek jako tomu bylo doposud, avšak podobně jako u dat z hloubkového senzoru lze použít funkce pro přiřazení barevné škály stupňům šedi pro vizuálně lepší vyobrazení viz Obrázek 47.

Sada kalibračních bodů je opět získána pomocí polohy rohů polí šachovnice, avšak v tomto případě existující funkce knihovny *OpenCV* k automatickému nalezení těchto bodů nepřináší dobré výsledky. Body na šachovnici je proto nutné určit manuálně za pomocí vytvořeného interaktivního GUI.

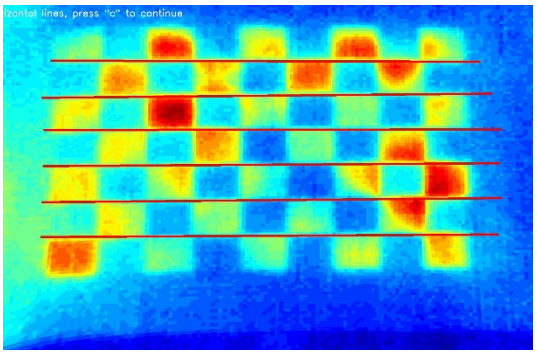


Obrázek 41 – Nalezené body šachovnice

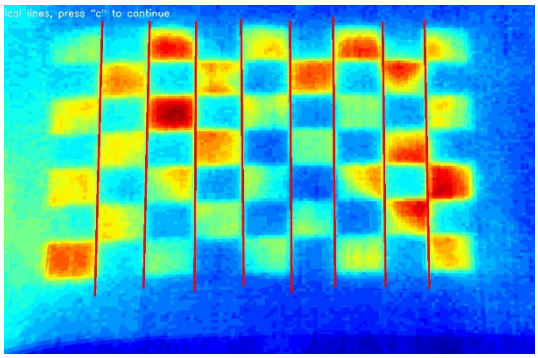
4.5.2 Homografie

Dle Kapitoly 7.5 předpokládáme, že data RGB-D jsou vzájemně registrována a pro dosažení registrace RGB-D-T jsou použita pouze data z kamery RGB. Dosažení přesné registrace obrazů z rozdílných kamerových systému je komplexní úloha a její řešení není zdaleka triviální. Dosažení uspokojivých výsledků bylo však v tomto případě dosaženo lineární projektivní transformací pomocí matice homografie.

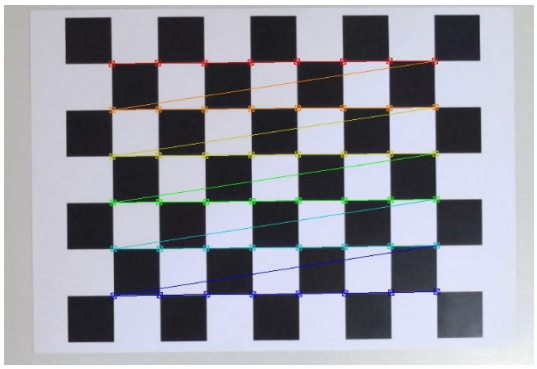
Vstupem do funkce pro získání matice homografie jsou dvě sady korespondujících bodů. Pro získání bodů je opět použit šachovnicový vzor. Obdobně jako u kalibrace, je kontrast dosažen pomocí rozdílné rychlosti zahřívání tmavých polí šachovnice radiačním zdrojem tepla. Protože po kalibraci lze předpokládat minimální zkreslení, je pro urychlení určení jednotlivých bodů implementována funkce pro nalezení průsečíků přímek. Vstup do funkce *findHomography* z knihovny *OpenCV* pak vypadá následovně.



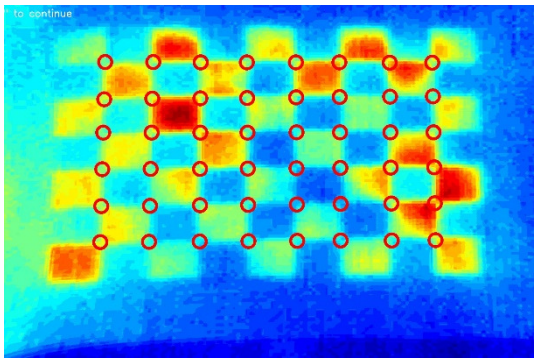
Obrázek 44 – označení vertikálních úsečkek



Obrázek 45 - označení horizontálních úsečkek

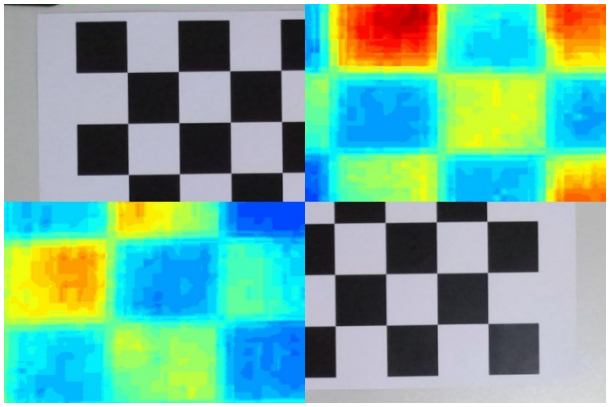


Obrázek 43 – nalezené body šachovnice v datech RGB kamery

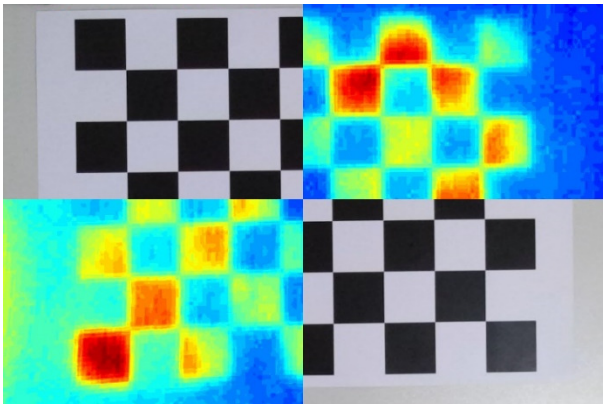


Obrázek 42 – vypočtené souřadnice bodů v datech z termokamery

Získaná matice slouží jako vstup do funkce *warpPerspective,* která obraz termokamery transformuje a je tak dosaženo následujícího výsledku registrace obrazů. Výsledek je opět vyobrazen zkombinováním kvadrantů obou obrazů.



Obrázek 47 – původní zkombinovaný obraz RGB-T



Obrázek 46 – zkombinovaný obraz RGB-T po registraci

4.6 Návrh prototypu pro detekci objektů

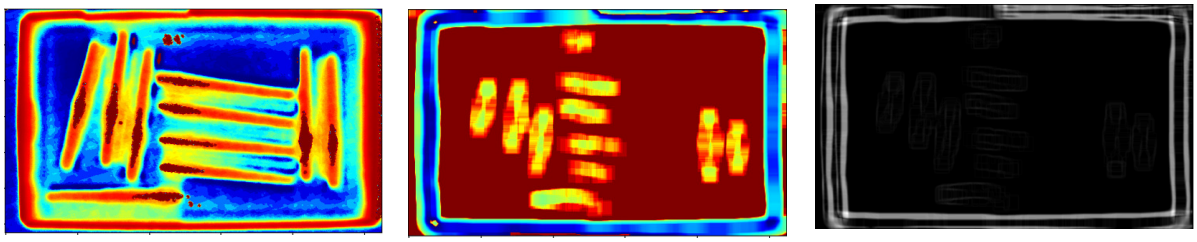
V této kapitole je navržen prototyp softwarového modulu, jehož primárním účelem je vývoj a demonstrace použitelnosti navrhovaných postupů pro řešení daného problému detekce objektů.

4.6.1 Poloha gitterboxu

Před samotným řešením polohy jednotlivých instancí polotovarů se ukázalo jako vhodné nejprve určit polohu a natočení samotného gitterboxu. K určení polohy gitterboxu je také využito korelační analýzy a algoritmu template matching (TM), který je podrobněji popsán v předchozí části práce. Je však zapotřebí určit polohu gitterboxu nezávisle na jeho obsahu, který se může v čase měnit a zásadně tak změnit obraz detekovaných hran. Tento problém je vyřešen použitím dat z hloubkového senzoru a předpokladem, že obsah gitterboxu se vždy nachází ve větší vzdálenosti než jeho okraj. Nezávislost nalezení polohy gitterboxu na jeho obsahu je zdůrazněna tím, že je obsah gitterboxu úmyslně rozmístěn chaoticky. Nejprve je získán vzor jako vstup do samotného algoritmu TM. Je vytvořeno interaktivní GUI, kde pomocí snímku z RGB kamery je ve třech krocích uživatelem zadána poloha gitterboxu.

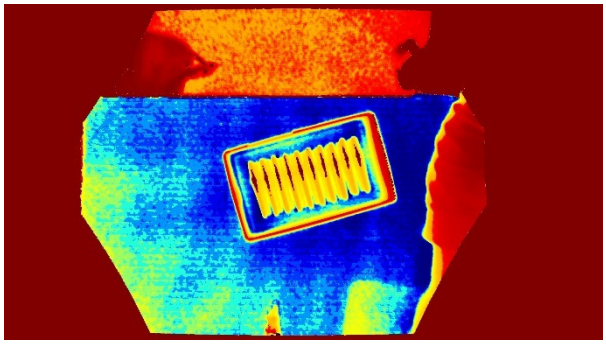
Obrázek 48 – zadání polohy gitterboxu ve třech krocích

Získaná poloha je využita k vytvoření vzoru z dat hloubkového senzoru. Ta jsou nejprve prahována tak, aby byly zachovány pouze pixely s určitou vzdáleností, kterou lze volit pomocí GUI pro dosažení vyhovujících hodnot. Po prahování jsou data filtrována a následně jsou detekovány hrany pomocí konvoluční masky Sobel.

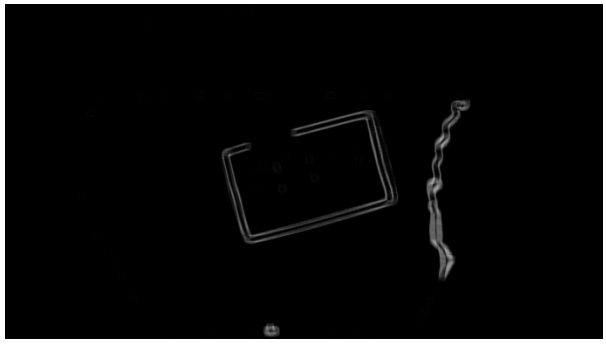


Obrázek 49 – získaní vzoru gitterboxu z dat hloubkového senzoru

Dále je množné přistoupit k samotné detekci polohy gitterboxu. Scéna je zachycena a data jsou upravena obdobným způsobem jako u získání vzoru a je tedy provedeno prahování a filtrace. Detekce hran je opět provedena pouze pro určené spektrum vzdáleností. Vstup do algoritmu TM je vyobrazen na Obrázku 20 na následující straně.



Obrázek 50 – scéna hloubkového senzoru



Obrázek 51 – nalezené hrany ve scéně

Template matching je proveden pro různá natočení scény v rozsahu -90 až 90 stupňů, tedy 180krát. To při rozlišení 1920x1080 znamená značnou výpočetní náročnost a na stolním PC je doba trvání výpočtu přibližně 20 sekund. Scéna i vzor jsou proto před vstupem do algoritmu TM zmenšeny zvoleným faktorem. Uspokojivých výsledků lze dosahovat při zmenšení až na 10 % původního rozlišení a doba výpočtu se tak sníží na 0.17 sekundy. Výsledek vyobrazený níže je pro lepší názornost získán se zmenšením 30 %, kdy doba výpočtu činí 1.32 sekundy.

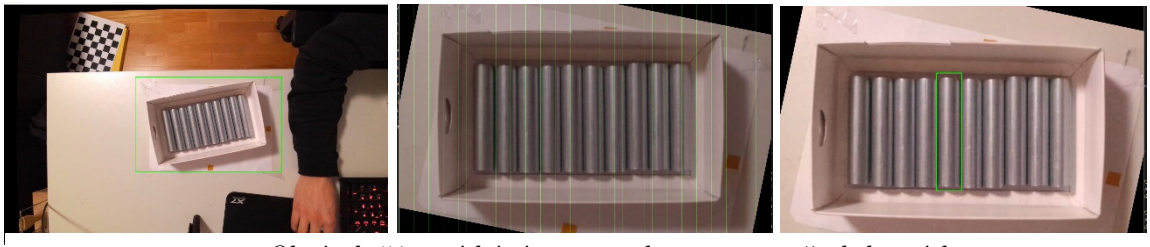
Obrázek 52 – obraz korelačních koeficientů

Obrázek 53 – nalezená poloha gitterboxu

Na Obrázku 55 je vyobrazen výstup algoritmu TM ve formě matice korelačních koeficientů. Každá poloha vzoru odpovídá jednomu elementu matice, a tedy pixelu obrázku. Za hledanou polohu je označeno maximum této matice. Pro následné určování polohy polotovarů je pro zrychlení a zefektivnění jako vstup použit pouze vnitřek gitterboxu viz Obrázek 58.

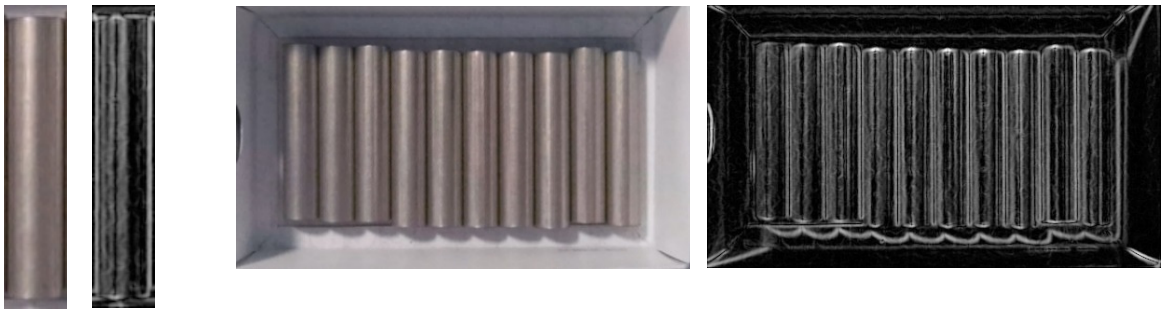
4.6.2 Poloha polotovarů uvnitř gitterboxu

Pro prvotní řešení úlohy lokalizace polotovarů byla zvolena metoda template matching (TM) popsaná v předchozí části práce. Následně je určována poloha jednotlivých instancí polotovarů. Zde je využito algoritmu TM pro snímky z RGB kamery. Předem je určen vzor pomocí obdobného postupu jako při určování vzoru gitterboxu.



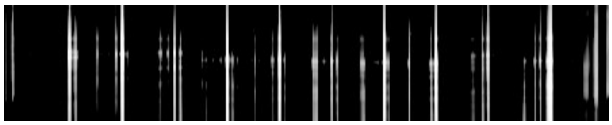
Obrázek 54 – získání vzoru polotovaru ve třech krocích

Scéna je následně automaticky přiblížena a natočena dle získané polohy gitterboxu. Na následujícím obrázku je zobrazen vstup do algoritmu TM pro detekci polotovarů.



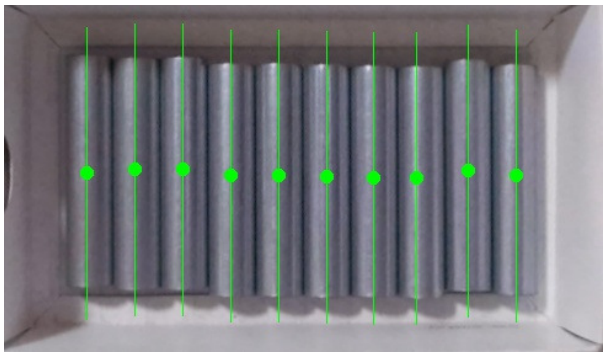
Obrázek 55 – vzor (vlevo) a scéna přiblížená do polohy gitterboxu (vpravo) z kamery RGB se zobrazením výsledků detekce hran

vnořeném cyklu a hledat tak celkovou nejlepší shodu. Vnořený cyklus však zvyšuje výpočetní náročnost a ukazuje se, že díky převládajícímu směru hran ve scéně lze natočení polotovarů uvnitř gitterboxu spolehlivě určit i se vzorem v nesprávném měřítku. Scéna je tedy nejprve opakovaně natáčena, a to v rozsahu -10 až 10 stupňů, protože zde není očekávatelný plný rozsah natočení polotovarů uvnitř gitterboxu. Po nalezení rotace s nejlepší shodou je algoritmus TM proveden opětovně pro různá zvětšení vzoru. Matice korelačních koeficientů je uložena pro nejlepší dosaženou shodu a vyobrazena na následujícím obrázku.

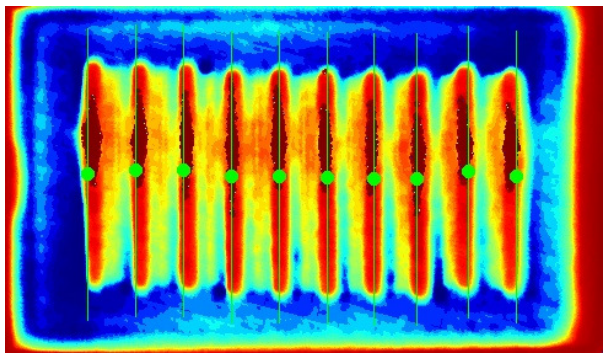


Obrázek 56 - obraz korelačních koeficientů

Z této matice jsou následně rozlišeny jednotlivé instance. Toho je dosaženo nalezením maxima ze součtu sloupců a prahováním ostatních součtů sloupců. Následně je v nalezených sloupcích určeno maximum, které definuje polohu v ose *y*.



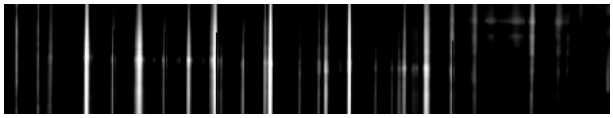
Obrázek 58 – nalezené polohy instancí



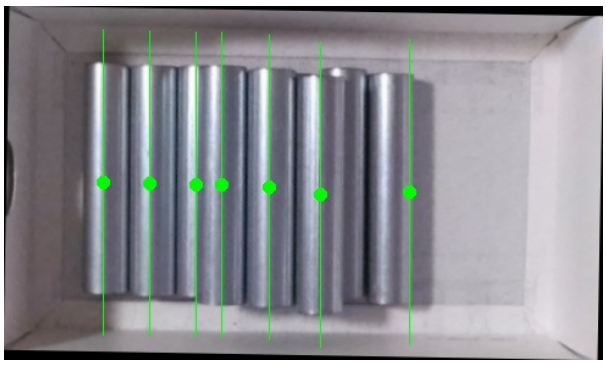
Obrázek 57 – označení instancí v datech D

Polohu v ose **z** lze následně určit pomocí dat D. Tmavé regiony v těchto datech jsou bohužel způsobeny vysokou odrazivostí povrchu polotovarů a vzdálenost zde odečítat nelze. Dalším problémem je evidentní zkreslení obrazu D vůči RGB i přes to, že je provedena transformace pomocí Kinect SDK. Pro další využití dat ke spolehlivému určení polohy v ose *z,* je zapotřebí data D a RGB kamery rektifikovat. Dále byla provedena detekce pro případ, kdy jsou polotovary naskládány ve dvou řadách. Většina objektů je detekována korektně, avšak vyskytuje se zde

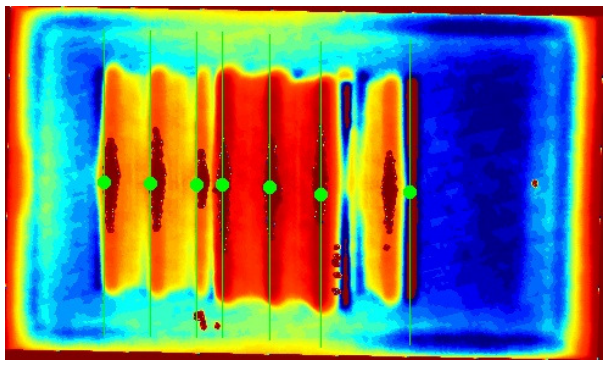
problém s citlivostí na osvětlení. Je vidět, že tímto přístupem vznikají shody se stíny polotovarů. V jiných případech vznikají shody také s odrazy polotovarů v jejich lesklém povrchu.



Obrázek 59 - obraz korelačních koeficientů



Obrázek 60 – příklad nevyhovující detekce



Obrázek 61 - příklad nevyhovující detekce v datech D

Určování polohy polotovarů pomocí korelační analýzy dat z RGB kamery dosahuje v některých případech vyhovujících výsledků (Obrázek 61), avšak pouze při dobrém konstantním osvětlení scény. Tento přístup je velmi citlivý na nastavení filtrů pro získání poloh z korelačního obrazu (Obrázek 62). Po správném nastavení pro konkrétní světelné podmínky se při jejich změně přesnost detekce výrazně snižuje. Z toho důvodu tento přístup nelze považovat za vyhovující řešení. Dosažené výsledky mohou však být s výhodou vyžity v následujícím řešení k urychlení procesu anotace dat pro supervizorované učení detektoru objektů.

4.6.3 Detektor objektů YOLO v4

Jako druhotné řešení úlohy lokalizace polohy polotovarů je zvoleno supervizorované učení hluboké konvoluční neuronové sítě architektury YOLO v4, která je detailněji popsána v předchozí části práce. K implementaci této architektury je využit open-source framework Darknet [44] [45], který byl pro použití tohoto detektoru objektů vytvořen jeho autory. Tento framework napsaný v jazyce C umožňuje snadnou a rychlou implementaci detektoru objektů YOLO. Navíc díky podpoře architektury CUDA umožňuje paralelizovat výpočetně náročné operace pomocí podporovaných GPU.

Prvním krokem je získání a anotace dat pro supervizorované učení. Jak již bylo zmíněno, pro usnadnění a urychlení tohoto procesu lze využít výsledky předchozí detekce. Snímky jsou uloženy společně s textovým souborem s označením poloh ohraničujících rámečků ve formátu, který podporuje framework Darknet. Pro každý snímek je tak vytvořen následující soubor anotací

$$\begin{matrix} class & x_r & y_r & w_r & h_r \\ class & x_r & y_r & w_r & h_r \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{matrix} ,$$

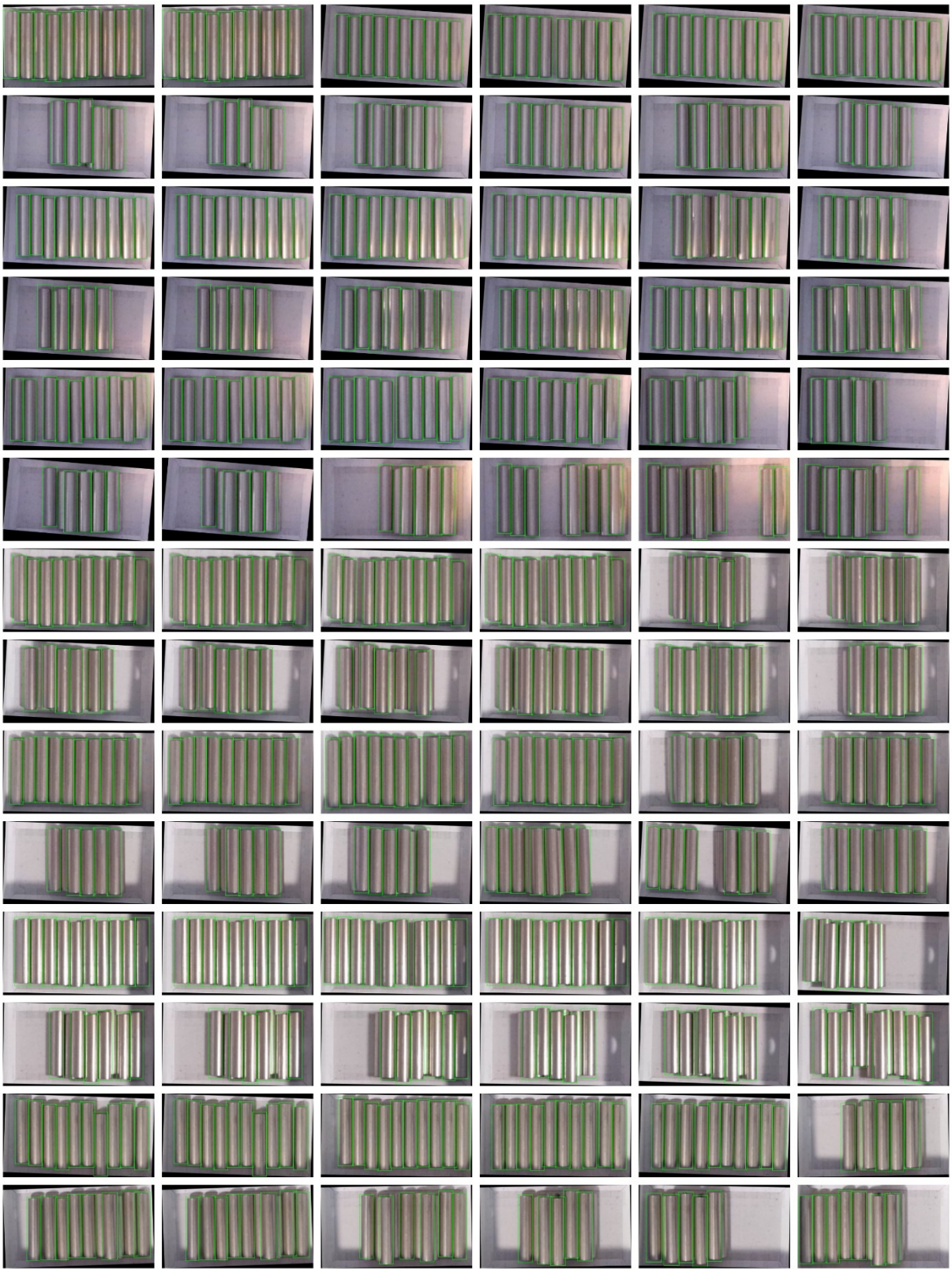
kde x_r a y_r jsou relativní polohy ohraničujícího rámečku v rozsahu 0 až 1, w_r a h_r jsou potom relativní šířka a výška tohoto rámečku také v rozsahu 0 až 1. Označení třídy je v tomto případě irelevantní, protože jsou detekovány objekty pouze jedné třídy. Celkem bylo pořízeno 70 snímků viz Obrázek 66 . Dále byla data augmentována pomocí změny kontrastu a jasu, obrazových filtrů a vertikálního či horizontálního převrácení.

Originální obrázek | *Snížený jas* | *Zvýšený jas*

Šum „Salt and pepper“ | *Gaussovský šum* | *Horizontální převrácení*

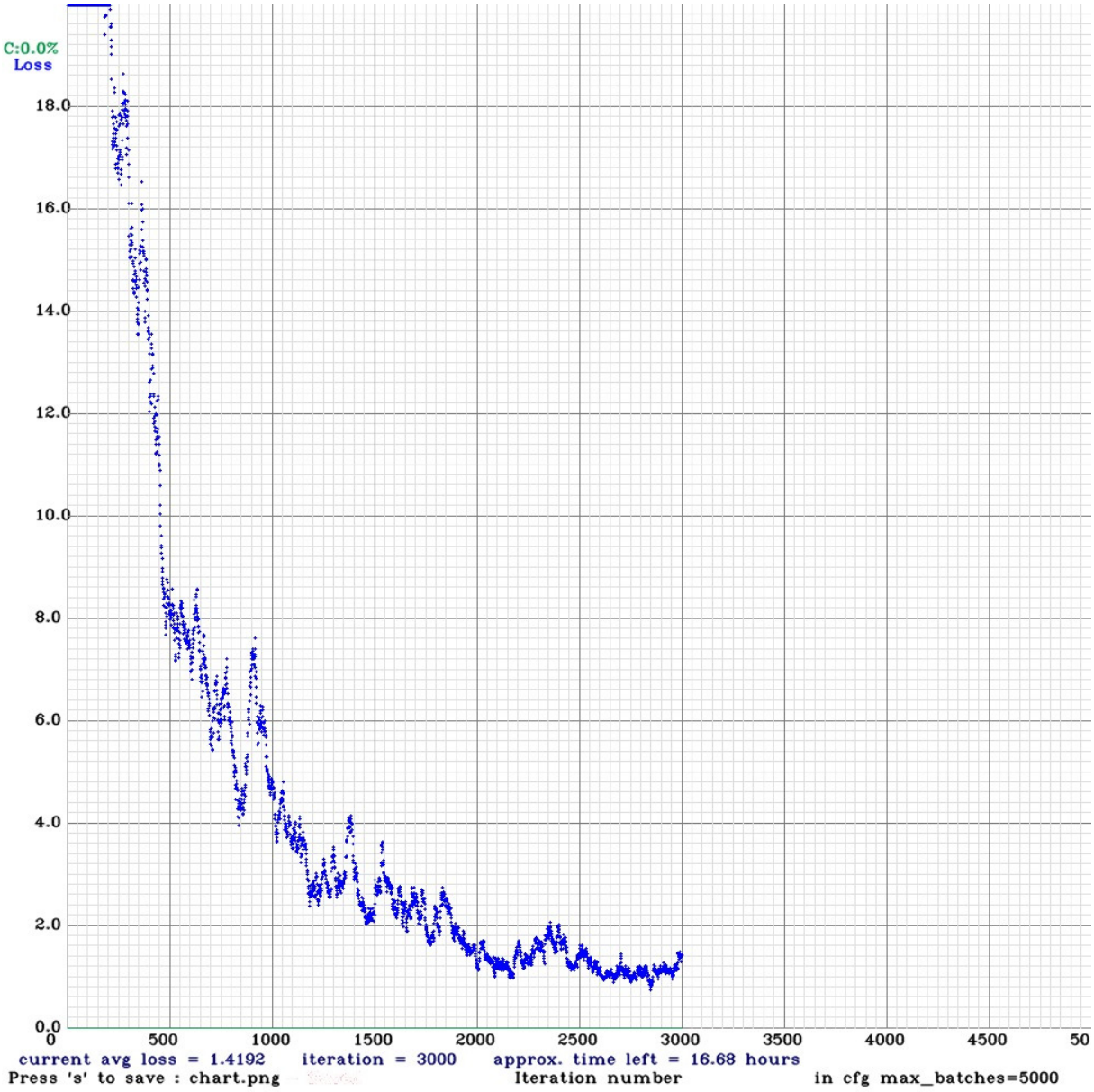
Obrázek 62 – Výběr z použitých metod augmentace obrazových dat

Výhody augmentace jsou diskutovány v předchozí části práce, přičemž je možné použít již získaných anotací pro takto upravené obrázky a jde tak o téměř bezpracné rozšíření datesetu. Celkem byl takto získán dataset o obsahu 1050 obrázků s průměrem 7 instancí na obrázek. Takto získaný dataset byl náhodně rozdělen na trénovací a testovací data poměrem 85/15.

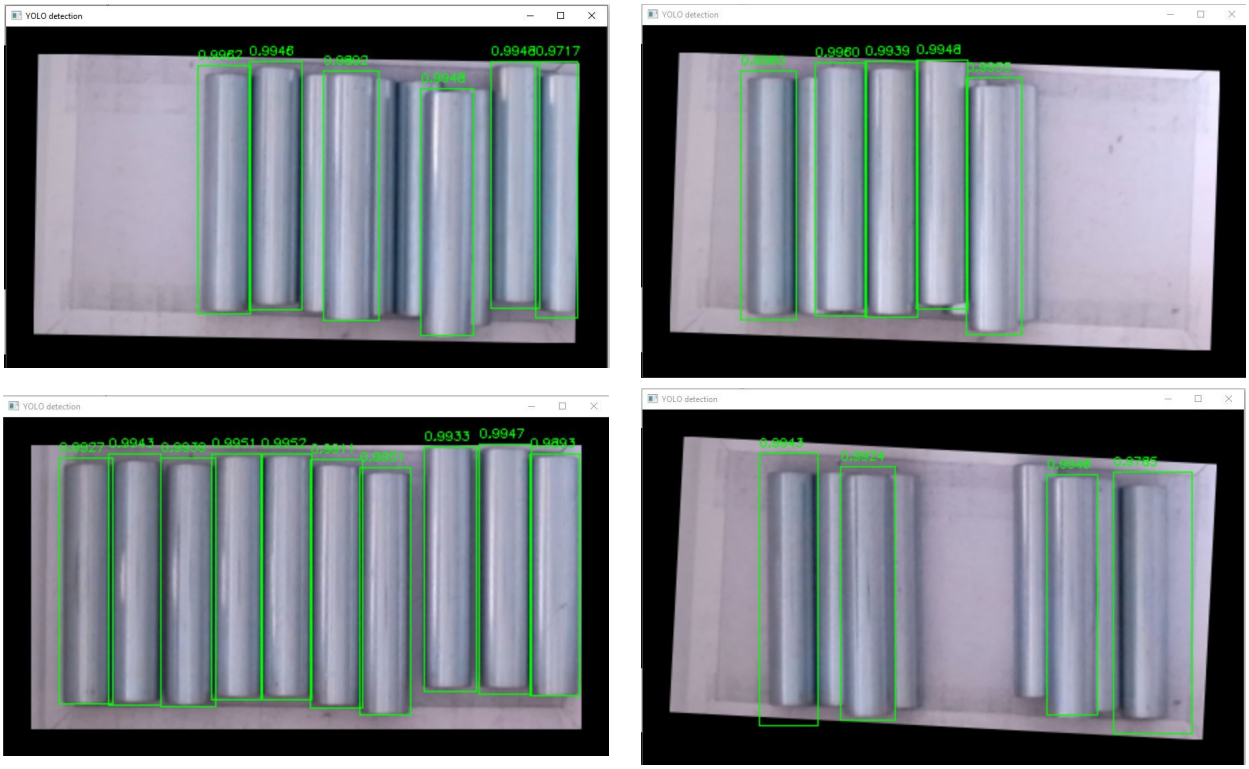


Obrázek 63 –vytvořený dataset sedmdesáti anotovaných obrázků před augmentací pro supervizorované učení detektoru objektů

Po korektním nastavení velikosti vstupních obrazových dat a dalších parametrů detektoru bylo zahájeno učení, které probíhalo na GPU Nvidia GeForce 940MX, a pro dosažení průměrné přesnosti detekce blížící se ke 100 % na testovacím datasetu trvalo 20 hodin a 3000 iterací při velikosti dávky 16. Učení bylo zastaveno při výrazné stagnaci poklesu ztrátové funkce. Výsledná přesnost detekce na testovací části datasetu byla při poslední iteraci učení stanovena na 100.0 % a její reálná hodnota před zaokrouhlením se tak velmi blíží optimu. Následně byl detektor otestován na nově získaných obrazových datech, které nejsou součástí trénovacího datasetu (viz Obrázek 68). Také zde je přesnost detekce velmi dobrá a ve většině případů se pohybuje nad hranicí 99 %.



Obrázek 64 – průběh ztrátové funkce při procesu učení detektoru YOLO v4

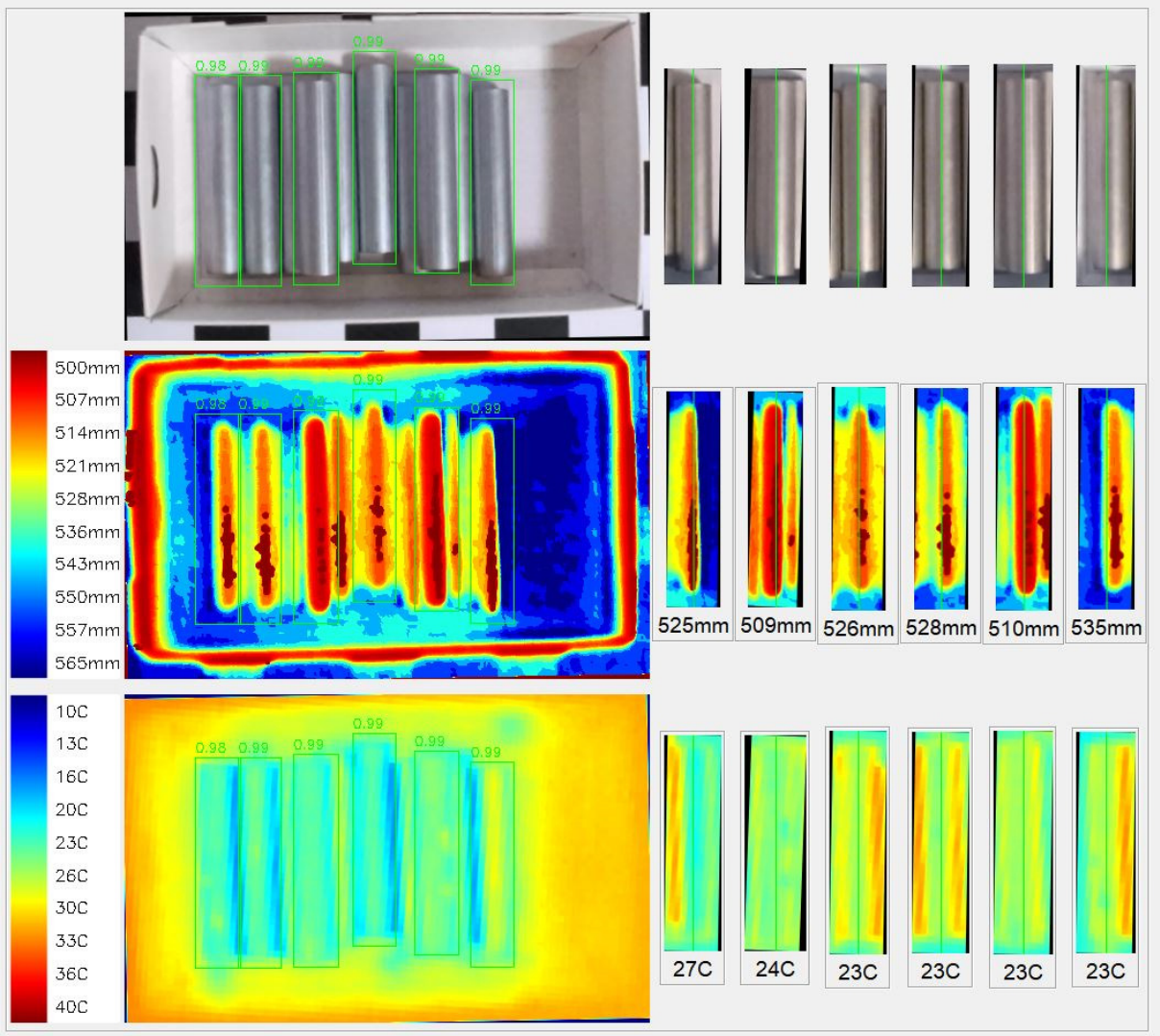


Obrázek 65 – testování detektoru objektů YOLO v4 na nově získaných obrazových datech

Vyhovující přesnosti dosahuje detektor jak v případě jednoduché scény s jednou vrstvou polotovarů (Obrázek 45 vlevo dole), ale také pokud jsou polotovary naskládány ve více vrstvách. Navíc byl detektor objektů pomocí správné anotace trénovacích dat natrénován tak, aby jako platnou detekci označil pouze volně přístupné polotovary, které nejsou zakryty další vrstvou.

4.7 Určení vzdálenosti a teploty polotovarů

Dalším krokem po nalezení poloh jednotlivých polotovarů uvnitř gitterboxu je definovat jejich polohu v prostoru pomocí dat z hloubkového senzoru. Při dosažené kvalitě detekce se jeví jako dostačující označit vertikální osu ohraničujícího rámečku jako osu polotovaru. Následně je vzdálenost osy ve směru z určena jako modus z hodnot pixelů v blízkém okolí osy. Odečítat vzdálenost v jednom bodě či jako aritmetický průměr na ose se ukázalo jako nevyhovující, protože je kvůli odrazivosti povrchu polotovarů hloubkový senzor zatížen šumem, který je příčinou rozsáhlých oblastí nekorektních hodnot.



Obrázek 66 – výsledná detekce polohy polotovarů v prostoru a určení jejich teploty

Následně je možné označit polohu polotovarů také v datech z termokamery a odečíst hodnoty z konkrétních pixelů pro určení teploty. Pro ilustraci byla v tomto případě scéna zahřívána radiačním zdrojem tepla a polotovary se pravděpodobně z důvodů nižší absorpce tepla jeví chladnější.

4.8 Diagram softwarového modulu

Na následujícím schématu je popsán chod softwarového modulu fúze senzorů včetně modulu pro zpracování dat a detekci. Není zde uveden chod SW modulu s vytvořeným GUI pro definici nezbytných vstupních atributů, které bude popsáno v následující kapitole. Při tvorbě tohoto diagramu lze rozvádět jednotlivé komponenty do větších podrobností a je nutné zvolit kompromis mezi názorností a přesností popisu. Jde tedy pouze o velmi hrubý přehled nejdůležitějších datových toků, který slouží k ilustraci principu vytvořeného SW modulu.

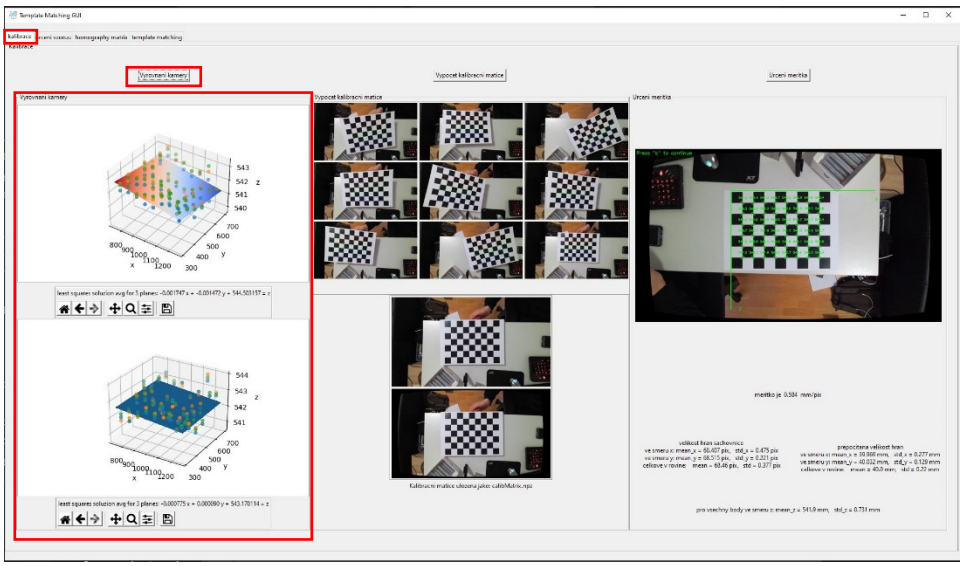
Obrázek 67 – legenda diagramu

Obrázek 68 – diagram SW modulu fúze senzorů

4.9 Popis vytvořeného GUI pro definici vstupních atributů

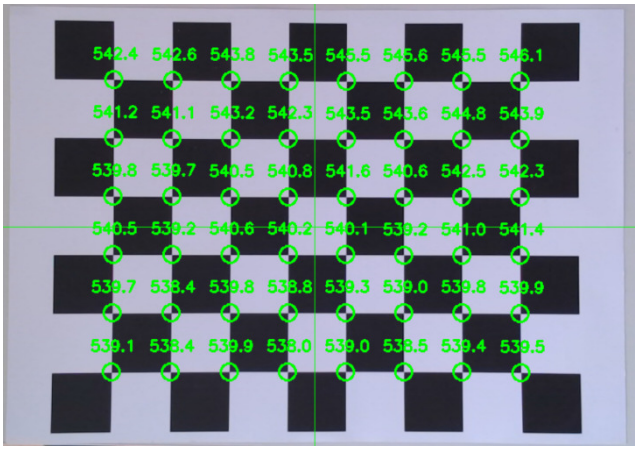
V této kapitole je popsáno a názorně vysvětleno vytvořené GUI, které slouží k definici vstupních atributů nezbytných pro korektní funkčnost SW modulu detekce. Celé grafické rozhraní je rozděleno do záložek podle vykonávaných funkcí.

4.9.1 Vyrovnání kamery



Obrázek 69 – GUI pro vyrovnání kamery

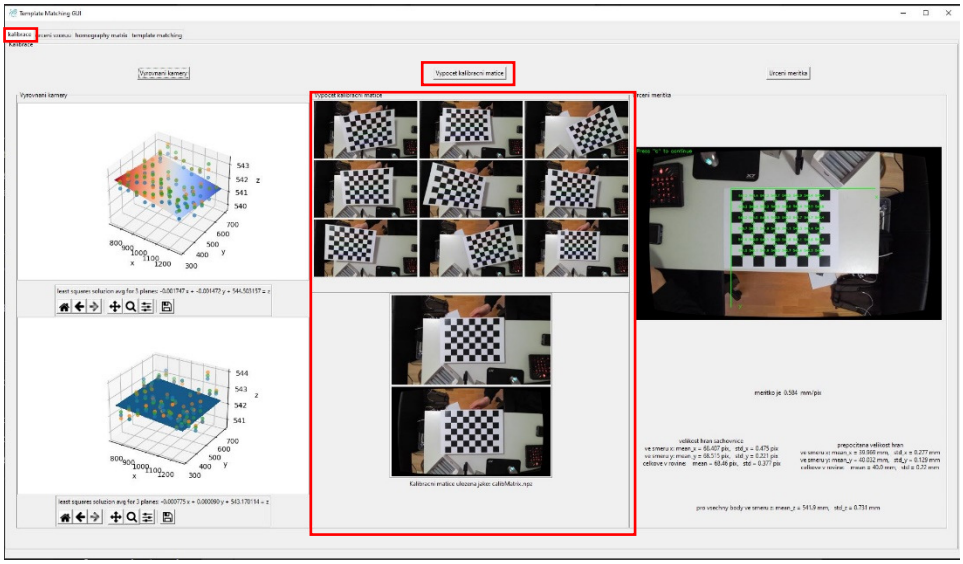
Prvním krokem je vyrovnání snímacího systému tak, aby směřoval kolmo ke snímané rovině. Po stisku tlačítka „vyrovnání kamery“ je zobrazen video stream z RGB kamery a jsou zobrazeny snímané vzdálenosti nalezených bodů šachovnicového vzoru, který je umístěn na podkladu. Uživatel je pomocí natáčení snímacího systému schopen dosáhnout přibližného vyrovnání.



Obrázek 70 – výstup funkce pro manuální vyrovnání kamery

Následně je nasnímanými body proložena rovina, která je zobrazena v 3D grafu. Protože nelze dosáhnout dokonalého vyrovnání je tato rovina stále vychýlena od roviny podkladu. Je proto určena korekční matice, která se odečítá od snímaných hodnot, které jsou po korekci zobrazeny v grafu níže.

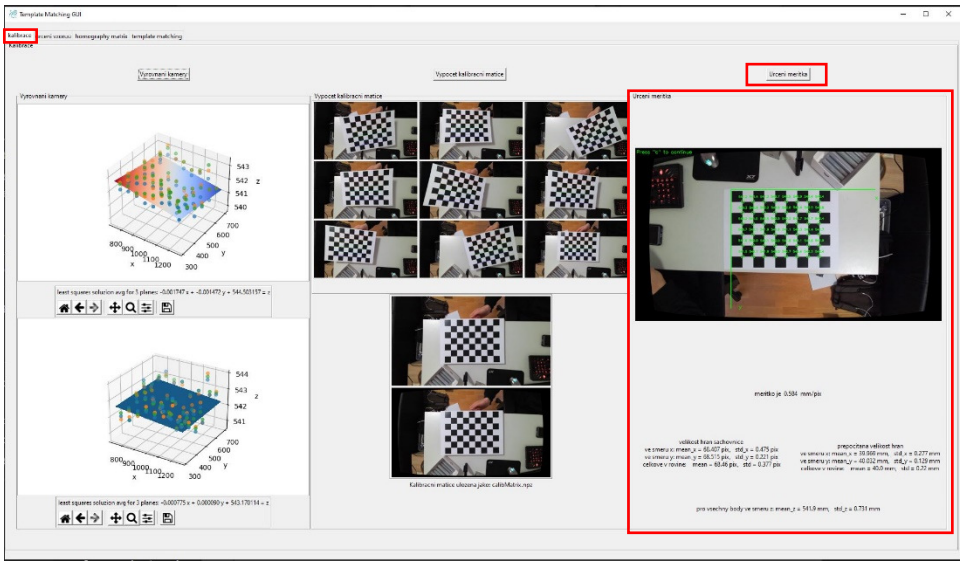
4.9.2 Kalibrace RGB kamery



Obrázek 71 – GUI pro kalibraci RGB kamery

Dalším krokem je určení kalibrační matice pro zmírnění zkreslení snímků. Po stisku tlačítka je uživateli zobrazen video stream z RGB kamery a pomocí stisku klávesy je možné pořídit několik snímků šachovnicového vzoru (nejméně 3), který musí být v jedné rovině v prostoru (na pevné desce). Uložené snímky jsou zobrazeny a následně je vypočtena a uložena kalibrační matice. Nakonec je zobrazen výsledek kalibrace porovnáním snímku před a po kalibraci.

4.9.3 Určení měřítka

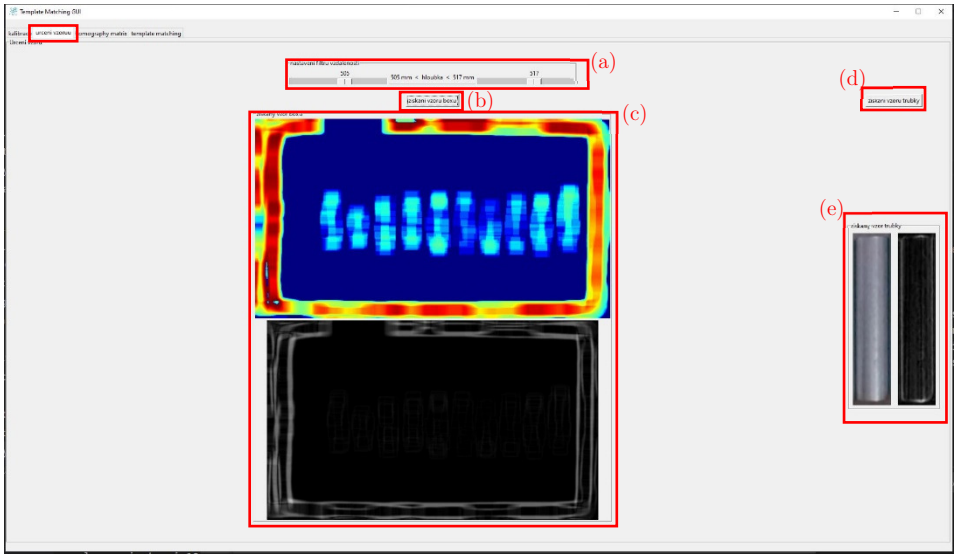


Obrázek 72 – GUI pro určení měřítka

Dále je možné určit měřítko neboli velikost pixelu v milimetrech. K tomu se po stisku tlačítka opět zobrazí stream snímků. Uživatel zarovná šachovnicový vzor se zobrazenými osami a po stisku tlačítka je z detekovaných bodů šachovnice určena průměrná velikost políček v horizontálním a vertikálním směru. Při znalosti jejich skutečné velikosti v milimetrech je pak vypočteno měřítko a jsou vyobrazeny výsledky, které udávají směrodatnou odchylku velikostí

jednotlivých polí. Při předpokladu dokonalého vzoru lze považovat tuto odchylku za způsobenou zkreslením kamery a lze tak ověřit kvalitu provedené kalibrace. Dále je také zobrazena směrodatná odchylka odečtu vzdáleností detekovaných bodů šachovnice ve směru z, které by měly ležet v jedné rovině, a je tak získán přehled o chybě měření hloubkového senzoru.

4.9.4 Určení vzoru gitterboxu



Obrázek 73 – GUI pro určení vzoru gitterboxu

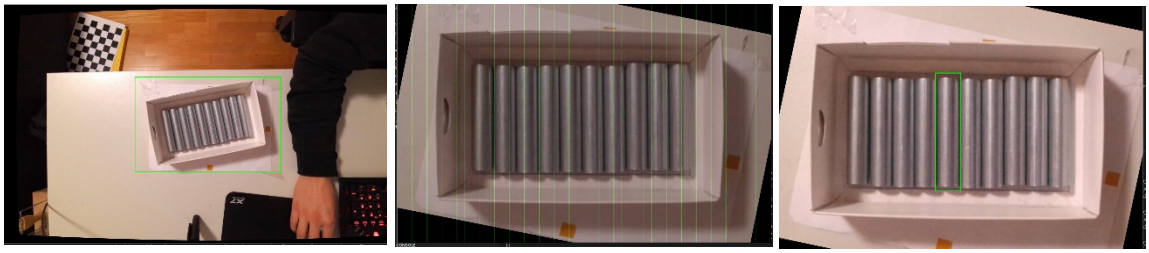
K určení polohy gitterboxu je využito korelační analýzy a algoritmu template matching (TM), který je podrobněji popsán v předchozí části práce, s využitím dat z hloubkového senzoru. K tomu je zapotřebí získat vzor jako vstup do samotného algoritmu TM. Za pomoci interaktivního GUI je po stisku tlačítka (b) zobrazen snímek z RGB kamery a ve třech krocích je uživatelem pomocí myši zadána poloha gitterboxu.

Obrázek 74 – zadání polohy gitterboxu ve třech krocích

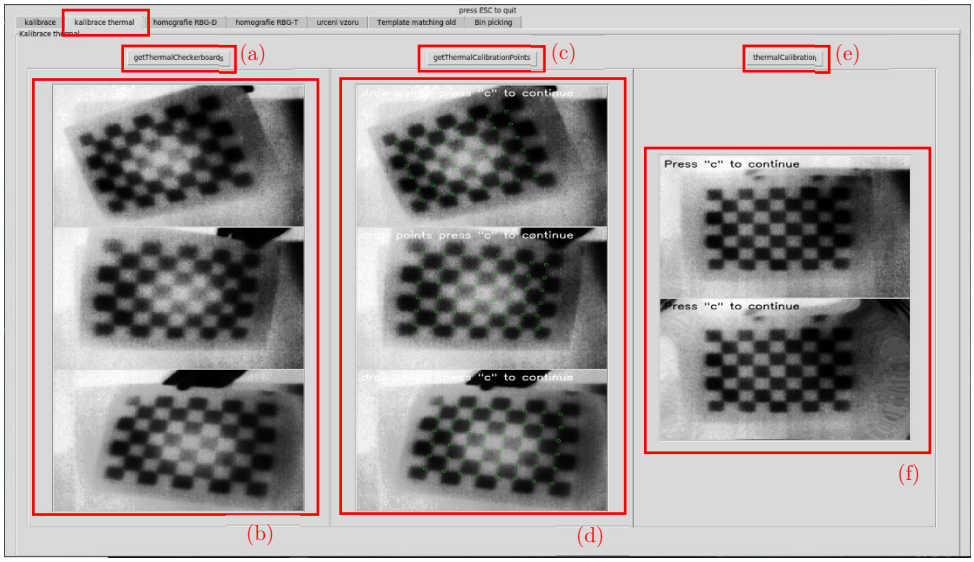
Zadaná oblast je převedena do dat hloubkového senzoru, ta jsou filtrována a je aplikován algoritmus *Sobel edge detector* pro nalezení hran. Výstelky jsou zobrazeny (c) a pomocí sliderů (a) je možné měnit rozsah filtrů vzdálenosti. Efekt této změny aktualizuje zobrazované výsledky, aby bylo možné dosáhnout vyhovujícího výsledku, kdy je odfiltrován obsah gitterboxu a zobrazuje se pouze jeho okraj.

4.9.5 Určení vzoru polotovaru

Obdobně jako v předchozím případě je po stisku tlačítka (d) možné zadat polohu polotovaru, jehož vzor je v tomto případě vytvořen ze samotného snímku z RGB kamery. Na tomto snímku jsou detekovány hrany a výsledky jsou zobrazeny (e).

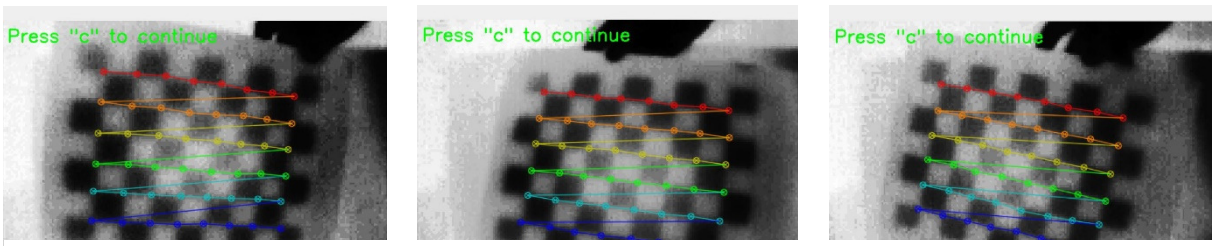


Obrázek 75 - získání vzoru polotovaru ve třech krocích



4.9.6 Kalibrace termokamery

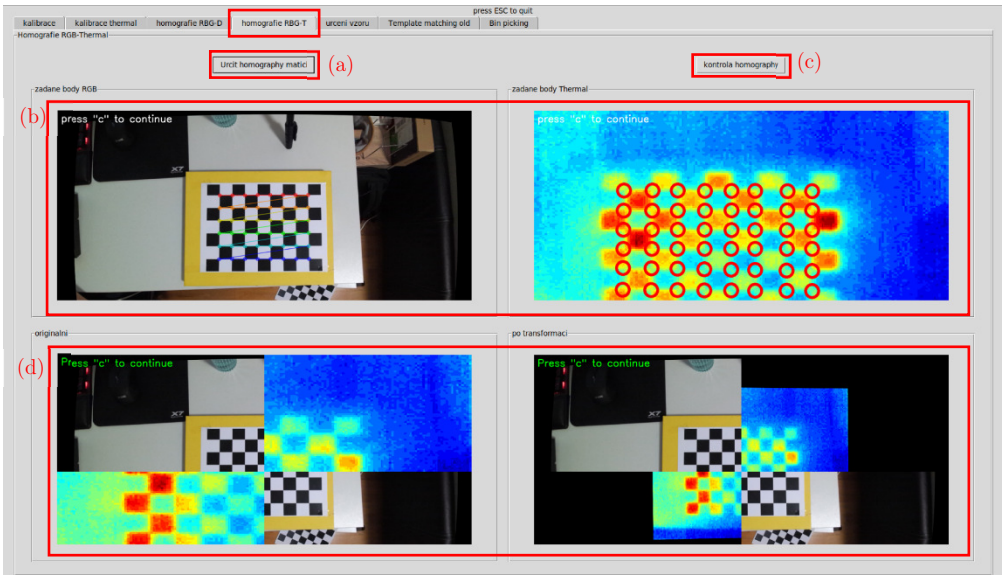
Na další záložce je možné určit kalibrační matici termokamery, pomocí které lze následně termokameru kalibrovat, což je nezbytné pro registraci obrazů RGB – T. Po stisku tlačítka (a) je zobrazen stream snímků z termokamery. Po dosažení potřebného kontrastu polí šachovnice například pomocí radiačního zdroje tepla je možné stiskem klávesy uložit 3 snímky šachovnice v různém natočení v prostoru. Opět je však nezbytné, aby šachovnice samotná byla v jedné rovině. Pořízené snímky jsou vyobrazeny (b) a po stisku tlačítka (c) na nich je možné postupně pomocí myši definovat všechny body pro kalibraci ve správném pořadí viz Obrázek 80 – manuálně zadané body šachovnic jako vstup do kalibrační funkce.



Obrázek 77 – manuáůně zadané body šachovnic jako vstup do kalibrační funkce

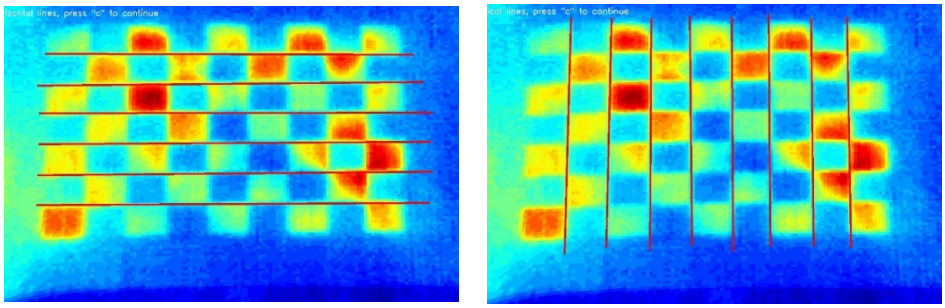
Následné je vypočtena kalibrační matice a její korektnost je možné otestovat pomocí stisku tlačítka (e) a vyobrazených výsledků kalibrace (f).

4.9.7 Homografie RGB-T



Obrázek 78 – GUI pro homografii RGB - T

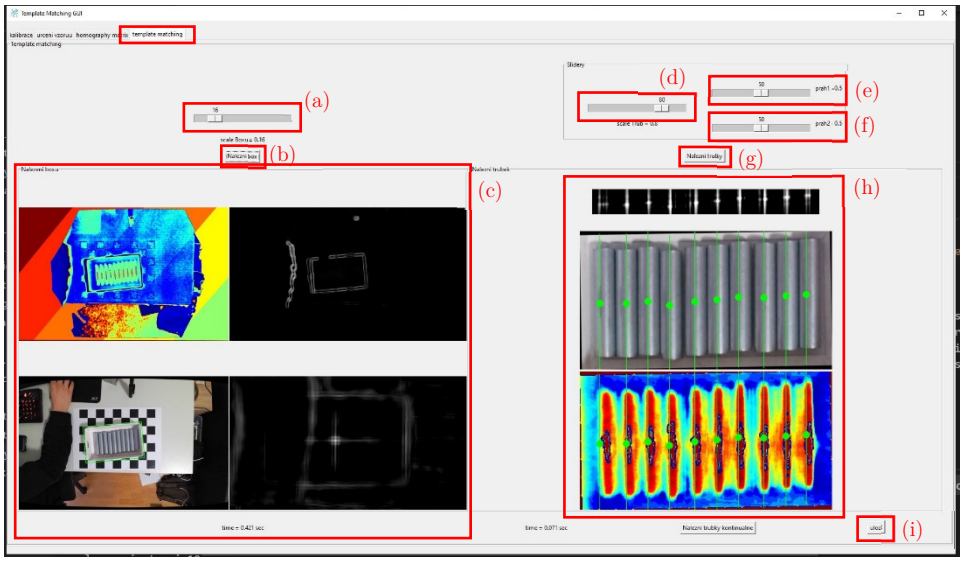
Dále je možné registrovat obrazy z RGB kamery a obraz z termokamery, pro získání pixel-to-pixel korespondence. Při stisku tlačítka (a) se zobrazí detekované body šachovnice na snímku z RGB kamery, a následně je možné obdobné body označit na snímku termokamery. Protože v tomto kroku předpokládáme již kalibrovaný a nezkreslený obraz termokamery, je pro usnadnění implementován algoritmus pro výpočet příslušných bodů pomocí průsečíků zadaných přímek. Nejprve je nutné definovat horizontální přímky a následně vertikální.



Obrázek 79 – definice bodů šachovnice pomocí přímek

Získané vstupní body jsou vyobrazeny (b) a matice homografie je vypočtena. Následně po stisku tlačítka (c) je možné zobrazit porovnání původních a registrovaných obrazů (d).

4.9.8 Detekce polohy pomocí algoritmu TM



Obrázek 80 – GUI pro detekci pomocí algoritmu TM

Na této záložce je možné provést detekci poloh polotovarů pomocí korelační analýzy – algoritmu TM. Při detekci se postupuje tak, že nejprve je nalezena poloha gitterboxu a následně je možné určit polohy jednotlivých polotovarů.

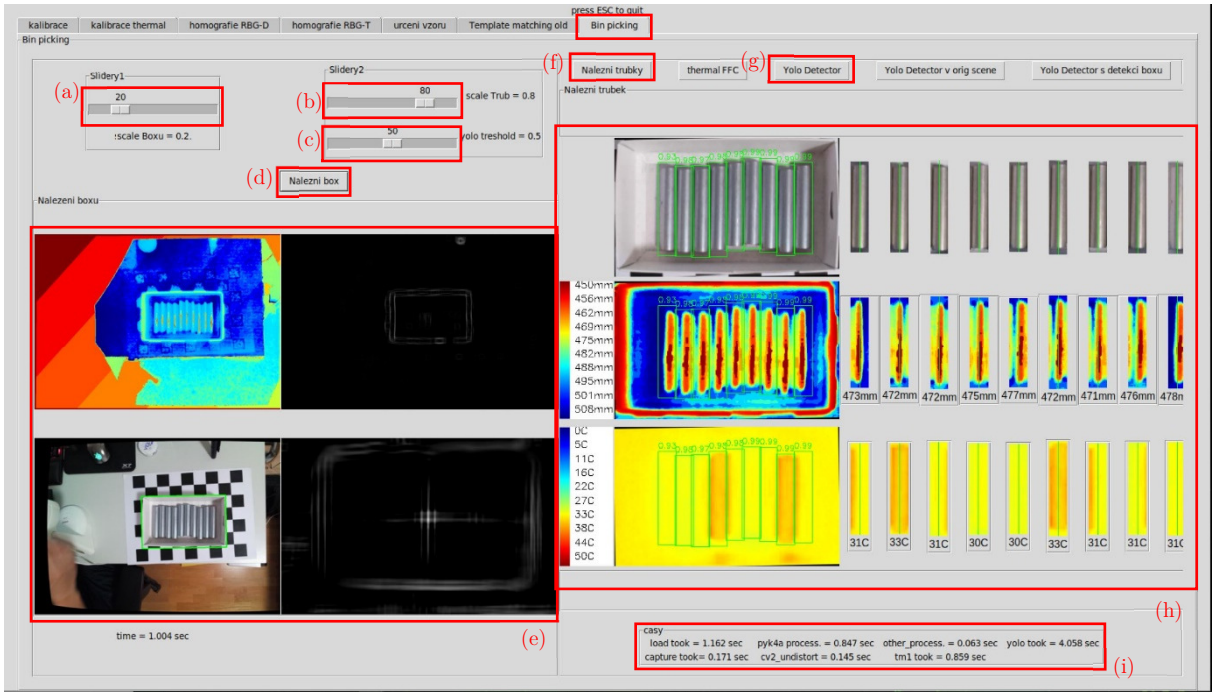
Detekce polohy gitterboxu

Pomocí slideru (a) lze nastavit zmenšení vstupních obrazových dat pro zrychlení algortimu TM, jehož přesnost se tím však snižuje. Experimentálně bylo zjištěno, že pro detekci polohy gitterboxu lze korektních výsledků dosahovat při zmenšení až na 15 % původní velikosti. Detekce polohy gitterboxu je spuštěna tlačítkem (b) a následně jsou vyobrazeny její výsledky (c) a nalezená poloha je uložena.

Detekce polotovarů

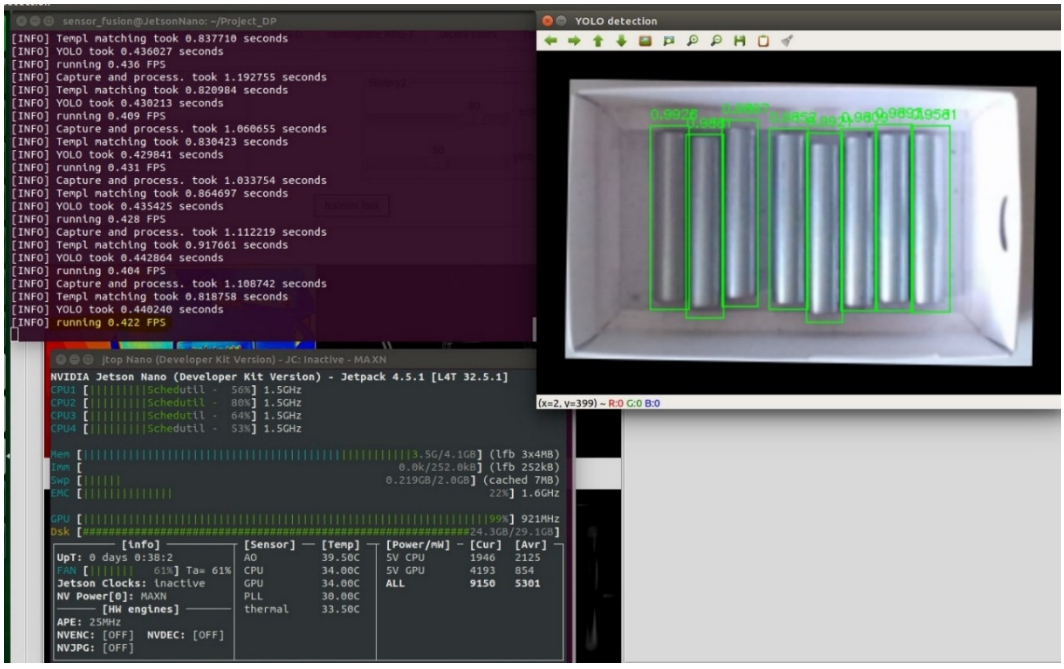
Obdobně jako v předchozím případě lze pomocí slideru (d) nastavit zmenšení vstupních obrazových dat pro zrychlení algortimu TM, avšak v tomto případě jsou výsledky tímto zmenšením velmi ovlivněny a ideální nastavení je přibližně 80 %. Dále jsou zde slidery pro úpravu extrakce všech instancí z obrazu korelačních koeficientů. Zatímco slider (e) upravuje mezní korelaci pro jednotlivé sloupce (polohu ve směru x), tak slider (f) upravuje tento práh pro konkrétní body (polohu ve směru y). Po stisku tlačítka (g) je provedena detekce a výsledky jsou vyobrazeny (h). Nastavení prahů (e)(f) se ovlivňuje navzájem a najít optimální nastavení trvá několik iterací. Tato citlivost na nastavení také nakonec vede k upuštění od řešení detekce tímto způsobem. Výsledky provedené detekce lze pomocí tlačítka (i) uložit pro tvorbu datasetu pro následné supervizorované učení detektoru objektů YOLO v4.

4.10 Finální detekce



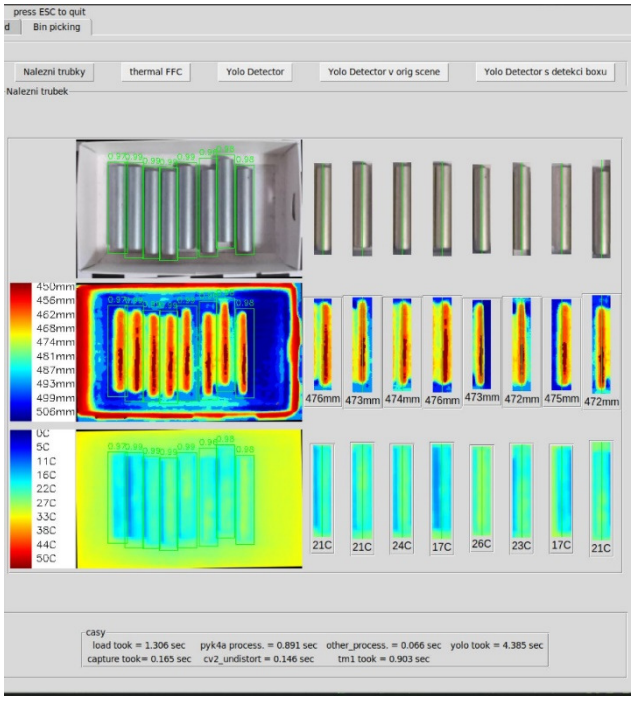
Obrázek 81 – GUI pro finální detekci a fúzi senzorů

Na této záložce je implementován finální algoritmus detekce podle diagramu na Obrázku 71 s využitím detektoru objektů YOLO v4. Obdobně jako v předchozím případě je nejprve zapotřebí určit polohu gitterboxu. K tomu je implementován identický algoritmus, slider (a), tlačítko (a) a zobrazení výsledků (e). Obdobně je zde i slider (b) pro zmenšení velikosti vstupu do algoritmu TM pro detekci polotovarů, protože je zde tento algoritmus také využit, avšak pouze pro nalezení úhlu natočení polotovarů. Poslední slider (c) slouží k nastavení parametru *threshold* detektoru objektů YOLO v4. Po stisku tlačítka (f) je provedena detekce a jsou vyobrazeny výsledky (h) a také časy trvání jednotlivých operací (i). Detektor je také možné pomocí tlačítka (g) spustit v kontinuálním režimu viz Obrázek 16.

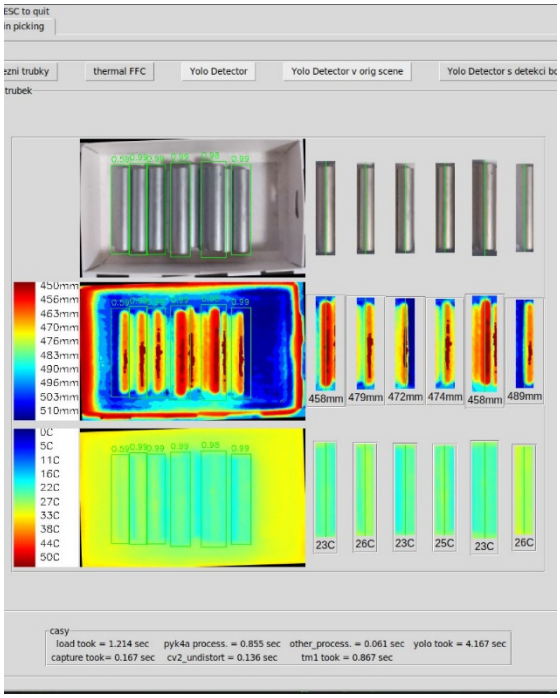


Obrázek 82 – kontinuální detekce

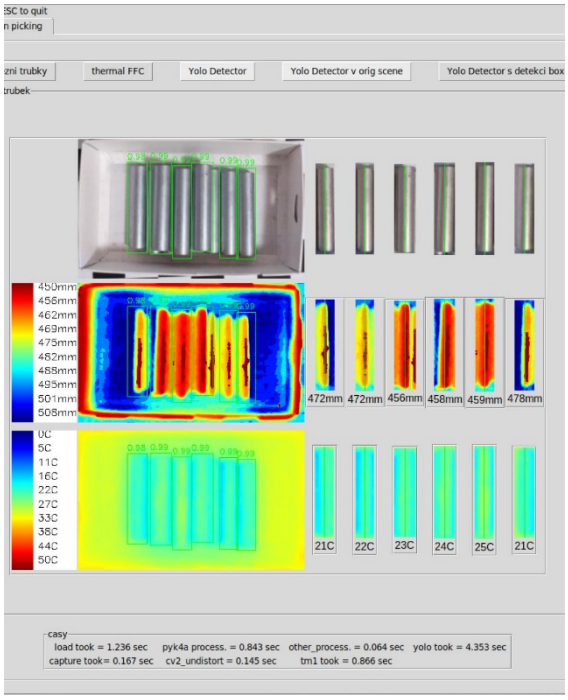
4.11 Testování



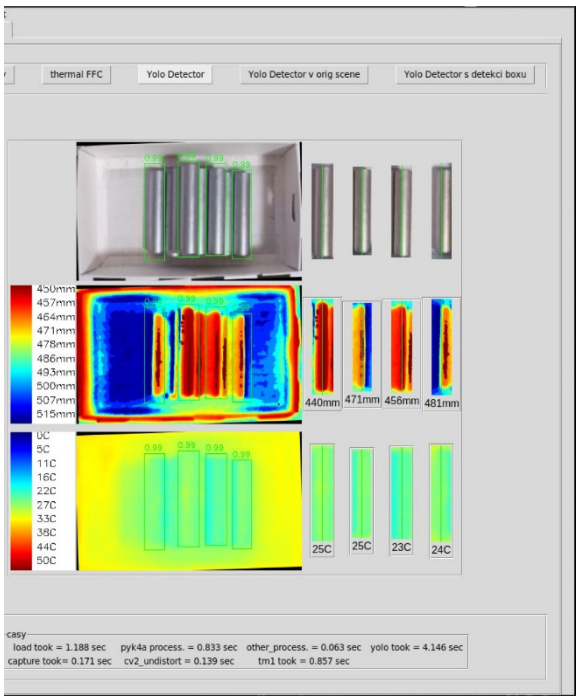
Obrázek 83 – polotovary v jedné vrstvě, podchlazené



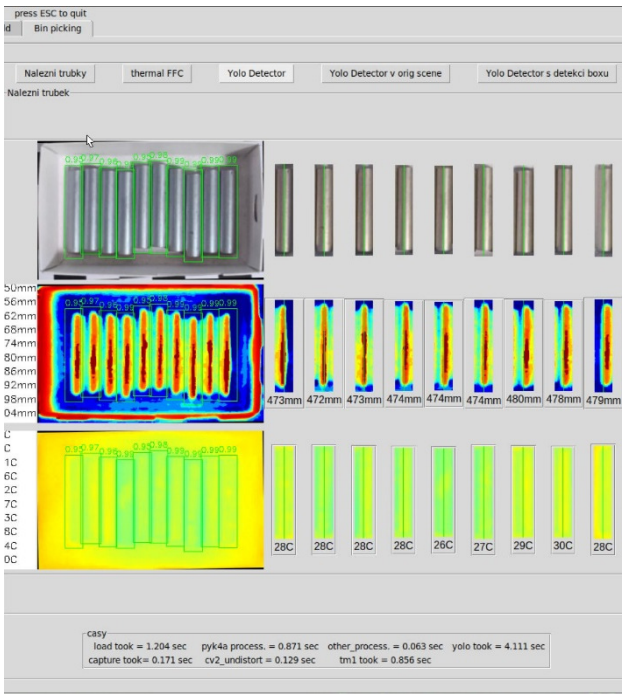
Obrázek 84 – polotovary ve dvou vrstvách, podchlazené



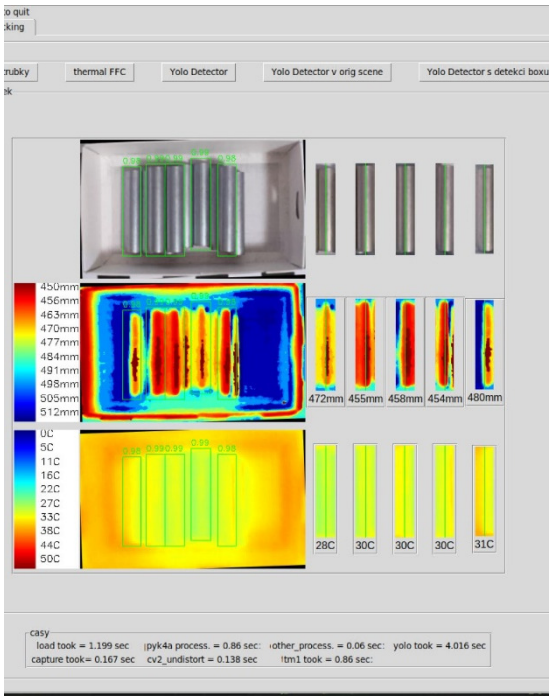
Obrázek 85 - polotovary ve dvou vrstvách, podchlazené



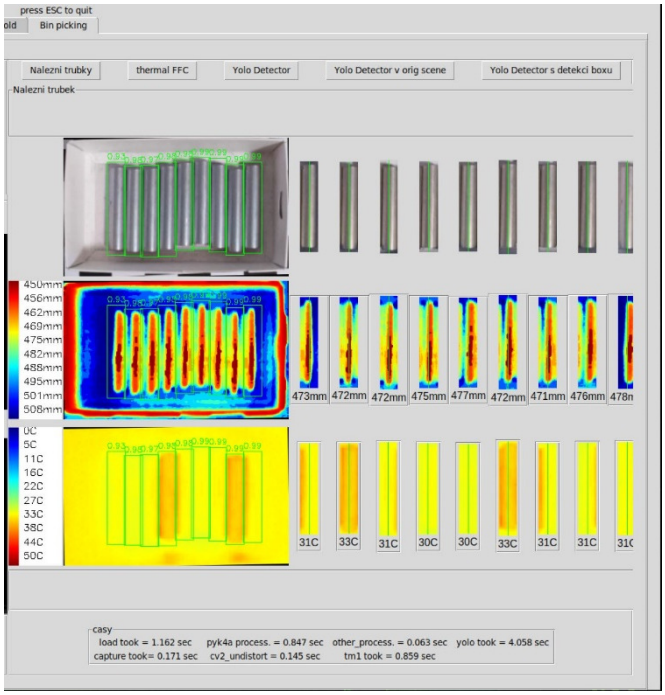
Obrázek 86 - polotovary ve třech vrstvách, podchlazené



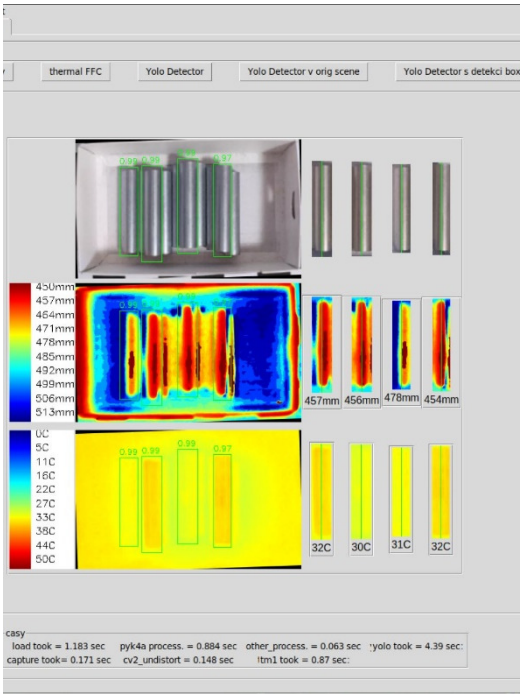
Obrázek 88 - polotovary v jedné vrstvě, scéna zahřívaná radiačním zdrojem tepla



Obrázek 87 - polotovary ve dvou vrstvách, scéna zahřívaná radiačním zdrojem tepla



Obrázek 90 - polotovary v jedné vrstvě, vybrané dva polotovary zahřáty



Obrázek 89 - polotovary ve dvou vrstvách, vybrané dva polotovary zahřáty

5 Závěr

V této práci byla nejprve provedena rešerše metod fúze senzorů a pro zadanou úlohu bylo vybráno schéma kooperativní fúze, tedy kombinace různých senzorů snímajících stejnou scénu pro rozšíření získávaných informací. Pro tuto práci byla vybrána kombinace senzoru viditelného spektra – RGB kamery, hloubkového senzoru a snímače infračerveného spektra – termokamery. Dále byla provedena rešerše metod strojového vidění a zadaná úloha byla definována jako detekce objektů, kde cílem je definice jak třídy, tak polohy hledaných objektů. Pro detekci objektů byly formulovány a popsány dva různé přístupy řešení. Jako první přístup je zvolena korelační analýza jinak nazývaná Template matching, kde je využito vzoru hledaného objektu a zkoumá se míra shody pro různé polohy a orientace. Druhým přístupem, který zažívá v současné době obrovský rozvoj, je pak strojové učení hlubokých konvolučních neuronových sítí. V této práci byly popsány základní principy supervizorovaného strojového učení a dále shrnuty a vysvětleny jednotlivé vrstvy konvolučních neuronových sítí. Dále byla provedena rešerše existujících architektur, ze kterých byla vybrána jedna z nejpopulárnějších architektur detektoru objektů současnosti YOLO v4. Tato architektura byla blíže vysvětlena a bylo popsáno její složení. Poslední bod teoretické části se zabývá tématem registrace obrazů, jejímž cílem je transformace obrazových dat různých kamerových systémů k dosažení shodné pozice odpovídajících si bodů neboli tzv. pixel-to-pixel korespondence.

V praktické části této práce byl vytvořen experiment, který ve zmenšeném měřítku simuluje zadanou úlohu. Je zvoleno schéma detekce, kdy snímací systém je umístěn nad snímanou scénou tak, že výsledný obraz RGB kamery je půdorysem a lze tak definovat polohu v souřadnicích x a y, zatímco data hloubkového senzoru představují třetí souřadnici z. Snímací systém se skládá ze snímače Microsoft Azure Kinect, který disponuje RGB kameru a hloubkovým senzorem. Dále pro získání teplotních dat je použit termografický snímač FLIR Lapton na vývojové desce OpenMV H7. Jako výpočetní HW je zvolen z rozměrových a rozpočtových důvodu Nividia Jetson Nano. Při implementaci SW je řešení je kladen důraz na open-source a použit je tak vysokoúrovňový programovací jazyk Python ve verzi 3.8, který disponuje hojným množstvím přidružených knihoven s licencí umožňující volné šíření. V této práci byla mimo jiné využívána zejména knihovna OpenCV. Pro spouštění a interakci s implementovanými SW moduly je vytvořeno jednoduché GUI pomocí knihovny Tkinter.

Nejprve je implementováno řešení úlohy registrace obrazů pomocí výpočtu homografie, které umožňuje pořizovat synchronizované snímky RGB-D-T. Pro řešení samotné úlohy detekce objektů je nejprve implementováno řešení využívající korelační analýzy. Problém je rozdělen na dvě části, kdy je nejprve nalezena poloha a natočení gitterboxu a následně je uvnitř gitterboxu určována poloha jednotlivých polotovarů. Uživatel pomocí interaktivního GUI definuje vzor polotovaru a vzor gitterboxu. K nalezení polohy gitterboxu se ukázalo vhodné využít filtrovaná data hloubkového senzoru. Tato metoda vede ke spolehlivému nalezení jeho polohy a natočení i v případech, kdy se liší jeho obsah. Korelační analýza byla následně použita pro nalezení polohy samotných polotovarů. Pro vysokou odrazivost povrchu data hloubkového senzoru, který je založen na době letu odraženého infračerveného paprsku, vykazují vysokou chybovost. Detekce polotovarů je tak provedena ve viditelném spektru. Metoda Template matching se však ukazuje být velmi citlivá na osvětlení scény. Při konzistentním osvětlení a správném nastavení parametrů

lze dosáhnout přijatelné spolehlivosti, avšak pouze pokud jsou polotovary vyskládány v jedné vrstvě. Při vyskládání polotovarů do více vrstev jsou touto metodou detekovány falešně pozitivní instance, které jsou způsobeny stíny nebo odrazy v lesklém povrchu.

Následně je implementováno řešení pomocí detektoru objektů YOLO v4, přičemž je využito již dosažených výsledků pomocí metody korelační analýzy. Díky spolehlivému algoritmu na určení polohy gitterboxu je úloha zjednodušena na detekci polotovarů uvnitř gitterboxu a nedokonalý algoritmus detekce samotných polotovarů je využit k vytvoření trénovacího datasetu. Pouze nevyhovující detekce vyžadují manuální úpravu a je tak možné rychle získat soubor anotovaných dat, který je navíc různými způsoby augmentován. Celkem je vytvořena databáze o rozsahu přibližně 1000 snímků, která je rozdělena na trénovací a testovací část v poměru 85/15. Naprosto dostatečné přesnosti detekce na testovacím datasetu je dosaženo již po 2000 iteracích supervizorovaného učení. Natrénovaný model detektoru objektů je následně implementován na nízkonákladový HW a díky akcelerací na grafickém adaptéru architektury CUDA je rychlost detekce přibližně 2 FPS. Nalezenou polohu je díky provedené registraci obrazů možné převést do snímků hloubkového senzoru a termokamery. Pomocí dat hloubkového senzoru je určena souřadnice z osy polotovarů. Zde je kvůli vysoké míře šumu způsobené zmiňovanou odrazivostí povrchu nutné využít filtraci a výsledná vzdálenost je určena jako nejčastěji zastoupená hodnota (modus) z hodnot v blízkosti osy. Díky snímku termokamery je možné obdobným způsobem určit jejich teplotu. Doba trvání jednoho cyklu od pořízení snímků po zobrazení výsledků detekce činí zhruba 2,5 vteřiny (0.4 FPS).

Výsledkem této práce bezesporu není produkt připravený k nasazení do průmyslu, avšak pouze prototyp, který demonstruje možnost využití senzorové fúze a použitých technologií pro řešení podobných úloh. Výše uvedené platí pro vytvořené grafické rozhraní, které neslouží pro interakci s koncovým uživatelem, ale pouze jako nástroj pro vývoj, a stejně tak samotný modul detekce, který je přizpůsoben použitému experimentálnímu uspořádání. Tento prototyp však nepochybně dokazuje způsobilost zvolené metodiky k tvorbě řešení vhodného k přímému nasazení do průmyslové praxe.

6 Bibliografie

[1] ELMENREICH, Wilfried. *An Introduction to Sensor Fusion*. Institut fur Technische Informatik Vienna University of Technology, Austria, 2002, , 27. Dostupné také z: https://www.researchgate.net/profile/Wilfried-Elmenreich/publication/267771481_An_Introduction_to_Sensor_Fusion/links/55d2e45908ae0a3417222dd9/An-Introduction-to-Sensor-Fusion.pdf

[2] KALMAN, R. E. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*. 1960, **82**(1), 35-45. ISSN 0021-9223. Dostupné z: doi:10.1115/1.3662552

[3] KHALEGHI, Bahador, Alaa KHAMIS, Fakhreddine O. KARRAY a Saiedeh N. RAZAVI. Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*. 2013, **14**(1), 28-44. ISSN 15662535. Dostupné z: doi:10.1016/j.inffus.2011.08.001

[4] SASIADEK, J.Z. Sensor fusion. *Annual Reviews in Control*. 2002, **26**(2), 203-228. ISSN 13675788. Dostupné z: doi:10.1016/S1367-5788(02)00045-7

[5] LI, Wangyan, Zidong WANG, Guoliang WEI, Lifeng MA, Jun HU a Derui DING. A Survey on Multisensor Fusion and Consensus Filtering for Sensor Networks. *Discrete Dynamics in Nature and Society*. 2015, **2015**, 1-12. ISSN 1026-0226. Dostupné z: doi:10.1155/2015/683701

[6] LECUN, Yann, Corinna CORTES a Christopher BURGES. *THE MNIST DATABASE of handwritten digits* [online]. [cit. 2021-05-09]. Dostupné z: http://yann.lecun.com/exdb/mnist/

[7] MILÁČEK, Stanislav. *Náhodné a chaotické jevy v mechanice*. Praha: Vydavatelství ČVUT, 2000. ISBN 80-010-2170-X.

[8] GRAHAM, Benjamin. *Fractional Max-Pooling*. Dept of Statistics, University of Warwick, 2015, , 10. Dostupné z: doi:arXiv:1412.6071

[9] BISHOP, Christopher. *Pattern recognition and machine learning*. [New York]: Springer, 2006. Information science and statistics. ISBN 978-0387310732.

[10] P. KINGMA, Diederik a Jimmy BA. *Adam: A Method for Stochastic Optimization*. 2015, , 15. Dostupné z: doi:arXiv:1412.6980v9

[11] HASTIE, Trevor, Robert TIBSHIRANI a Jerome FRIEDMAN. *The elements of statistical learning: data mining, inference, and prediction*. 2nd ed. New York: Springer, 2009. Springer series in statistics. ISBN 978-0-387-84858-7.

[12] Template Matching. *Open Source Computer Vision* [online]. [cit. 2021-01-27]. Dostupné z: https://docs.opencv.org/3.4/de/da9/tutorial_template_matching.html

[13] REDMON, Joseph, Santosh DIVVALA, Ross GIRSHICK a Ali FARHADI. You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, , 779-788. ISBN 978-1-4673-8851-1. Dostupné z: doi:10.1109/CVPR.2016.91

[14] GIRSHICK, Ross, Jeff DONAHUE, Trevor DARRELL a Jitendra MALIK. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2014, , 580-587. ISBN 978-1-4799-5118-5. Dostupné z: doi:10.1109/CVPR.2014.81

[15] GANDHI, Rohith. R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms. *Towards data science* [online]. [cit. 2021-04-11]. Dostupné z: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[16] GIRSHICK, Ross. Fast R-CNN. *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015, , 1440-1448. ISBN 978-1-4673-8391-2. Dostupné z: doi:10.1109/ICCV.2015.169

[17] REN, Shaoqing, Kaiming HE, Ross GIRSHICK a Jian SUN. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016.

[18] HE, Kaiming, Georgia GKIOXARI, Piotr DOLLAR a Ross GIRSHICK. Mask R-CNN. *Facebook AI Research (FAIR)*. 2018. Dostupné z: doi:arXiv:1703.06870

[19] BOCHKOVSKIY, Alexey, Chien YAO WANG a Hong YUAN MARK LIAO. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. Dostupné z: doi:arXiv:2004.10934

[20] WANG, Chien-Yao, Hong-Yuan LIAO, I-Hau YEH, Yueh-Hua WU, Ping-Yang CHEN a Jun-Wei HSIEH. CSPNET: A NEW BACKBONE THAT CAN ENHANCE LEARNING CAPABILITY OF CNN. *Institute of Information Science Academia Sinica, Taiwan*. 2019, , 14. Dostupné z: doi:arXiv:1911.11929v1

[21] HE, Kaiming, Xiangyu ZHANG, Shaoqing REN a Jian SUN. *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition*. 2015, , 14. Dostupné z: doi:arXiv:1406.4729v4

[22] LIU, Shu, Lu QI, Haifang QIN, Jianping SHI a Jiaya JIA. Path Aggregation Network for Instance Segmentation. *The Chinese University of Hong Kong §Peking University, SenseTime Research, YouTu Lab, Tencent*. 2018, , 11. Dostupné z: doi:arXiv:1803.01534

[23] BOCHKOVSKIY, Alexey, Chien-Yao WANG a Hong-Yuan MARK LIAO. *YOLOv3: An Incremental Improvement*. 2020, , 17. Dostupné z: doi:arXiv:2004.10934v1

[24] YUN, Sangdoo, Dongyoon HAN, Seong OH, Sanghyuk CHUN, Junsuk CHOE a Youngjoon YOO. CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features. *Clova AI Research, NAVER Corp., Clova AI Research, LINE Plus Corp., Yonsei University*. 2019, , 14. Dostupné z: doi:arXiv:1905.04899v2

[25] GHIASI, Golnaz a Quoc LE. DropBlock: A regularization method for convolutional networks. *Google Brain*. 2018, , 11. Dostupné z: doi:arXiv:1810.12890v1

[26] MISRA, Diganta. Mish: A Self Regularized Non-Monotonic Activation Function. *Landskape KIIT, Bhubaneswar, India*. 2020, , 14. Dostupné z: doi:arXiv:1908.08681v3

[27] GOTTESFELD BROWN, Lisa. A Survey of Image Registration Techniques. *ACM Comput,ng Surveys,VoI 24, No. 4*. Department of Computer Sctence, Colunzbza Unzl,ersity, New York, NY 10027, 1992.

[28] BAHNSEN, Chris. *Thermal-visible-depth image registration*. 2013. Dostupné také z: https://projekter.aau.dk/projekter/files/77295345/thesisfinal.pdf. Master's thesis. School of Information and Communication Technology. Vedoucí práce Thomas B. Moeslund.

[29] ZHAO, Jian a Sen-ching CHEUNG. Human segmentation by geometrically fusing visible-light and thermal imageries. *Multimedia Tools and Applications*. 2014, **73**(1), 61-89. ISSN 1380-7501. Dostupné z: doi:10.1007/s11042-012-1299-2

[30] KIM, Aerin. Camera Calibration: Camera Geometry and The Pinhole Model. *Towardsdatascience.com* [online]. [cit. 2021-07-14]. Dostupné z: https://towardsdatascience.com/camera-calibration-fda5beb373c3

[31] ZHANG, Z. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **22**(11), 1330-1334. ISSN 01628828. Dostupné z: doi:10.1109/34.888718

[32] Camera Calibration. *OpenCV: Open Source Computer Vision* [online]. [cit. 2021-07-14]. Dostupné z: https://docs.opencv.org/4.5.2/dc/dbb/tutorial_py_calibration.html

[33] What Is Camera Calibration?. *Mathworks.com* [online]. [cit. 2021-07-14]. Dostupné z: https://www.mathworks.com/help/vision/ug/camera-calibration.html

[34] BELONGIE, Serge a David KRIEGMAN. Homography Estimation. *Department of Computer Science and Engineering, University of California, San Diego.* [online]. , 3 [cit. 2021-07-15]. Dostupné z: http://cseweb.ucsd.edu/classes/wi07/cse252a/homography_estimation/homography_estimation.pdf

[35] FISCHLER, Martin A. a Robert C. BOLLES. Random sample consensus. *Communications of the ACM*. 1981, **24**(6), 381-395. ISSN 0001-0782. Dostupné z: doi:10.1145/358669.358692

[36] Feature Matching + Homography to find Objects. *Open Source Computer Vision* [online]. [cit. 2021-07-14]. Dostupné z: https://docs.opencv.org/4.5.2/d1/de0/tutorial_py_feature_homography.html

[37] OpenCV: findHomography(). *Open Source Computer Vision* [online]. [cit. 2021-07-15]. Dostupné z: https://docs.opencv.org/4.5.2/d9/d0c/group__calib3d.html#ga4abc2ece9fab9398f2e560d53c8c9780

[38] DUBROFSKY, Elan. *Homography Estimation*. 2007. Dostupné také z: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=3E6C84C8B8DB6254BCBACA5F5ED699EC?doi=10.1.1.186.4411&rep=rep1&type=pdf. Master's thesis. Department of Computer Science, University of British Columbia.

[39] Azure Kinect DK documentation. *Microsoft* [online]. [cit. 2021-04-02]. Dostupné z: https://docs.microsoft.com/en-us/azure/kinect-dk/

[40] *FLIR LEPTON® Engineering Datasheet*. Dostupné také z: https://cdn.shopify.com/s/files/1/0803/9211/files/flir-lepton-engineering-datasheet-203.pdf?v=1608437667

[41] GORORDO, Ibai. PyKinectAzure. *Github* [online]. [cit. 2021-05-26]. Dostupné z: https://github.com/ibaiGorordo/pyKinectAzure

[42] Rpc library. *OpenMV docs* [online]. [cit. 2021-05-26]. Dostupné z: https://docs.openmv.io/library/omv.rpc.html

[43] *OpenCV: Open Source Computer Vision* [online]. [cit. 2021-05-26]. Dostupné z: https://docs.opencv.org/master/

[44] REDMON, Joseph. *Dartknet: Open Source Neural Networks in C* [online]. [cit. 2021-05-25]. Dostupné z: https://pjreddie.com/darknet/

[45] BOCHKOVSKIY, Alexey. *Darknet github repository* [online]. [cit. 2021-05-25]. Dostupné z: https://github.com/AlexeyAB/darknet

[46] *OpenMV: Products* [online]. [cit. 2021-04-02]. Dostupné z: https://openmv.io/collections/products

[47] PATEL, Krut. *MNIST Handwritten Digits Classification using a Convolutional Neural Network (CNN)* [online]. [cit. 2021-05-09]. Dostupné z:

https://towardsdatascience.com/mnist-handwritten-digits-classification-using-a-convolutional-neural-network-cnn-af5fafbc35e9

[48] LI, Fei-Fei, Justin JOHNSON a Serena YEUNG. *Lecture 11: Detection and Segmentation* [online]. [cit. 2021-05-09]. Dostupné z: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf

[49] *Obrazové filtry: Konvoluce* [online]. Mendelova univerzita v Brně [cit. 2021-05-13]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=18431

[50] *Max-pooling / Pooling* [online]. [cit. 2021-05-14]. Dostupné z: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

[51] FERNANDEZ, Luis, Viviana AVILA a Luiz GONçALVES. A Generic Approach for Error Estimation of Depth Data from (Stereo and RGB-D) 3D Sensors. *Preprints (www.preprints.org)*. 2017, , 13. Dostupné z: doi:10.20944/preprints201705.0170.v1

[52] A. JENKINS, Francis a Harvey E.WHITE. *Fundamentals of Optics: Fourth Edition*. McGraw-Hili Primls Custom Publishing, 1976. ISBN 0-07-256191-2.

[53] Jetson Nano Developer Kit. *Nvidia developer* [online]. [cit. 2021-05-25]. Dostupné z: https://developer.nvidia.com/embedded/jetson-nano-developer-kit