# Řešení úkolu pro TurtleBot s detekcí objektů a SLAM

Michal Bouda, Erik Doležal, Ondřej Váňa

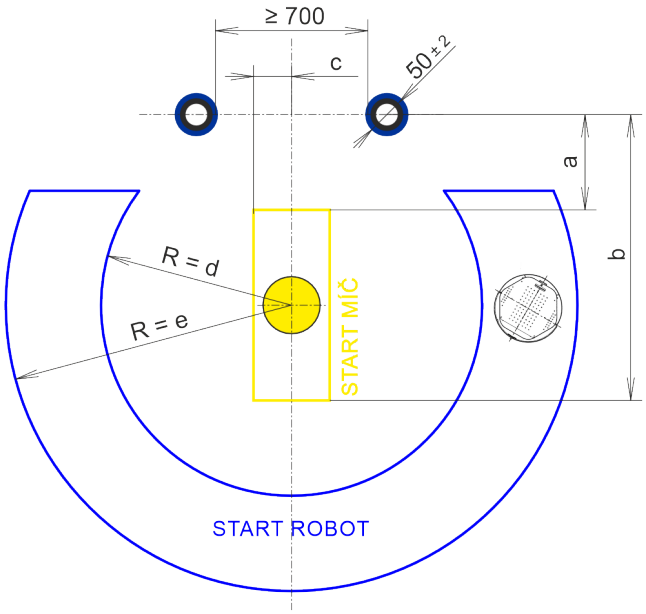
12. dubna 2025

## Obsah



## 1 Zadání

Úkolem práce je napsat řešení pro úkol, kde má dát Roomba TurtleBot gól do branky označené modrými pilíři.




Obrázek 1: Rozložení objektů v prostoru (ze zedání)

Rozložení problému je vidět na obrázku (1). Robot má za úkol najít míč a ten poté dostat mezi modré pilíře, které jsou od sebe vzdálené minimálně 700 mm. Míč je vzdálen od osy branky maximálně c = 300 mm. Míč se bude nacházet v minimální vzdálenosti a = 500 mmod brány a bude nejdále b = 2m. TurtleBot bude na začátku umístěn od středu míče ve vzdálenosti d = 0,7 m až e = 2,5 m. V prostoru před branou se mohou vyskytovat překážky v podobě zelených a červených pilířů. Za bránou může být neomezený počet pilířů.

# 2 Řešení

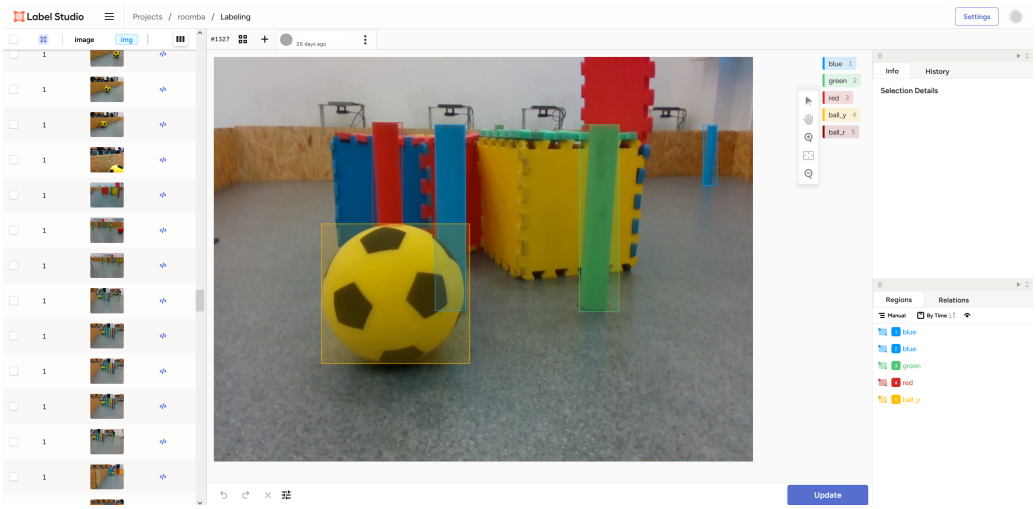
Naše řešení jsme rozdělili na tři hlavní části. První část je zpracování obrazu, druhá část je SLAM a třetí část je plánování a pohyb robota. Tyto části běží v hlavní smyčce programu. Při řešení problému bylo využíváno nástroje GitHub Copilat pro jednoduché úkony pro zrychlení práce.

## 2.1 Zpracování obrazu

Detekci objektů děláme pomocí konvoluční neuronové sítě YOLO (You Only Look Once). Důvodem k tomuto rozhodnutí bylo, aby náše řešení dobře zvládalo změny v osvětlení a jiné rušivé vstupy jako například špinavý žlutý míč. Segementace obrazu pomocí barev, by tak mohla být nespolehlivá. Jako vstup používáme obraz z Intel RealSense D435 kamery.

### 2.1.1 Trénování CNN

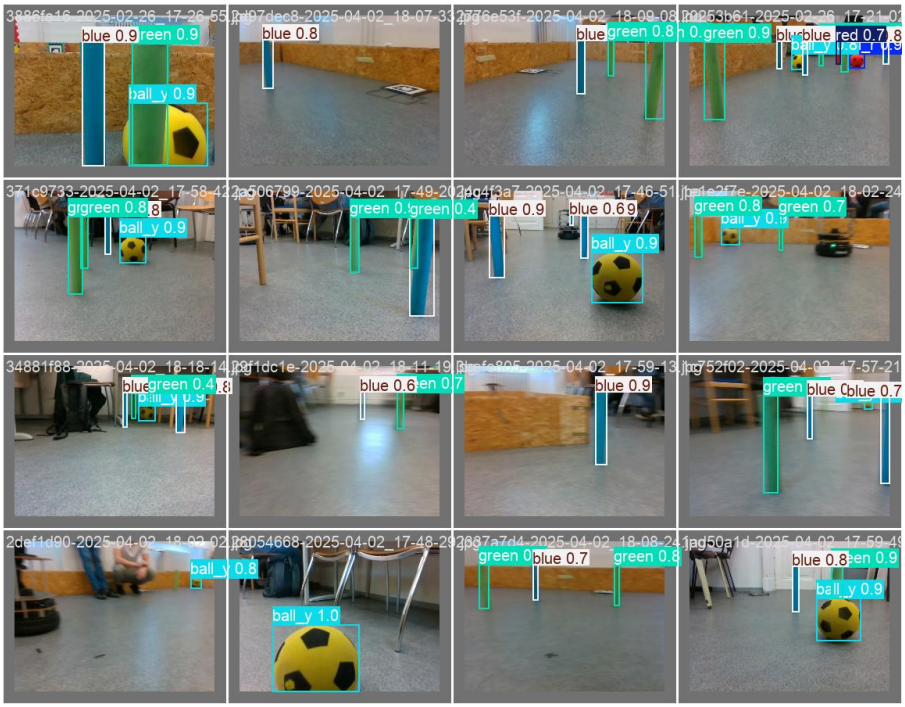
YOLO model jsme museli nejdříve natrénovat na rozponávání pilířů a míče. Učinili jsme tak na více než 670 obrázcích, které jsme pořídili pomocí kamery na robotovi. Dalších 120 jsme použili pro validaci. Obrázky jsme ručně anotovali pomocí programu Label Studio. Jak vypadá anotace je vidět v obrázku (2).



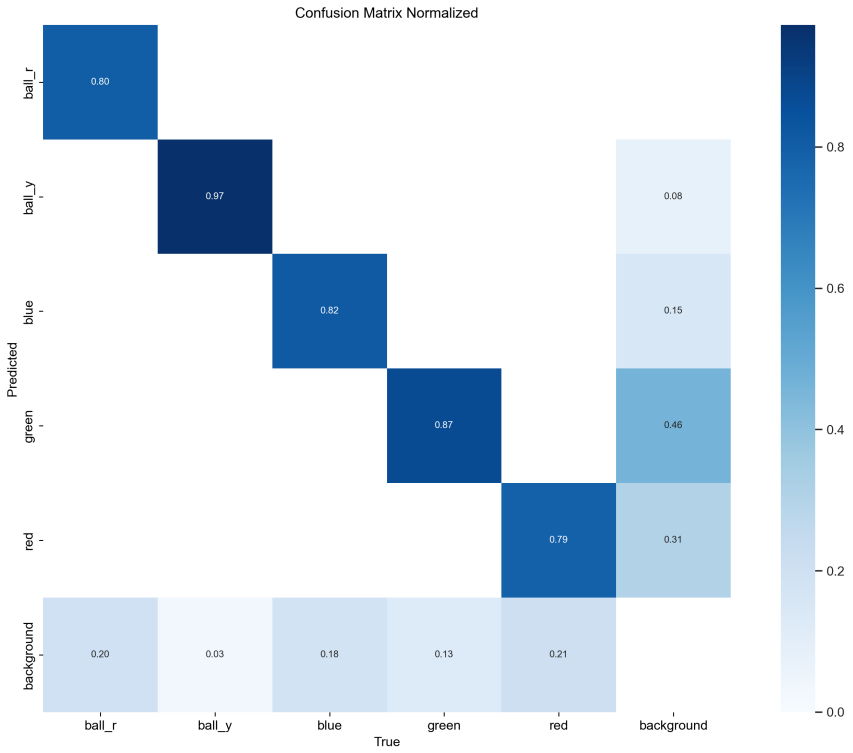

Obrázek 2: Anotování v Label Studiu

Pro anotaci jsme použili nastavení *Object Detection with Bounding Boxes*, tedy jsme anotavali pomocí obdelníků. Zvolili jsme možnost bez rotace, všechny obdelníky mají tedy rovnoběžné strany se stranami obrazu. Toto představilo problém, který jde vidět v obrázku (3). Díky zakřivení obrazu z kamery, obdelníky nesedí přímo na pilíře, což byl lehce vyřešitelný problém (2.1.3). Možnost segmentace, myšleno maskou jsme zamítli z různých důvodou, např. pracné

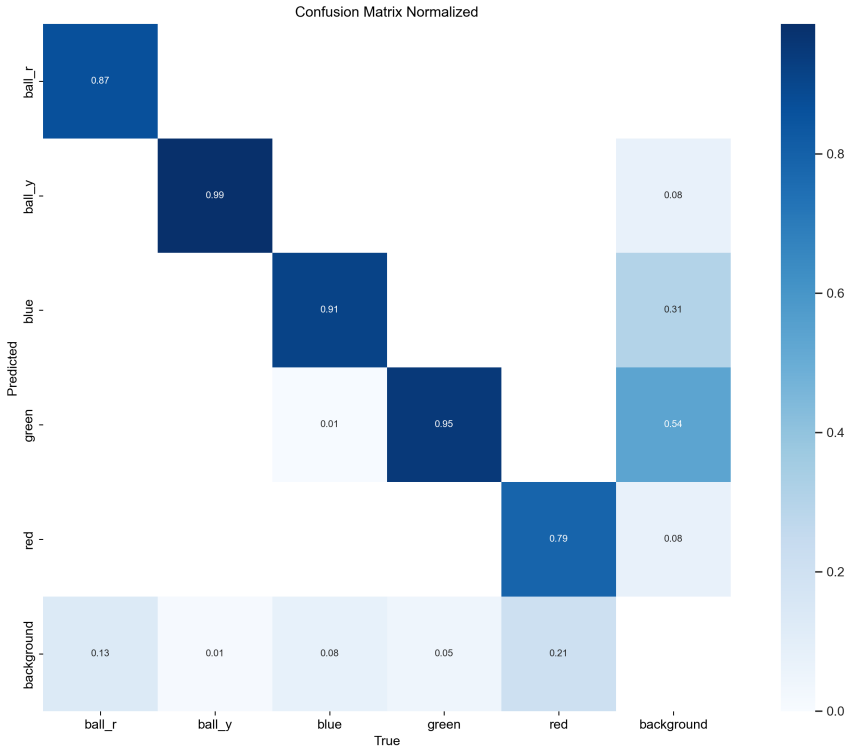
anotace. Anotovaný data set byl poté vyexportován ve formátu YOLO. Pomocí Python kódu a knihovny od Ultralitics jsme natrénovali model. Vyzkoušeli jsme YOLO verze v8 a 11. Ukázalo se, že verze 11 je mnohem přesnější, zvláště při změně osvětlení. Protože detekce musí probíhat rychle, zkoušeli jsme modeli 11s a 11n. I když je model 11s přesnější, byl pro naše použití již moc pamalý. Běžel déle než 50ms. Pro model 11n jsme zkoušeli i různé rozlišení. Rozlišení 160p a 240p bylo zpracováno dostatečně rychle a s uspokojivími výsledky. Oba modely byly natrénované na 300 epochách.




Obrázek 3: Rozpoznané objekty pomocí YOLO 11n při rozlišení 240p




Obrázek 4: Normalizovaná matice záměn pro model 11n 160p




Obrázek 5: Normalizovaná matice záměn pro model 11n 240p

Model 160p (4) je v některých podmínkách znatelně horší oproti 240p (5) v rozpoznávání modrých pilířů. To může způsobit velké problémy, protože modré pilíře tvoří branku. Nejčastějším objektem, který byl zaměněn s pozadím, je zelený pilíř. Který, když je detekován navíc,

minimálně překaží úspěšnému vyřešení problému. V obrázku (3) si ještě můžeme všimnout, že model dobře zvládá rozpoznávání objektů, které mají zaměnitelné barvy nebo tvar s objekty, které detekuje. Model například rozpoznal míč, když je z čísti za pilířem, nebo objekty když jsou ve stínu pod stolem.

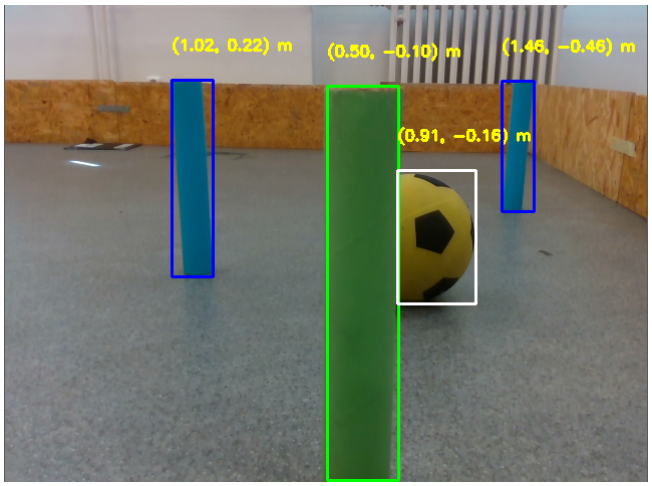
### 2.1.2 Rozpoznávání obrazu

K rozpoznávání používáme buď model YOLO 11n při rozlišení 160p nebo při 240p v závislosti na prostředí a pokud více benefitujeme z rychlejší detekce nebo její přesnosti. Na robotovi zajišťuje rozpoznávání class `Camera`. Třída má metodu `get detections`, která získá obraz z kamery. Seznam objektů vracíme pro potřeby SLAM jako numpy array poloha x, poloha y, objekt. Poloha x, y je poloha relativně k robotovi v prostoru. K detekci nepoužíváme knihovnu YOLO.

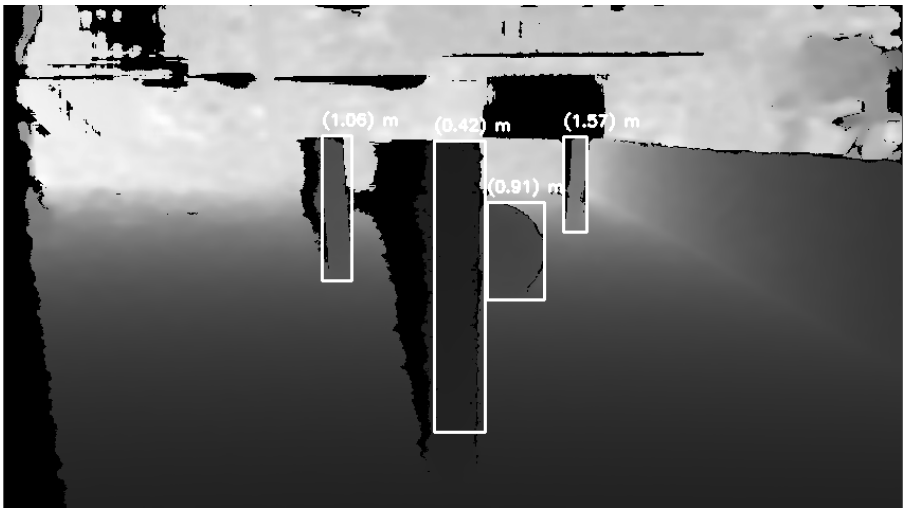
Samotné rozpoznávání ale neběží pomocí YOLO. Model nejdříve konvertujeme do ONNX (Open Neural Network Exchange). Důvodem pro toto rozhodnutí je dlouhá doba, kdy YOLO knihovna zpracovála obraz. ONNX to zvládá rychleji z části díky tomu, že je optimalizovaná pro spouštění na procesoru. Toto ale přineslo spoustu výzev, protože YOLO knihovna řešila spoustu věci za nás. Museli jsme naimplementovat počítaní pravděpodobností detekce pomocí softmax. Také pomocí Non-Maximum Suppression z knihovny TorchVision řešíme odstranění duplicitních detekcí. A samozřejmostí je nepracování s detekcemi, které nedosahují nějaké hranice jistoty.

### 2.1.3 Pozice objektů v prostoru

Z počátku jsme využívali k určování pozice objektů point cloud, který jsme získali z kamery. To se ukázalo jako velice časově náročné. Proto jsme se rozhodli, že budeme počítat pozici objektů přímo z hloubkové kamery.




Obrázek 6: Obrázek z RGB kamery s rozpoznanými objekty




Obrázek 7: Obrázek z hloubkové kamery s rozpoznanými objekty

Hloubková kamera vrací pouze pixeli s hloubkou. Aby byla poloha objektů spočítána co nejrychleji, pracujeme pouze s pixeli z hloubkové kamery (7), které jsou detekovány jako objekty pomocí RGB kamery (6). Nejdříve vytvoříme transformační matici, která převadí mezi souřadnicemi kamery a hloubkové kamery. Poté se vypočítá median souřadnicí v bounding boxu, což řeší problém s obdelníky přesně nepasujícími na detekované objekty. Vyřešíme také 10 stupňový sklon kamery a pomocí K matice převedeme souřadnice z pixelů na reálné souřadnice. Za souřadnice detekce se také přidá třída objektu.

## 2.2 SLAM

## 2.3 Plánování

## 2.4 Pohyb

# 3 Závěr