VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

VYUŽITÍ DVOUROZMĚRNÝCH ČÁROVÝCH KÓDŮ PRO ODHAD POLOHY KAMERY V APLIKACÍCH AR

USAGE OF TWO-DIMENSIONAL BARCODES FOR CAMERA POSITION ESTIMATION IN AR APPLICATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

Daniel Boháč

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Roman Adámek

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Daniel Boháč**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**Ing. Roman Adámek**
Akademický rok:	2019/20

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

Využití dvourozměrných čárových kódů pro odhad polohy kamery v aplikacích AR

Stručná charakteristika problematiky úkolu:

Pro lokalizaci kamery v prostoru se v současné době využívá řada metod, mimo jiné i speciální referenční značky. Tyto značky jsou využívány pouze k odhadu polohy kamery a jejich identifikaci, neobsahují ale žádné další informace. Naproti tomu dvourozměrné čárové kódy jsou navrženy k tomu, aby umožnily uchovávat informace a svými charakteristikami připomínají výše zmíněné speciální referenční značky. Cílem této práce je prozkoumat možnosti využití dvourozměrných čárových kódů pro odhad polohy kamery v porovnání s běžně užívanými referenčními značkami. Dále také vytvoření ukázkové aplikace využívající rozšířenou realitu, ve které budou s výhodou využity dvourozměrné čárové kódy.

Cíle bakalářské práce:

1. Proveďte rozbor dostupných dvourozměrných čárových kódů používaných pro ukládání informací z hlediska jejich kapacity, licenčních práv a snadnosti detekce.
2. Na základě rozboru vyberte jeden nebo více vhodných dvourozměrných kódů a vytvořte algoritmus pro odhad polohy kamery vůči těmto kódům. Udělejte algoritmus univerzální, aby bylo možné odhadovat polohu více kódů v obraze.
3. Porovnejte přesnost odhadu polohy vámi zvoleného kódu vzhledem k referenčním značkám přímo určeným k odhadování polohy v rozšířené realitě.
4. Vytvořte ukázkovou aplikaci využívající principy rozšířené reality s využitím vybraného dvourozměrného čárového kódu, určeného k ukládání dat. Tuto aplikaci navrhněte tak, aby se v ní ukázaly výhody dvourozměrných čárových kódů oproti referenčním značkám.

Seznam doporučené literatury:

TRUCCO, Emanuele a Alessandro VERRI. Introductory techniques for 3-D computer vision. Upper Saddle River, NJ: Prentice Hall, c1998. ISBN 0132611082.

SOLEM, Jan Erik. Programming computer vision with Python. Sebastopol, CA: O'Reilly, 2012. ISBN 1449316549.

KAEHLER, Adrian a Gary R. BRADSKI. Learning OpenCV 3: computer vision in C++ with the OpenCV library. Sebastopol, CA: O'Reilly Media, [2017]. ISBN 1491937998.

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

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc.
ředitel ústavu

doc. Ing. Jaroslav Katolický, Ph.D.
děkan fakulty

Abstrakt

Tato práce se zabývá tvorbou algoritmu pro odhad polohy kamery využitím QR kódů v aplikacích rozšířené reality. Ten musí zvládnout pracovat i s více kódy v obraze. K tomu je využito knihovny počítačového vidění OpenCV a programovacího jazyka Python. Algoritmus je poté zahrnut do jednoduše použitelné knihovny. Vytvořené řešení je porovnáváno vůči ArUco referenčním značkám. Nakonec je vytvořena ukázková aplikace.

Summary

This thesis deals with the creation of algorithm for camera pose estimation using QR codes in augmented reality applications. It must be able to work with multiple codes in the image. The OpenCV computer vision library and the Python programming language are used for this. The algorithm is then included in an easy-to-use library. The created solution is compared against ArUco markers. Finally, a sample application is created.

Klíčová slova

Python, OpenCV, QR kód, počítačová vize, odhad polohy, PnP problém, rozšířená realita, referenční značky, ArUco

Keywords

Python, OpenCV, QR code, computer vision, pose estimation, PnP problem, augmented reality, fiducial markers, ArUco

Bibliografická citace

BOHÁČ, Daniel. *Využití dvourozměrných čárových kódů pro odhad polohy kamery v aplikacích AR*. Brno, 2020. Dostupné také z: `https://www.vutbr.cz/studenti/zav-prace/detail/124934`. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky. Vedoucí práce Roman Adámek.

Prohlašuji, že jsem bakalářskou práci *Využití dvourozměrných čárových kódů pro odhad polohy kamery v aplikacích AR* vypracoval samostatně pod vedením Ing. Romana Adámka, s použitím uvedených zdrojů.

Daniel Boháč

Brno . . . .

Rád bych poděkoval Ing. Romanu Adámkovi za odborné vedení práce, jeho cenné rady a trpělivost. Dále patří velké díky přátelům a rodině, za jejich podporu.

Daniel Boháč

Obsah

1 Úvod

Rozšířená realita Augmented Reality (AR) spočívá ve vykreslování počítačové grafiky nad obraz z kamery tak, že jej informačně obohacuje. To může být ve formě textových informací, obrázků, či 3D modelů. Existuje způsob, kterým lze grafiku vykreslit do jiného souřadnicového systému než toho určeného kamerou. Nejprve je však nutné zjistit vztah mezi nimi. Tento problém je v podstatě odhadem polohy kamery a jde o jeden ze základních úkonů prováděných v oblasti počítačového vidění. V aplikacích AR navíc musí probíhat v reálném čase.

Ačkoli jsou myšlenky, na kterých je odhad polohy založen, pro většinu metod podobné, způsoby realizace se různí. Některé zakládají na více kamerách, jiné berou v úvahu i data z dalších senzorů. Jeden ze specifických přístupů používá referenční značky. Ty představují záchytné body v prostoru, pomocí kterých je vytvořen souřadnicový systém, vůči kterému je poloha kamery odhadována.

Hojně využívané jsou tisknutelné čtvercové referenční značky. Ty jsou vytvořeny, pro snadnou detekci v obraze a jednoznačné určení jejich prostorové orientace. Rohy těchto značek jsou využity pro sestrojení souřadnicového systému náležícího konkrétní značce. To pak umožňuje odhad vzájemné polohy značky a kamery, či již zmíněnou projekci grafiky. Nevýhodou těchto značek je, že jednotlivé značky musí být definovány předem a nenesou jinou informaci, než identifikátor.

Dvourozměrné čárové kódy mají s čtvercovými referenčními značkami na první pohled velmi podobné provedení. Zejména u těch, které taktéž tvoří čtyři rohy, se nabízí jejich využití pro odhad polohy kamery. Zároveň díky tomu, že slouží pro uchování dat, a jsou k nim tedy vytvořené dekodéry, není nutné je předem jednotlivě definovat. Potenciálně pak nabízejí možnosti aplikací, kde jsou samotné instrukce, např. pro vizualizaci, přímo zakódovány do datového obsahu.

Tato bakalářská práce se nejprve zabývá výběrem vhodného dvourozměrného čárového kódu pro odhad polohy kamery. Poté již přípravou na tvorbu algoritmu právě tohoto procesu. Přičemž jsou hodnoceny a voleny nástroje i postupy. Těmi je myšleno např. knihovna počítačového vidění, programovací jazyk a postup při odhadu polohy kamery vůči referenčním značkám. S každou volbou je pak oblast zájmu zužována. Důležitou požadovanou funkcionalitou algoritmu je schopnost odhadu polohy více dvourozměrných čárových kódů v obraze. Během řešení bylo upřesněno, že algoritmus má být zahrnut do jednoduše použitelné knihovny.

Praktická část se zabývá samotnou tvorbou zmíněné knihovny, tedy i algoritmu odhadu polohy kamery vůči zvolenému dvourozměrnému čárovému kódu. Vytvořené řešení je pak porovnáváno, prostřednictvím měření, vůči již existujícímu řešení se čtvercovými referenčními značkami. Nakonec je vytvořena ukázková aplikace, využívající vytvořeného algoritmu a principů rozšířené reality.

2 Rešerše

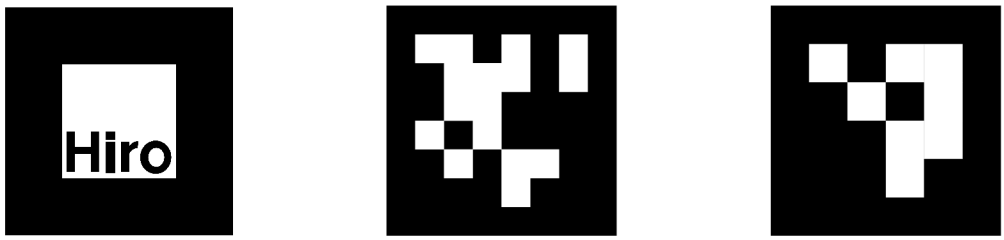
Prvním krokem k dosažení cílů je rešerše. Jejích předmětů je vícero. Ze všeho nejdříve jsou to referenční značky a dvourozměrné čárové kódy. Poté je zkoumán samotný postup pro odhad polohy kamery využitím referenčních značek. Dále jsou to dostupné knihovny pro počítačové vidění a dekódování vybraných dvourozměrných čárových kódů, využití těchto knihoven pro splnění cílů a nakonec podobná, již realizovaná řešení. Závěry plynoucí z jednotlivých částí rešerše mají vliv na část další. S postupem je oblast zájmu vždy více konkrétní.

2.1 Referenční značky

Referenčními značkami (markery) jsou v této práci myšleny rovinné značky, které se využívají v aplikacích AR, pro relativní odhad polohy kamery vůči nim. Značky mohou být buďto umístěny napevno k okolí, nebo na pohybujících se předmětech. Pro jednoznačné řešení odhadu polohy kamery, musí značka poskytovat alespoň čtyři snadno detekovatelné body. Většinou jimi jsou vnější rohy značky. K jejich detekci je často nutné zajistit i tzv. tichou zónu, což je oblast okolo značky, do které nesmí nic zasahovat (pouze podklad). [1]

Aby bylo možné používat více značek najednou, nesmí být stejné. Vnější uspořádání často bývá u značek jednoho typu shodné, vnitřní naopak odlišné, právě pro určení identity a prostorové orientace. Identifikace je řešena buďto předdefinováním omezeného počtu jednotlivých značek, nebo zakódováním identifikátoru do vnitřního uspořádání. Jednotlivé značky mezi sebou nesmí být snadno zaměnitelné. [1]

Ideální referenční značka je ze všech směrů detekovatelná pod stejnými úhly. Často jsou referenční značky čtvercové, mohou být ale i kruhové. Pro snadnou detekci je vhodné, aby byly značky co nejvíce kontrastní. Nejčastějším provedením je černá na bílém podkladu, či naopak. Existují i značky vícebarevné, to sice navyšuje možný počet identifikátorů, avšak detekce je problematická, zejména kvůli větší závislosti na světelných podmínkách. [1]



Obrázek 2.1: Referenční značky. Zleva: ARToolKit marker [2], Apriltag [3], ArUco marker [4]

Je velké množství vytvořených provedení takovýchto značek. Jednotlivá řešení, umožňující i jejich další konfiguraci, se označují jako systémy referenčních značek. Na Obr. 2.1 jsou zástupci tří takových systémů.

ARToolKit podoporuje především čtvercové markery s poměrně tlustým okrajem. Do jejich středu je možné umístit libovolný obrazec. Taková značka však musí být před použitím nejdříve natrénována. ARToolKit je dostupný pod General Public License (GPL). [5]

AprilTag podporuje vícero možných rozložení značek (nemusí být pouze čtvercové). Skupiny značek jsou generovány předem, nedochází tedy k žádnému trénování, ale pouze k předdefinování. To je možné díky zakódovanému identifikátoru. Zpracování těchto značek je možné v reálném čase i na slabším hardwaru. AprilTag je dostupný pod Berkley Software Distribution (BSD) licencí. [6]

2.1.1 ArUco

Je to systém referenčních značek vyvinutý výzkumnou skupinou Applications of Artificial Vision univerzity Cordoba, poprvé publikovaný roku 2014. Markery jsou tvořené čtvercovou mřížkou s černým okrajem o šířce jednoho pole (může být i širší). Vnitřní uspořádání představuje binární kód, jež slouží k identifikaci jednotlivých markerů. Předdefinování se řeší již připravenými skupinami tzv. slovníky. Těch existuje vícero, dělí se podle rozměrů vnitřní mřížky a počtu markerů ve slovníku. Knihovna tohoto systému umožňuje velice rychlou detekci a snadné použití. Zejména jednoduché využití tohoto systému usnadnilo jeho rozšíření a jedná se zřejmě o nejpopulárnější systém referenčních značek. Původní verze vznikla pod BSD licencí, avšak nejnovější verze je pod GPLv3. [7, 8]

2.2 Dvourozměrné čárové kódy

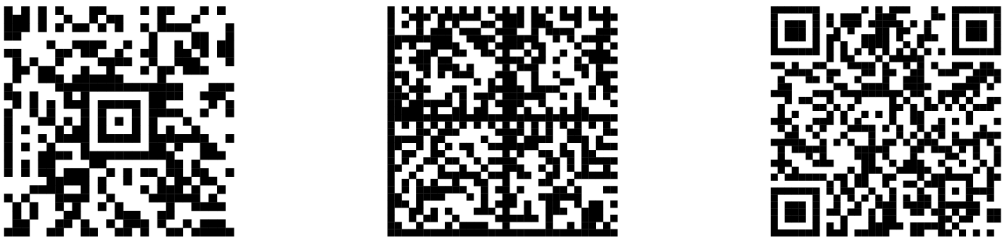
Jako čárové kódy jsou označovány strojově čitelné obrazce, uchovávající informace v podobě binárního kódu. Dvourozměrné čárové kódy nabízejí větší datovou kapacitu na stejné ploše oproti jednorozměrným. To díky bitům zapsaných ve dvou směrech namísto pouze jednoho. [9]

Mají široké možnosti využití. Používají se všude tam, kde je vhodné zapisovat větší množství informací na malou plochu. To je od výroby součástek, přes montáž, až po reklamní účely. Zejména v automatizovaných výrobách představují důležitý sdělovací prostředek, mezi objektem a strojem. Dále tyto kódy zastávají významnou roli v logistice, při identifikaci zásilek a sdělování dalších instrukcí. [9]

Existuje mnoho typů těchto kódů. Každý z nich má specifický, neměnný vzor, pomocí kterého lze daný kód detekovat. Některé kódy mají i jisté výhody. Těmi může být např. zabezpečení dat, samoopravné algoritmy či schopnost zakódovat neobvyklé znaky. Většinou mají čtvercový tvar a černobílé (případně jinak kontrastní) provedení. Možné jsou i vícebarevné kódy, pro ještě větší datovou kapacitu. [9]

Mnoho vytvořených typů dvourozměrných čárových kódů se neujalo. Jejich rozšíření je ovlivněno mnoha faktory. Těmi jsou např. licence, potřebné podmínky, datová kapacita, dostupnost dokumentace a nástrojů pro generování/zpracování či jejich rychlost. Zejména licence je pro větší rozšíření klíčová. Potřebnými podmínkami, jsou myšleny podmínky světelné, materiálové a požadavky na kvalitu čtecího zařízení.

S ohledem na sekci 2.1 jsou předmětem dalšího zájmu čtvercové, snadno detekovatelné dvourozměrné čárové kódy s co největší datovou kapacitou a dostupnými nástroji pro jejich generování i zpracování.



Obrázek 2.2: Dvourozměrné datové kódy. Vygenerovány pomocí [10]. Všechny obsahují stejný řetězec názvu práce bez interpunkce. Zleva: Aztecký kód, Data Matrix, QR kód.

2.2.1 Aztécký kód

Patentován roku 1995, později uvolněn jako volné dílo. Roku 2008 se stal mezinárodním standardem pod normou ISO/IEC 24778. Je to čtvercový dvourozměrný čárový kód s charakteristickou značkou v jeho středu, připomínající vzhled Aztéckých pyramid. Odtud plyne také jeho název. Tato značka slouží k detekci kódu, přitom nemá symetrický okraj, aby mohla sloužit i pro určení jeho orientace. Datová oblast se pak rozprostírá okolo této středové značky. Vyznačuje se tím, že nevyžaduje tichou zónu okolo kódu, ačkoli některá čtecí zařízení ji přesto potřebují. Má 32 velikostních verzí a je schopen pojmout až 3067 alfanumerických znaků. Zároveň podporuje i samoopravné kódování (Reedovy-Solomonovy kódy). Aztécké kódy se využívají především v dopravě pro identifikaci jízdenek. Na Obr. 2.2 je příklad tohoto kódu. [11, 12, 13]

2.2.2 Data Matrix

Původní Data Matrix dvourozměrný čárový kód vznikl už roku 1987. Dodnes používaná verze až poté v roce 1995. Ta je označována jako ECC 200 a byla uvolněna jako volné dílo. Roku 2000 se Data Matrix stal mezinárodním standardem pod normou ISO/IEC 16022. Má tvar čtverce, či obdélníku, jehož vnější okraj představuje značku pro detekci a určení orientace. Datová oblast se pak nachází uvnitř tohoto okraje. Má více velikostních verzí a může pojmout až 2335 alfanumerických znaků. Při větším datovém obsahu je vnitřně rozdělen na vícero datových oblastí, jako je vidět i na Obr. 2.2. Kód vyžaduje tichou zónu okolo svého obvodu. Verze ECC 200 podporuje i použití samoopravných kódů. Používá se pro přímý popis součástek (často laserem), především elektronických a v automotive. Jde nejspíše o druhý nejznámější dvourozměrný čárový kód. [9, 12, 14]

2.2.3 QR kód

Představen a patentován roku 1994 společností Denso Wave Inc., ta se však zavázala patent neuplatňovat. Roku 2000 se stal mezinárodním standardem pod normou ISO/IEC 18004. Je to čtvercový dvourozměrný čárový kód se třemi detekovatelnými značkami v jeho rozích. Pomocí nich lze určit i orientaci kódu. Jelikož byl kladen důraz na schopnost detekce ze všech stran, jsou tyto značky poměrně velké a oddělené od datové oblasti. Zároveň byl kladen důraz i na rychlost odezvy. Kód vyžaduje zajištění tiché zóny kolem svého obvodu. Má 40 velikostních verzí a dokáže pojmout až 4296 alfanumerických znaků. Od velikostní verze 2 obsahuje navíc značky zarovnání, které mají usnadňovat detekci pod úhlem. Podporuje čtyři úrovně opravitelnosti využitím samoopravných kódů. Existuje mnoho odvozených verzí tohoto kódu. Jedná se zřejmě o nejznámější dvourozměrný čárový kód, zejména i díky možnosti jejich čtení chytrými telefony. Kromě použití ve výrobě, či logistice se s ním lze setkat v tisku, reklamních médiích, či na webových stránkách. Příklad QR kódu je na Obr. 2.2. [15]

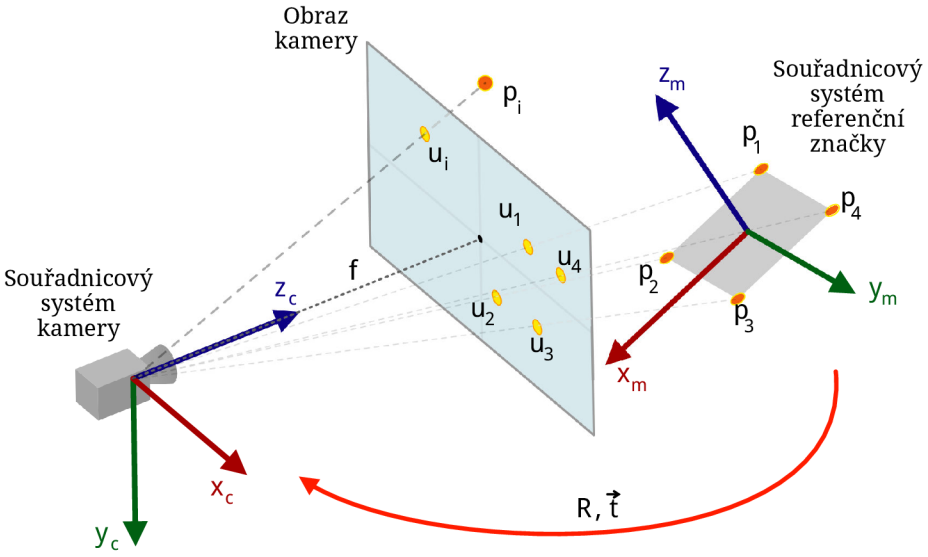
2.2.4 Zhodnocení

Jelikož je možné všechny tři kandidáty použít bez licence, nehraje tato skutečnost při výběru roli. Dále není důležitá maximální datová kapacita, jelikož jsou kódy s touto kapacitou příliš rozsáhlé, což znesnadňuje jejich detekci i dekódování za neideálních podmínek. Nelze ani jednoduše porovnávat datovou hustotu kandidátů, zejména protože do ní vstupuje více proměnných. Těmi jsou např. úroveň samoopravitelnosti, typ enkódovaných dat a další. Zběžným hledáním na internetu lze usoudit, že nejdostupnější jsou nástroje pro generaci a zpracování QR kódů. Tím je myšleno množství knihoven a jimi podporovaných operačních systémů i programovacích jazyků. Navíc je QR kód již navržen s ohledem na to, aby jej bylo možné detekovat z co nejvíce možných směrů. V případě Aztéckého kódu a Data Matrix, se spíše předpokládá přímé skenování (kolmo na kód). Dále je tedy rešerše prováděna s ohledem na výběr QR kódu, jako dvourozměrného čárového kódu pro který bude vytvořen algoritmus odhadu polohy kamery.

2.3 Odhad polohy kamery pomocí referenčních značek

Základní úlohou v oblasti AR je odhad polohy kamery vůči okolí. V tomto případě vůči referenční značce. Dále jsou uvažovány pouze aplikace AR pracující v reálném čase, tedy i odhad polohy kamery musí probíhat v reálném čase. Jde o to, pomocí 2D souřadnic bodů (rohů referenční značky) v obraze a známého rozvržení těchto bodů v 3D prostoru (rozměry značky), zjistit vztah mezi nimi. Nejvíce používanou cestou pro vyjádření tohoto vztahu je řešení tzv. Perspective n-Point (PnP) problému. [16]

2.3.1 PnP problém



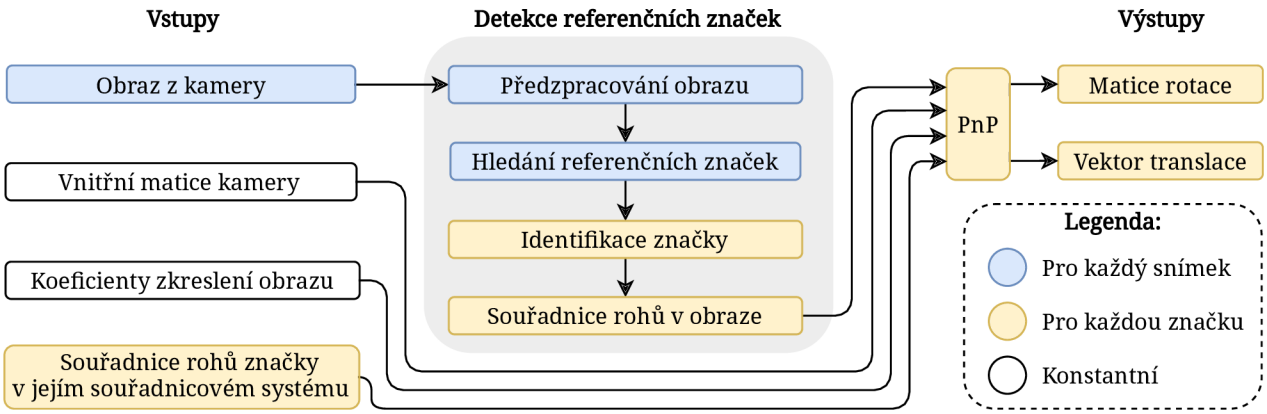
Obrázek 2.3: PnP problém. Vytvořeno podle [17] s úpravami.

Na Obr. 2.3 jsou znázorněny důležité části PnP problému. Souřadnice bodů $\mathbf{p_i}$ jsou známé v (3D) souřadnicovém systému referenční značky. Souřadnice bodů $\mathbf{u_i}$ jsou zase známy ve (2D) souřadnicovém systému obrazu kamery, který je stejný jako souřadnicový systém kamery, avšak nemá osu z. Body $\mathbf{u_i}$ představují 2D projekci bodů $\mathbf{p_i}$. Matice rotace $\mathbf{R}$ (3×3) a vektor translace $\vec{\mathbf{t}}$ (3×1) jsou neznámými. Vyjadřují změnu mezi souřadnicovými systémy, tudíž i vzájemnou polohu kamery vůči referenční značce. Dále je nutné pro řešení poskytnout vnitřní matici kamery $\mathbf{K}$ (3×3) a koeficienty zkreslení obrazu. Ty se zjišťují kalibrací kamery.

Aby měl PnP problém řešení, je nutné použít alespoň tři body. Při použití tří bodů má však úloha více řešení, přidá-li se bod čtvrtý, nejednoznačnost značně klesá. Existuje vícero algoritmů pro řešení tohoto problému s různými výhodami i nevýhodami. Zejména se liší výpočetní náročností a přesností odhadu polohy. Zároveň existují i řešiče optimalizované pro speciální případy, jako jsou čtvercové značky. [16]

2.3.2 Postup pro obdržení matice rotace a vektoru translace

Jak již bylo uvedeno v podsekci 2.3.1, klíčovým procesem při odhadu polohy kamery je řešení PnP problému a byly zmíněny i jeho důležité části. Na Obr. 2.4 je blíže znázorněn postup pro získání matice rotace a vektoru translace pomocí referenčních značek. Matice rotace může být převedena i na vektor, což je její nejkompaktnější vyjádření. Vektory rotace a translace mají potom dohromady šest hodnot. To koresponduje se šesti stupni volnosti tělesa v 3D prostoru. Dohromady tedy zcela určují změnu polohy. Vnitřní matice kamery a koeficienty zkreslení obrazu jsou neměnné pro danou kameru, pokud se nemanipuluje s jejím optickým uspořádáním. Souřadnice rohů značky v jejím souřadnicovém systému je nutno předem definovat. Pokud mají značky různé rozměry, tak musí být pro každou značku definovány zvlášť. Při velkém počtu referenčních značek může dojít k poklesu snímkovací frekvence kamery v důsledku velkého množství náročnějších výpočtů. [16]



Obrázek 2.4: Postup pro obdržení matice rotace a vektoru translace pomocí referenčních značek. Vytvořeno zobecněním návodů z [16].

2.3.3 Kalibrace kamery

Pro odhad polohy kamery, se využívá její dírkový model (pinhole camera model). Ten zjednodušeně popisuje geometrii, kterou vzniká obraz. Obr. 2.3 vychází právě z dírkového modelu kamery, kde f představuje ohniskovou vzdálenost. Ta tvoří součást modelu popisující kameru a vyskytuje se ve vnitřní matici kamery $\mathbf{K}$, jíž lze získat právě kalibrací. Dírkový model se však odlišuje od reality, proto je třeba zahrnout parametry, které jej realitě více přiblíží. První vychází z umístění snímacího čipu. Jeho střed nemusí být, v důsledku výrobních limitací, umístěn přesně na optické ose. Ta je zároveň nositelkou osy z souřadnicového systému kamery. Průsečík této osy a snímacího čipu je centrem projekce. Polohový rozdíl tohoto průsečíku a středu snímacího čipu pak vyjadřují (v pixelech) parametry c_x a c_y. Další parametry vycházejí z geometrie elementů snímacího čipu. Jsou jimi ohniskové vzdálenosti f_x a f_y s rozměrem pixelů. To vyplývá ze vztahů $f_x = f \cdot s_x$ a $f_y = f \cdot s_y$, kde f je ohnisková vzdálenost v milimetrech a s je rozměrový poměr pixelů na milimetr. Strukturu vnitřní matice kamery lze pak vidět ve vztahu 2.1. [18, 16]

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.1}$$

Dále se ještě dírkový model rozchází s reálnou kamerou tím, že je namísto velmi malé dírky použita čočka. To vnáší do obrazu zkreslení, jelikož ideální čočku nelze vyrobit. Vždy se uvažují alespoň dva hlavní druhy zkreslení: radiální a tangenciální. První zmíněné je způsobeno tvarem čočky a potlačuje se, u běžných webkamer, dvěma korekčními koeficienty. U kamer s velkým radiálním zkreslením přidáváme třetí, případně i více. Tangenciální pak vzniká, pokud není snímací čip přesně kolmý na optickou osu a potlačuje se dvěma korekčními koeficienty. Dohromady tedy máme alespoň čtyři koeficienty zkreslení obrazu, které jsou důležitou součástí popisu kamery. [18]

Důvodem kalibrace kamery je právě získání vnitřní matice a koeficientů zkreslení obrazu. Existuje více metod jejího uskutečnění. Často používaná je kalibrace pomocí kalibrační desky. Spočívá v umístění kalibračního obrazce na rovinnou plochu a jeho snímání z různých vzdáleností, úhlů a natočení. Tímto obrazcem bývá černobílá šachovnice, ovšem existuje mnoho dalších, vylepšených vzorů. Fyzické rozměry šachovnice jsou vstupem do kalibračního algoritmu, musí být proto známé a přesné. Více snímků znamená přesnější výstup. [18, 16]

2.4 Knihovny pro počítačové vidění

Knihovny počítačového vidění jsou klíčové pro zpracování obrazu a realizace všech cílů práce. Zajišťují jak základní (čtení a zobrazování snímků kamery), tak pokročilé funkce (projekce grafiky do souřadnicových systémů). Existuje poměrně mnoho takových knihoven, někdy jsou navíc založeny i na principech strojového učení. Pro tuto práci jsou však vhodné hlavně knihovny schopné zpracovávat obraz v reálném čase. Je požadováno, aby vybraná knihovna umožňovala kalibraci kamery i odhad její polohy. To nejlépe algoritmem optimalizovaným pro čtvercové značky. Důležitou roli hraje také dostupnost kvalitní dokumentace či dalších zdrojů informací.

2.4.1 OpenCV

Je to open-source knihovna počítačového vidění a strojového učení. Obsahuje více než 2500 optimalizovaných algoritmů. Je organizována do několika modulů, jež zajišťují různé funkcionality. Má velkou uživatelskou základnu, spolu s rozsáhlou online dokumentací i návody. Využívá jí mnoho firem po celém světě, včetně těch velkých. Podporuje programovací jazyky: C++, Python, Java, Matlab a operační systémy: Windows, Linux, Android či Mac OS. Samotná knihovna je napsána v programovacím jazyku C++. Je zaměřena zejména na počítačové vidění v reálném čase. Umožňuje kalibrovat kameru i optimalizovaně řešit PnP problém čtvercových značek. Podporuje práci s ArUco referenčními značkami. Od verze 4.0 umožňuje detekovat i dekódovat QR kódy, avšak pouze jeden v obraze. [19, 16]

2.4.2 BoofCV

Je to open-source knihovna počítačového vidění v reálném čase. Je organizována do několika balíčků, jejichž názvy jsou např. zpracování obrazu, kalibrace a rozpoznávání. Dokumentace je vedená spíše skrze příklady kódu a komentáři v nich. Je napsána v Javě a podporuje pouze tento programovací jazyk. Možnými operačními systémy jsou pak: Windows, Linux a Mac OS. Knihovna umožňuje kalibraci kamery a odhad její polohy vůči referenčním značkám. Dále lze pomocí této knihovny detekovat a dekódovat i vícero QR kódů v obraze. [20]

2.4.3 Zhodnocení

Volbu knihovny počítačového vidění, ovlivnila především nabídka programovacích jazyků. Pro člověka, jež se ještě nepouštěl do programování větších projektů, totiž jazyk Java není vhodnou volbou. Naopak programovací jazyk Python, je jedním z nejjednodušeji uchopitelných a je vhodný i pro úplné začátečníky. Tomu vděčí především jednoduché syntaxi i velké uživatelské základně. Proto je zvolena knihovna počítačového vidění OpenCV, která bude využívána právě skrze programovací jazyk Python na operačním systému Windows.

2.5 Knihovny pro čtení QR kódů

Čtením QR kódů je v této sekci myšlena detekce a následné dekódování. Přestože knihovna OpenCV umožňuje QR kód přečíst, je vhodné poohlédnout se i po dalších knihovnách, přímo určených k tomuto úkonu. Každá implementace čtecí funkcionality je totiž provedená (alespoň mírně) odlišně. Od toho se odvíjejí rozdíly v rychlosti či úspěšnosti za neideálních obrazových podmínek. V úvahu pak přichází pouze knihovny, které lze použít v jazyce Python a nejlépe i v kombinaci s knihovnou OpenCV.

2.5.1 Rozdělení detekce a dekódování

Knihovny pro čtení čárových kódů většinou (pro uživatele) nedělí procesy detekce a dekódování. Pro algoritmus odhadu polohy kamery vůči QR kódům, však bude vhodné tyto úkony rozdělit. Tím je myšlen postup, kdy pouze detekce probíhá v každém snímku, kdežto pokusy o dekódování jen do úspěšného obdržení obsažených informací. Toto rozdělení pak umožní neplýtvat výpočetním výkonem na dekódování v každém snímku. Bude jej místo toho možno využít pro samotný odhad polohy kamery (vůči každému kódu v obraze) a dalších operací. Pomocí výstupů detekce se předpokládá možnost výřezů obrazu jednotlivých detekovaných QR kódů. Ty pak budou individuálně přečteny vybranou knihovnou. Ačkoli proběhne detekce navíc, bude usnadněna menší velikostí obrazu a vysokou pravděpodobností výskytu QR kódu v něm. Se zmíněným postupem souvisí ještě několik problémů, které budou uvedeny v další sekci. Tento postup navíc umožňuje poměrně jednoduchou změnu čtecí knihovny s ohledem na její požadavky.

2.5.2 Vhodní kandidáti

Požadavky, které byly stanoveny na úvodu sekce 2.5 splňují pouze dvě knihovny. Jsou jimi ZXing a Zbar. Obě jsou open-source a schopné číst mnoho typů jak jednorozměrných, tak dvourozměrných čárových kódů, včetně QR kódů. ZXing je napsána v Javě, kdežto ZBar v jazyce C. Vývoj obou knihoven už před lety skončil. Dnes jsou již pouze v tzv. udržovacím módu, který zajišťuje jejich možné použití, případně i opravy chyb. [21, 22]

2.5.3 Porovnání

Jak již bylo zmíněno v úvodu sekce 2.5, některé knihovny zvládají práci s QR kódy lépe než jiné. V rámci dokumentace knihovny BoofCV [20] byla porovnávána funkcionalita čtení QR kódů vůči dalším knihovnám. To za různých obrazových podmínek. Mezi těmito knihovnami byly právě i OpenCV (4.0.1), ZXing a ZBar. Z výsledků tohoto porovnání vyplývá, že ZBar a ZXing jsou podobně úspěšné, kdežto OpenCV si nedokáže dobře poradit s některými ztíženými podmínkami, jako je třeba rozostřený obraz. Rozdíly jsou pak znatelné u rychlosti čtení, to však vzhledem k postupu zmíněném v podsekci 2.5.1, není až tak důležité. Rychlost čtení závisí právě i na obrazových podmínkách. Při tvorbě algoritmu, bude vyzkoušeno čtení QR kódů, již výše zmíněnými třemi knihovnami (OpenCV, ZXing, ZBar). Až poté bude zvolena ta nejvhodnější.

2.6 Využití OpenCV a Pythonu pro řešení cílů

Jako primárního zdroje návodů je vhodné využít těch, jež jsou součástí oficiální online dokumentace OpenCV [16]. Je k dispozici přímo sekce návodů pro jazyk Python. V ní je možné najít velké množství témat, od instalace, přes základní operace, až po strojové učení. Konkrétními přínosnými tématy jsou: kalibrace, základní operace s obrazem, hledání kontur, odhad polohy kamery pomocí PnP, možnosti GUI a vykreslování jednoduché grafiky do obrazu. Dále jsou vhodným zdrojem informací, pro konkrétní dotazy, internetová fóra.

OpenCV pro jazyk Python rozsáhle využívá knihovny NumPy. To je vysoce optimalizovaná knihovna pro numerické operace s podobnou syntaxí, jakou má Matlab. Mnoho obrazových manipulací lze provádět právě skrze tuto knihovnu. [19]

Knihovna OpenCV je dostupná ve dvou verzích. První je klasická a druhá je rozšířená o příspěvkové moduly. Ty ještě nebyly integrovány do hlavní (klasické) verze. Jde především o moduly nové, nedostatečně prověřené, či jinak nepřipravené. Jedním z nich je i modul pro práci s ArUco markery. Konkrétně se jedná o implementaci starší verze ArUco systému, ještě pod licencí BSD (více v podsekci 2.1.1). V dokumentaci jsou i k tomuto modulu návody. Jeden dokonce vcelku podrobně popisuje kroky detekce markerů v OpenCV. Tento návod poslouží jako základ pro tvorbu obdobného algoritmu, avšak pro QR kódy. [16, 23]

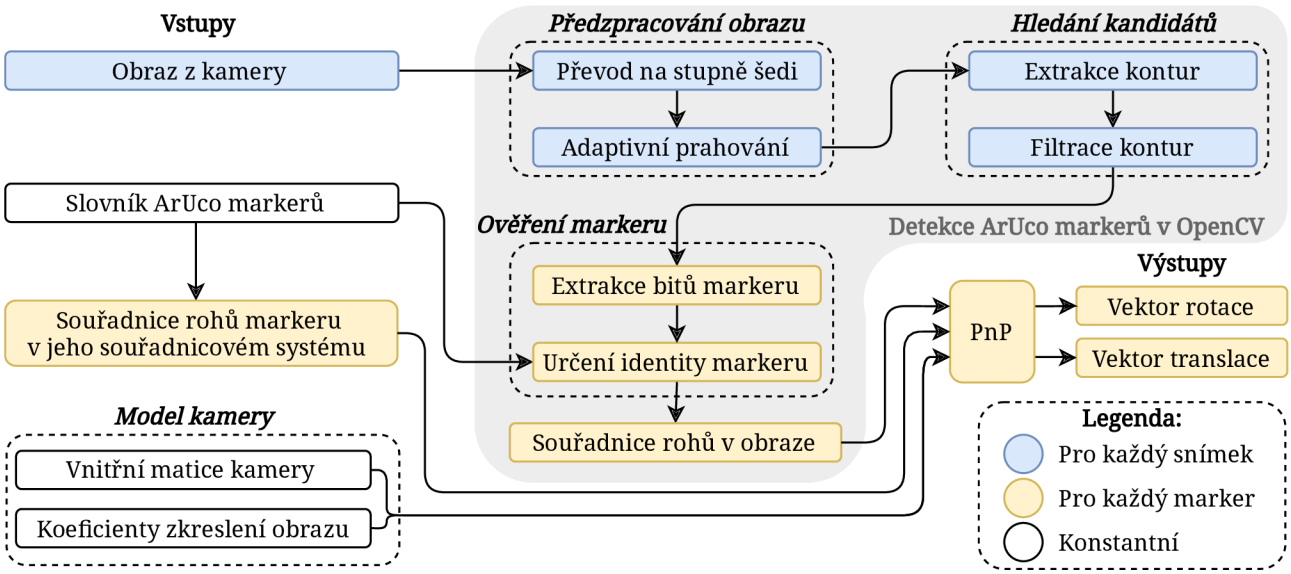
Postup uvedený v podsekci 2.5.1 má za následek ještě dva problémy, jež je nutné vyřešit. Prvním je identifikace jednotlivých rohů QR kódu, což je nutné k zjištění jeho prostorové orientace. Druhým je potřeba sledovače jednotlivých QR kódů, v po sobě jdoucích snímcích tak, aby bylo možné dekódovanou informaci přiřadit tomu správnému. Tento sledovač musí být co nejjednodušší, aby jeho provoz nebyl nakonec výpočetně náročnější, než dekódování v každém snímku z kamery.

2.6.1 Postup detekce ArUco markerů

Tato podsekce vychází z návodu v online dokumentaci OpenCV. Ten se zabývá především postupem detekce ArUco markerů užitým ve stejnojmenném příspěvkovém modulu. Ukázka kódu je v něm sice pro jazyk C++, to však není velkým problémem. Na Obr.2.5 je diagram vytvořený na jeho základě. Detekční proces má dva hlavní kroky. Prvním je hledání kandidátů na markery. Druhým je ověření, zda jde opravdu o marker a (s tím spojené) přiřazení jemu příslušného identifikačního čísla (ID). Funkce zajišťující detekci, vrací seznam detekovaných markerů. Každému detekovanému markeru jsou přiřazeny souřadnice jeho rohů v obraze i jeho ID. V následujících odstavcích bude blíže rozebrán postup hledání kandidátů, jelikož to je krok, který lze uplatnit při vytváření vlastního řešení. [16]

Po načtení obrazu z kamery je obraz předzpracován. To spočívá nejprve v převodu z barevného (BGR) modelu na stupně šedi. Po tomto kroku, je každý obrazový bod popsán pouze hodnotami od 0 (černá) do 255 (bílá). Takový obraz lze snadno binarizovat (prahovat). Pro tento účel je nejdříve stanovena hraniční hodnota. Obrazové body s nižší než hraniční hodnotou jsou pak přepsány na černou a s vyšší na bílou. V nejjednodušším případě se stanovuje jedna hraniční hodnota celý obraz. Avšak pro detekci (nejen) markerů, je vhodnější prahování adaptivní. To je způsob, při kterém se hraniční hodnota upravuje v závislosti na hodnotách okolních obrazových bodů. I z nerovnoměrně osvětlené scény lze pak získat dobré výsledky dalších operací. [16]

Je-li obraz binarizován, lze užít funkce pro hledání kontur *findContours()*. Kontury jsou v daném případě hranice mezi bílými a černými plochami v obraze. Poté je třeba ještě použít funkci *approxPolyDP()* pro aproximaci polygonu z každé kontury. Následuje filtrace kontur. Lze zahodit ty, jež nejsou definovány pouze čtyřmi body, nebo nejsou konvexní (stejně tak i kontury příliš malé, velké, atd.). Výsledkem jsou kontury kandidátů na markery. [16]

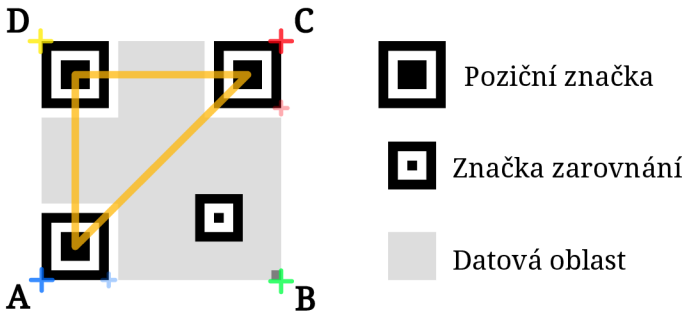


Obrázek 2.5: Postup pro obdržení matice rotace a vektoru translace pomocí ArUco markeru v OpenCV. Vytvořeno podle návodu: Detekce ArUco markerů z [16].

Pro všechny důležité funkce, jsou v návodu uvedeny i výchozí hodnoty jejich parametrů. Ty jsou nastaveny tak, aby funkce pracovaly dobře při běžných podmínkách. Obraz detektorem prochází několikrát (ve výchozím nastavení třikrát), pokaždé s jiným nastavením velikosti okolí adaptivního prahování. Může totiž dojít k tzv. rozbití obrazu, což zapříčiní nefunkčnost dalších operací. Vhodné nastavení tedy závisí na velikosti markeru v obraze. Více průchodů s různými parametry, pak zajišťuje, co největší rozsah detekovatelných vzdáleností. [16]

Elegance příspěvkového ArUco modulu spočívá v jednoduchosti jeho použití. Pokud chce uživatel odhadovat polohu kamery vůči markerům, vystačí si se dvěma funkcemi. Konkrétně jsou to funkce *detectMarkers()* a *estimatePoseSingleMarkers()*. Jednoduché je i případné přenastavení jednotlivých parametrů funkcí užitých v detektoru. To lze provést přepsáním příslušné instance objektu třídy *DetectorParameters*. V návodu je popsán i způsob vytvoření markerů z vybraných slovníků pro tisk, případně si lze vygenerovat slovník vlastní. [16]

2.6.2 Získání souřadnic a identifikace rohů QR kódů



Obrázek 2.6: Zjednodušená struktura QR kódu a značení pro identifikaci rohů. Vytvořeno podle [25] a [24].

Zjednodušená struktura QR kódu je na Obr. 2.6, zároveň znázorňuje i níže popsaný postup identifikace jednotlivých rohů. Detekce bude záviset především na pozičních značkách, což bylo v podstatě již zmíněno v podsekci 2.2.3. Poziční značku si lze představit, jako tři čtverce do sebe

vložené. Pro její detekci je možné využít vzájemných poměrů obsahů ploch značku tvořících. Tento poměr je 7:5:3 a měl by platit i při perspektivních zkresleních. Funkce pro hledání kontur v OpenCV dokáže vracet i jejich hierarchii. Je tedy možné zjistit, která konkrétní kontura je vložená do jiné a naopak. Takovým způsobem lze dohledat ty nejvíce vnořené a od nich postupovat vně. OpenCV umožňuje i jednoduše vypočítat obsah kontur. Tímto způsobem lze najít kandidáty na poziční značky QR kódu. [24, 16]

V tomto odstavci se uvažuje kolmý záběr na QR kód a již získané vnější kontury všech tří pozičních značek. Pro řešení orientace kódu, se požaduje, aby každou tuto konturu reprezentoval jeden bod. Vhodné je těžiště kontury, jehož souřadnice jsou pomocí OpenCV zjistitelné. Na základě vzájemné vzdálenosti těžišť, je možné určit identitu kontury poziční značky, jejíž jeden bod je rohem D. Dále lze, pomocí extrémů kontury trojúhelníka, definovat identity zbylých dvou pozičních značek, pro jednotlivé možné polohy. Obdobným způsobem je zjistitelné, které body jednotlivých kontur, jsou zároveň i rohy samotného QR kódu. Souřadnice čtvrtého rohu (B) je nutné dopočítat pomocí dvou nejbližších pozičních značek (A, C). Pronesou-li se vhodné strany těchto značek přímkami, jejich průsečíkem bude právě čtvrtý roh kódu. Postup využívající extrémů kontur se však zhroutí v mezních polohách, nebo pod úhly, kdy (vlivem perspektivy) již neplatí předpoklad o identitě poziční značky D. [24, 16]

2.6.3 Jednoduchý sledovač objektů v obraze

Důvod nutnosti použití sledovače byla zmíněna již v úvodu sekce 2.6. Takový jednoduchý sledovač pro OpenCV v jazyce Python již existuje [26] a je k němu dostupná i knihovna. Je však vhodné blíže prozkoumat jeho funkcionalitu a metody, kterých využívá.

Zmíněný sledovač, je založen na dopočtu těžišť (dále jen bod) z poskytnutných ohraničujících rámečků sledovaných objektů. Sledováním, je pak myšleno přiřazení stejného ID konkrétnímu bodu v po sobě následujících snímcích. To je zde uskutečňováno pouze na základě euklidovské vzdálenosti bodů, právě v po sobě následujících snímcích. Bodu z aktuálního snímku se přiřadí ID nejbližšího bodu ze snímku předchozího. To samozřejmě přináší i řadu problémů, v podobě možné záměny ID, při některých situacích. Tou je např. přechod jednoho sledovaného objektu přes druhý. Je-li počet bodů v aktuálním snímku vyšší, než v předchozím, přiřadí se nejvzdálenějšímu bodu nové ID. Sledovač umožňuje i definovat počet snímků, pro který se uchová poslední zaznamenaná souřadnice bodu objektu. To kvůli případnému selhání detekce v některých snímcích, či krátkému vystoupení objektu ze záběru. Sledovač je nutno před vstupem do hlavní smyčky programu inicializovat, aby mohl pracovat s daty mezi jednotlivými snímky. [26]

Užitečné je i blíže prozkoumat řešení přiřazování jednotlivých ID. Sledovač využívá pouze jednu třídu, jejíž instance obsahují hlavně řazené slovníky, které v podstatě zastávají funkci paměti. Obsahují informace o ohraničujících rámečcích, souřadnicích těžišť a počtu snímků bez konkrétních ID v obraze. Dále jsou to ještě instance, definující hranici počtu snímků pro zachování ID a další v pořadí, jež bude přiřazeno. Třída má několik metod instance, pro účely: registrace objektu, jejího zrušení, přiřazení ID a aktualizace sledovače. Poslední zmíněnou, je myšleno přepis dosavadních hodnot v řazených slovnících. [26]

2.6.4 Multithreading ke čtení obrazu z webkamery

Čtení obrazu z kamery je běžná operace v oblasti počítačového vidění. Obecně je načtení obrazu z kamery závislé na rychlosti možného datového toku. Pokud tedy má jít o aplikaci s obrazem snímaným v reálném čase, kdy čtení a zpracování probíhá sekvenčně, existuje zbytečná prodleva způsobená právě načítáním obrazu. Tyto procesy však lze (i v Pythonu) rozdělit do separátních vláken procesoru. Hlavní vlákno pak zpracovává obraz, kdežto další se zabývá pouze načtením snímku z kamery. Tím lze docílit menší latence i vyšší snímkovací frekvence v aplikacích. [27]

2.7 Měření přesnosti odhadu polohy

Metodika a provedení měření bude inspirována prací [28]. V ní autor mimo jiné analyzuje přesnost detekce markerů. Konkrétně bude obdobně provedeno měření detekčního rozsahu, jak vzdálenostního, tak úhlového, včetně kritéria 50 % detekovaných snímků. Při měření přesnosti odhadu polohy kamery se pak, stejně jako ve zmíněné práci, budou posuzovat data ze 100 provedených odhadů.

2.8 Některá dosud realizovaná řešení

Existuje již několik obdobných řešení pro využití QR kódů namísto referenčních značek pro odhad polohy kamery. Tato práce jimi nebyla ovlivněna, kvůli snaze vytvořit další řešení. Nicméně je vhodné se o nich krátce zmínit.

Autoři v [29] charakterizují výhody QR kódu oproti referenčním značkám a představují systém jejich detekce. Zveřejňují i postup pro využití v AR. Jejich řešení je založeno na knihovně ARTooolKit. Popisují, že tato knihovna dokáže vracet 3D pozice středů pozičních značek. Tímto způsobem jsou schopni řešit identitu rohů i pod perspektivním zkreslením, obdobně jako je popsáno v podsekci 2.6.2. Není však zmíněno, zda jejich systém dokáže pracovat i s více QR kódy v obraze.

V článku [30] autoři taktéž shrnují výhody QR kódu vůči referenčním značkám a porovnávají i několik přístupů k jejich detekci. Dále standardní QR kód rozšiřují o černý rámeček ve stylu markerů. S takto vylepšeným kódem pak dále pracují.

Práce [31] se zabývá využitím QR kódu pro lokalizaci mobilního robota. Autoři pracují s knihovnou OpenCV a ZBar. Předpokládají umístění kódů na strop, tudíž neřeší problémy, jako je identifikace rohů kódu pod perspektivním zkreslením.

Článek [32] se zabývá využitím QR kódů k chirurgické navigaci. Představuje pokročilý systém s vysokou přesností odhadované polohy. To při maximálně pěti snímcích za sekundu. Autoři však upozorňují na značný prostor pro optimalizaci algoritmu.

Dále ještě existuje i mnoho hybridních systémů, jež kombinují QR kódy s různými referenčními značkami. QR kód u takových systémů zastává funkci informačního zdroje (většinou odkaz na internetový server). Referenční značka pak slouží, pro odhad polohy kamery a následně jako podklad pro vykreslení grafiky. Tato práce se však zabývá využitím nijak neupravených standardních QR kódů.

3 Upřesnění cílů

Na základě rešerše (kapitola 2) a konzultace s vedoucím práce byly upřesněny cíle práce následovně:

- Vybraným dvourozměrným čárovým kódem je QR kód.
- Zvoleným programovacím jazykem je Python společně s knihovnou počítačového vidění OpenCV.
- Algoritmus pro odhad polohy kamery vůči QR kódům zahrňte do knihovny, jež bude uživatelsky přívětivá, podobně jako příspěvkový modul ArUco knihovny OpenCV.
- Přesnost odhadu polohy kamery vytvořeného řešení s QR kódy, bude porovnávána vůči řešení s ArUco markery ze stejnojmenného příspěvkového modulu knihovny OpenCV.

Porovnání vůči jiným řešením nebude provedeno, jelikož obdobné řešení využitelné v jazyce Python není k dispozici a např. s BoofCV by bylo komplikované. To vzhledem ke skutečnosti, že vyhodnocování dat bude taktéž prováděno v Pythonu.

4 Postup řešení a výsledky

Prvním krokem v postupu je kalibrace používané kamery. Poté je vytvořena již zmíněná (v kapitole 3) knihovna, načež je toto řešení měřením porovnáno oproti ArUco markerům ze stejnojmenného příspěvkového modulu knihovny OpenCV. Nakonec je vytvořena ukázková aplikace, využívající vytvořeného řešení. QR kódy použité v této kapitole jsou generovány pomocí [33].

4.1 Použitý hardware a software

Webkamera používaná v této práci nese název Microsoft LifeCam Studio. Komunikuje s počítačem přes rozhraní USB 2.0, disponuje automatickým ostřením a dokáže poskytovat záznam v rozlišení 1280×720 při až 30 snímcích za sekundu. [34]

Počítač běží na operačním systému Windows 10 a jeho procesorem je Intel Core i7-6700HQ. Dále je používán Python verze 3.8 a následující knihovny:

- OpenCV verze 4.2.0.34 (podsekce 2.4.1, sekce 2.5 a 2.6),
- NumPy verze 1.18.5 (sekce 2.6),
- Centroid tracker verze 0.0.9 (podsekce 2.6.3),
- PyZBar verze 0.1.8 (podsekce 2.5.2).

Původně měla být ve výčtu i knihovna pro dékódování čárových kódů ZXing (podsekce 2.5.2), tu se však nepodařilo úspěšně nainstalovat. Při zběžných pokusech o dekódování různých velkostí a verzí QR kódů, knihovna OpenCV, oproti ZBar, pracovala buďto déle, či jejím použitím k úspěšnému dekódování ani nedošlo. Při tvorbě vlastní knihovny bude tedy pro tuto funkcionalitu využita pouze knihovna ZBar.

4.2 Kalibrace používané webkamery

Kalibrace byla provedena dle návodu, v oficiální dokumentaci OpenCV [16], pro jazyk Python. O kalibraci kamery je pojednáno v podsekci 2.3.3. Jako kalibrační vzor byla použita šachovnice 10×3, vytištěná na kancelářském papíru formátu A3. Ten byl připevněn lepící páskou na rovnou desku. Strana jednoho pole měla rozměr 35 mm. Automatické ostření kamery bylo vypnuto a manuálně nastaveno. Zřejmě v důsledku toho, že ovladače této kamery pro Windows 10 nejsou zcela ideální, případně možnou závadou, je hloubka ostrosti přibližně 85-125 mm. To není ideální, zrovna ale nebyla jiná webkamera k dispozici. Pro porovnání a aplikaci bude i takováto kamera stačit, avšak bude nutné brát na tuto skutečnost ohledy. Pro kalibraci bylo pořízeno 110 snímků šachovnice. Výsledkem je vnitřní matice kamery a koeficienty zkreslení obrazu, důležité pro odhad polohy kamery viz. sekce 2.3.

4.3 Tvorba knihovny

Jak již bylo zmíněno v kapitole 3, knihovna by měla být vytvořena s ohledem na jednoduchost jejího použití. Proč je příspěvkový modul ArUco v OpenCV dobrým příkladem, je vysvětleno v podsekci 2.6.1. Po jeho vzoru tedy bude několik hlavních funkcí zajišťovat veškeré důležité procesy. Dále je potřeba zmínit myšlenku o rozdělení procesů detekce a dekódování (rozvedena v podsekci 2.5.1). V důsledku toho, je nutné použit sledovač objektů, jehož funkce je vysvětlena v podsekci 2.6.3. Poslední zmíněná podsekce popisuje i způsob přiřazování jednotlivých ID objektům. Toho bude využito při řešení uchovávání a přiřazování již dekódovaného obsahu QR kódům. Výhodou QR kódu vůči referenčním značkám je, že není nutná jejich dopředná registrace. Tato výhoda bude ještě rozšířena, o možnost obsáhnutí informace týkající se reálných rozměrů, přímo v jeho datech. Tato informace bude očekávána na konci řetězce, oddělena mezerou, v definovaném formátu (např. S30). Vzhledem ke kalibraci kamery v jednotkách mm, musí být obsažená hodnota uvedena taktéž v mm.

Základ knihovny tvoří čtyři třídy, využívané v hlavních funkcích, jsou jimi:

- *Memory*,
- *CodeCandidate*,
- *Code*,
- *DetectorParameters*.

Třída *Memory* řeší již výše zmíněné uchování dekódovaného obsahu. Tato třída je myšlenkou (i provedením) podobná třídě zajišťující chod sledovače. Obsahuje tři instance slovníků, jejichž klíči je ID QR kódu ze sledovače. Čtvrtou instancí je počet snímků, po který se data ve slovnících uchovávají, v případě nepřítomnosti konkrétního ID ve snímku. Jeden slovník uchovává informaci o velikosti kódu (více k tomuto níže), další zbytek dekódovaných dat a poslední slouží právě k počítání snímků, ve kterých se QR kód s příslušným ID neobjevil. Třída má několik metod instancí všechny jako vstup požadují ID. Tyto metody zajišťují navýšení, či vynulování počítadla snímků do smazání a právě i samotné odstranění dat spjatých s konkrétním ID.

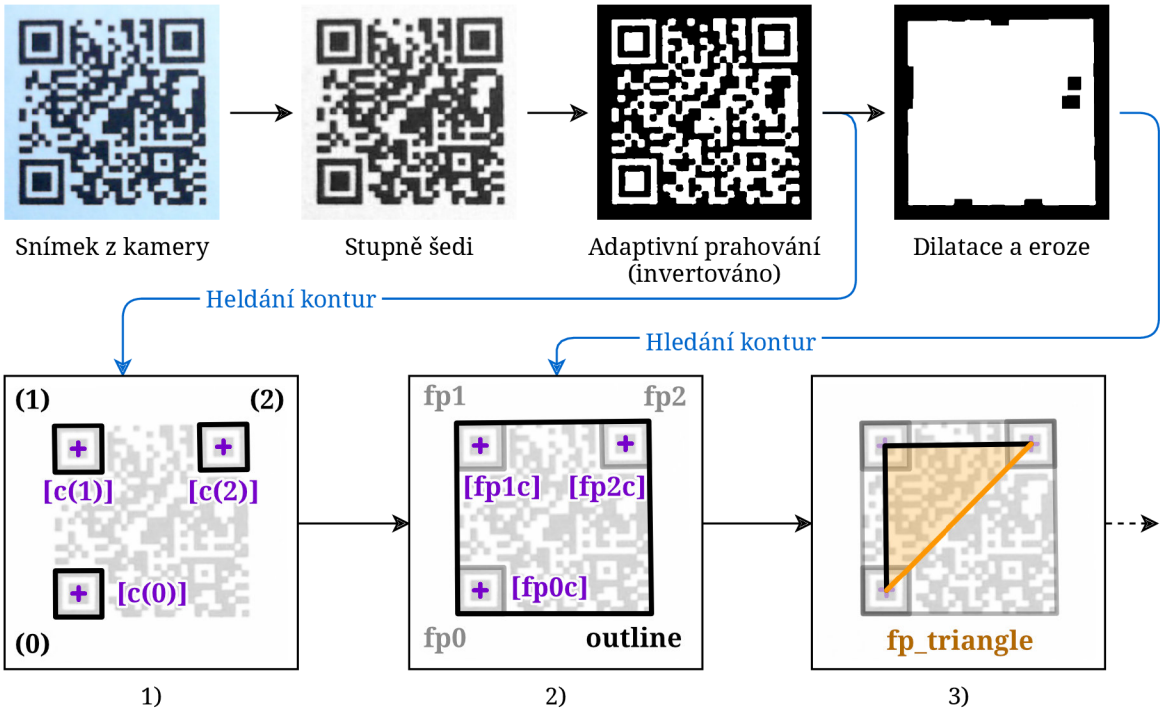
Třída *CodeCandidate* je důležitá v procesu detekce QR kódů. Její instance slouží k uložení souřadnic bodů, kontur, či dalších dat. Většina z nich je používána při identifikaci jednotlivých rohů QR kódu. Instance jsou předem definovány jako None.

Třída *Code* je uzpůsobena k tomu držet všechna důležitá data, která by uživatel očekával a některá navíc. Jsou jimi: ID (přiřazené sledovačem), datový obsah, rozměry, vektor rotace i translace, souřadnice jednotlivých rohů, rozměr relativní k velikosti QR kódu v obraze a ohraničující rámeček. Po úspěšné detekci ještě nemusí být určeny instance rozměru a datového obsahu, jelikož ty závisí na úspěšném dekódování. Instance vektorů rotace a translace jsou určeny až po výstupu z funkce pro odhad polohy. To ale taktéž závisí na jednom úspěšném dekódování. Neurčené instance mají hodnotu None.

Poslední třídou je *DetectorParameters*, pomocí níž lze přenastavovat parametry jednotlivých procesů detekce QR kódů. Má určitá výchozí nastavení.

4.3.1 Algoritmus detekce a identifikace rohů QR kódů

Knihovna OpenCV je importována pod zkratkou *cv2*. Algoritmus detekce QR kódů stojí na podobné kostře jako postup v podsekci 2.6.1. Avšak specifická detekce samotných kódů a identifikace rohů je inspirována postupy zmíněnými v podsekci 2.6.2 a doplněná o vlastní úvahy. Kontury jsou detekovány funkcí *cv2.findContours()* s aproximační metodou pro šetření paměti. Detekované kontury vždy prochází ještě další aproximační funkcí *cv2.approxPolyDP()* s nastavitelnými parametry. Dále jsou pak hodnoceny jen kontury definované pouze čtyřmi body.

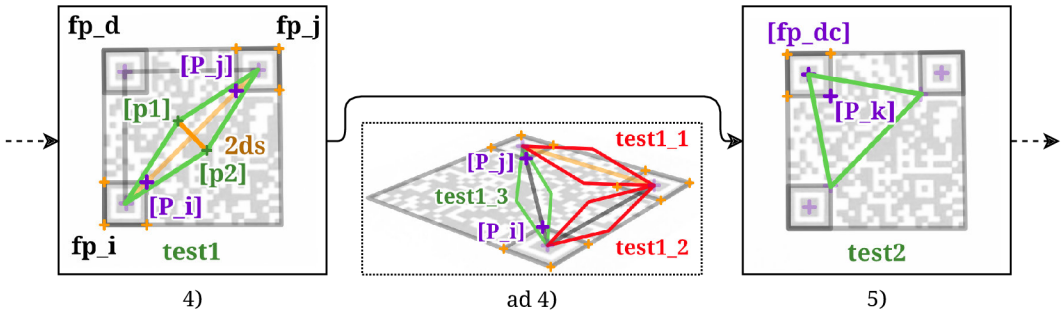


Obrázek 4.1: Algoritmus detekce a identifikace rohů QR kódu, část první. Inspirováno postupy v [16, 24].

Na Obr. 4.1 je znázorněn postup při detekci QR kódů a počáteční fáze identifikace rohů. Nejprve je přečten snímek z kamery a převeden na stupně šedi. Poté je na obraz aplikováno Gaussovské rozostření (není na obrázku) pro zbavení se šumu. Právě šum by způsobil výpočetně náročnější hledání kontur. Následně je obraz adaptivně prahován, přičemž je tato operace nastavena tak, aby převrátila hodnoty barev. To kvůli intuitivnější hierarchii kontur po jejich nalezení. Dalším krokem je právě hledání. Na základě hierarchie kontur a jejich filtrace je možné separovat vnější konturu všech pozičních značek v obraze. Jedním z hlavních použitých kritérií pro poziční značku, je poměr obsahů kontur 3:5:7. Ten je rozšířen o nastavitelnou hodnotu tolerance. Po získání takových kontur, jsou vypočteny souřadnice jejich těžišť (*krok 1*).

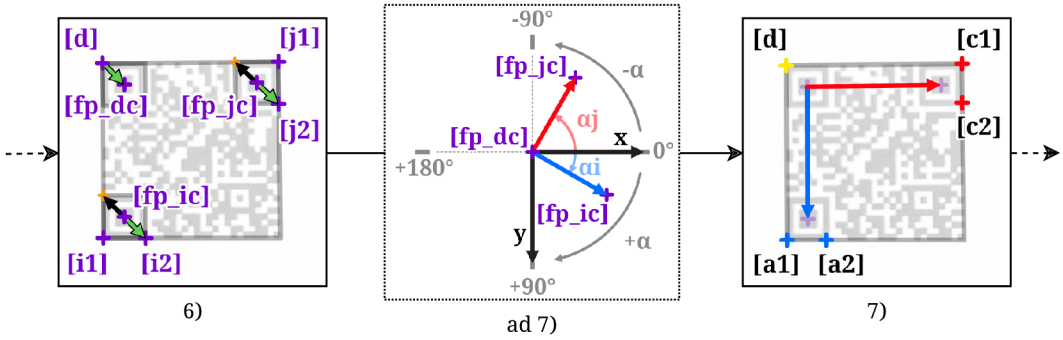
Na již prahovaný obraz se použijí morfologické operace dilatace a eroze, opět s nastavitelnými parametry. Tyto operace musí, pro úspěšnou detekci, zapříčinit slití QR kódu v jeden čtyřúhelník. V takovém obraze jsou pak hledány další kontury. Ty které projdou základním filtrem, jsou testovány pomocí funkce *cv2.pointPolygonTest()*. Ta zjišťuje, zda se bod, o určitých souřadnicích, nachází uvnitř konkrétní kontury. Pokud kontura ohraničuje právě tři těžiště pozičních značek, je vytvořen objekt třídy CodeCandidate. Vnější kontury těchto objektů ještě prochází filtrem, ošetřujícím problém, více do sebe vnořených kontur, se stejnými pozičními značkami. Tímto způsobem je vyřešena detekce více QR kódů v obraze (*krok 2*). Dále je řešen problém identifikace rohů. Pomocí těžišť přiřazených pozičních značek je vytvořena trojúhelníková kontura a je vypočítán její obsah. Tato hodnota je pak upravena a používána dále v procesu identifikace, jelikož je vztažená k velikosti QR kódu v obraze. Zároveň je i uložena do instance objektu třídy CodeCandidate (*krok 3*).

Obr. 4.2 znázorňuje další postup pro identifikaci jednotlivých rohů QR kódu. Tato část algoritmu řeší především identifikaci poziční značky D (stejné značení jako na Obr. 2.6). To i pod úhly, kdy již neplatí předpoklad, že body spojující nejdelší stranu trojúhelníka patří pozičním značkám A a C. Místo toho je uvažováno, že spojnice těžišť pozičních značek A a C, prochází vždy i rohy těchto značek.



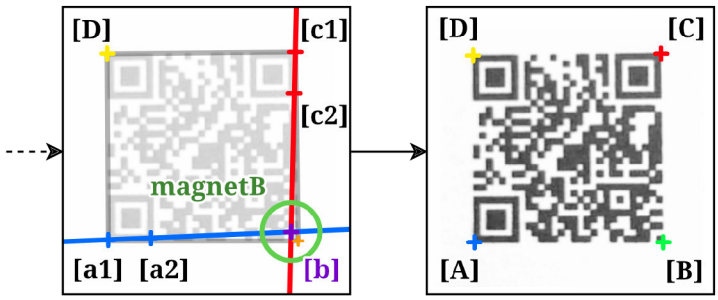
Obrázek 4.2: Algoritmus detekce a identifikace rohů QR kódu, část druhá.

Nejprve jsou spočítány vzdálenosti mezi jednotlivými těžišti. Již zmíněná hodnota vztažená k velikosti QR kódu v obraze, je dále vynásobena nastavitelným parametrem. Výsledek je nazván testovací vzdáleností. Poté je vytvořen z dvojice bodů s největší vzájemnou vzdáleností vektor, pomocí kterého je nalezena souřadnice bodu v jeho polovině. Od tohoto bodu, jsou, kolmo na vektor, odsazeny další dva body, právě o testovací vzdálenost. Dále je těmito dvěma body, přes těžiště tvořící vektor, vedena testovací kontura. Probíhá obdobný test, jako v *kroku 2*. Testovány jsou všechny body tvořící kontury příslušných dvou pozičních značek. Uvnitř testovací kontury se musí nacházet právě jeden bod od každé testované poziční značky. Pokud uvažovaná dvojice pozičních značek tímto testem neprojde, testuje se další. Nyní je identifikována poziční značka D a jsou známy indexy vnitřních bodů, v popisu kontur dalších dvou (*krok 4*). Dále je vytvořena poslední testovací kontura, pro zjištění indexu vnitřního bodu kontury poziční značky D (*krok 5*). Body popisující konturu jsou ve sledu jejího spojení. Pokud je popsána čtyřmi body, lze jednoduše zjistit bod na opačné straně. Proto jsou souřadnice tří ze čtyř vnějších rohů QR kódu známy. Identita je však určena pouze u bodu D.



Obrázek 4.3: Algoritmus detekce a identifikace rohů QR kódu, část třetí.

Na Obr. 4.3 je znázorněn další postup při identifikaci rohů QR kódu. Je nutné určit souřadnice bodů pro pozdější odhad souřadnic rohu B. Pro tyto pomocné body platí předpoklad, znázorněný v *kroku 6*, o směrech vektorů. Tomuto testu podléhají jen dva protilehlé body kontur neurčených pozičních značek. Není proto těžké jej realizovat využitím indexů. K tomuto testu, ale i dalšímu kroku je využívána funkce z matematického modulu *math.atan2()*, jež umožňuje jednoznačně určit směr vektoru. Její výstup v souřadnicovém systému obrazu, po převedení na stupně, je znázorněn v dodatku ke *kroku 7*. Pro určení identity pozičních značek A a C jsou vytvořeny dva vektory. Jsou orientované z těžiště poziční značky D do zbylých dvou těžišť. Pro identitu pozičních značek, v souřadném systému obrazu, pak vždy platí, že úhel vektoru A předbíhá úhel vektoru C (*ad krok 7*). Pro využití této úvahy, bylo zapotřebí ošetřit i situace mezi hranicí znaménka výstupu. Nyní jsou tedy známy souřadnice rohů A, C, D a souřadnice pomocných bodů pro odhad rohu B (*krok 7*).



Obrázek 4.4: Algoritmus detekce a identifikace rohů QR kódu, část čtvrtá. Inspirováno postupem v [24].

Na Obr. 4.4 je znázorněna poslední část algoritmu pro identifikaci rohů QR kódu. Je využito vztahu z [35] pro výpočet souřadnice průsečíku dvou přímek, kde každá je definována dvěma body. Souřadnice bodů kontury však nejsou zcela přesné a odhad souřadnic rohu se tímto způsobem, v každém snímku, skutečnému rohu kódu pouze přibližuje. Je-li však zachována dostatečně velká tichá zóna v závislosti na parametrech detektoru, můžeme s výhodou užít vnější kontury QR kódu z *kroku 2*. Tento postup je v nastavení parametrů detektoru možno zakázat, je-li však povolen, funguje následovně.

Pokud se v oblasti kolem odhadnutých souřadnic nachází bod tvořící vnější konturu, stane se tento bod i samotným rohem *B*. Oblast je vymezená poloměrem, jehož velikost je nastavitelný parametr násobený hodnotou vztaženou k velikosti QR kódu v obraze (*krok 8*). Tato metoda je nejpřesnější, má-li QR kód v rohu alespoň jedno černé pole. To je většinou možné, vzhledem k různým použitelným vzorům masky datové oblasti. Po celém detekčním procesu je vytvořen objekt třídy *Code* a do instancí jsou uloženy souřadnice jednotlivých rohů i hodnota vztažená k velikosti QR kódu v obraze (*krok 9*).

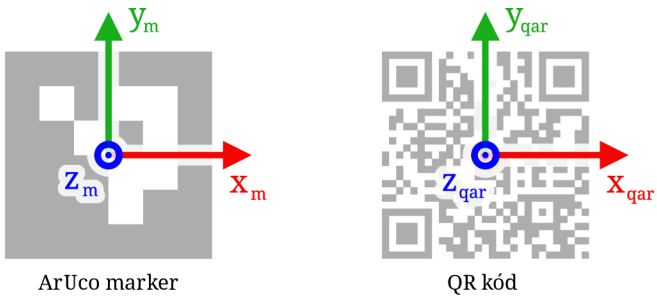
4.3.2 Přiřazení obsahu QR kódů

Po detekci a identifikaci jednotlivých rohů QR kódu, je možné řešit jeho obsah. V této fázi se již pracuje s objekty třídy *Code*. Nejprve je nutné zajistit funkci sledovače. Ten pracuje s ohraničujícím rámečkem objektu, což je obdélník, zarovnaný v souřadném systému obrazu, který ohraničuje celý QR kód užitím souřadnic jeho rohů. Tento rámeček je vstupem do metody instance sledovače. Z výstupu je pak možno přiřadit ID konkrétnímu objektu *Code* a uložit jej i do příslušné instance. Ukládá se i onen ohraničující rámeček. Existují-li pro daný objekt již uložená data, jsou mu na základě ID přiřazena. Neexistují-li pro daný objekt v objektu třídy *Memory* uložená data, je přistoupeno k pokusu o dekódování. Ohraničující rámeček se rozšíří po všech čtyřech stranách o hodnotu vztaženou k velikosti kódu v obraze. To aby bylo zajištěno, že ve výřezu opravdu bude celý QR kód i s dostatečnou tichou zónou. Využitím knihovny ZBar, je pak realizován pokus o dekódování, pouze tohoto výřezu. Je-li úspěšný, vytáhne se z uloženého řetězce informace o rozměru QR kódu a uloží se do příslušných instancí objektu třídy *Code* a *Memory*. Stejně tak i zbytek dekódovaného řetězce. V případě, že není nalezena informace o rozměrech kódu, je jí přiřazena hodnota nula. To pak odlišuje QR kód ještě nedekódovaný (příslušná instance je None) od již dekódovaného.

4.3.3 Odhad polohy QR kódů

Funkce pro odhad polohy přijímá pouze objekty třídy *Code*. V jejích parametrech lze povolit i odhad polohy pro objekty s informací o rozměrech rovnou nule. Ta bude při odhadu nahrazena číslem jedna. Takový odhad je pak jednotkově vztažený k délce strany QR kódu. Jde jej stále

užít např. pro vykreslování grafiky v AR. Pro odhad polohy je použita funkce *cv2.solvePnP()* s metodou vhodnou pro čtvercové referenční značky. Konkrétní metoda předpokládá čtyři body objektu v jedné rovině a souřadnicový systém pak situuje do jeho středu. Této funkci je potřeba dodat body objektu, čili souřadnice rohů v souřadnicovém systému QR kódu, přičemž je užita informace o rozměrech. Dále souřadnice těch stejných bodů (rohů) v souřadnicovém systému kamery (obrazu). A nakonec ještě vnitřní matici kamery a koeficienty zkreslení obrazu získané kalibrací. Body jsou seřazeny tak, aby výsledný souřadnicový systém QR kódu měl stejnou orientaci, jako mají ArUco markery (viz. Obr. 4.5). Více je k odhadu polohy kamery napsáno v sekci 2.3.



Obrázek 4.5: Orientace souřadnicových systémů ArUco markeru [4] a QR kódu.

4.3.4 Použití

Knihovna nese název *QR code augumented reality* a v ukázkovém kódu bude importována jako *qar*. Knihovna sledovače objektů poskytuje třídu *CentroidTracker*. Funkce zajišťující hlavní operace knihovny jsou pak následující:

a) *qar.init()*,

b) *qar.detectCodes()*,

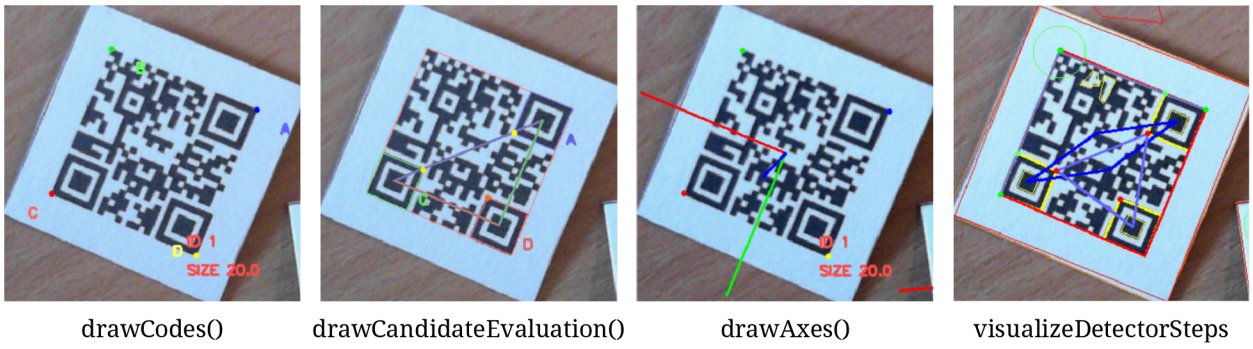
c) *qar.estimatePose()*.

Funkce **a)** inicializuje sledovač objektů a paměť pro detekovaný obsah QR kódů. Vytvoří tedy objekty tříd *CentroidTracker* a *Memory*. Parametrem této funkce může být počet snímků (výchozí hodnota je 50), pro který se zachovává dané ID a data s ním spojená, v případě nepřítomnosti daného ID v obraze. Je nutné ji vyvolat ještě před vstupem do hlavní programové smyčky. V té jsou postupně vyvolávány funkce b) a následně c), pro každý snímek.

Funkce **b)** provádí procesy popsané v podsekcích 4.3.1 a 4.3.2. Základně má tři vstupy, v případě úpravy parametrů detektoru čtyři, a jeden výstup. Vstupy jsou snímek z kamery, objekty sledovače a paměti, případně pak i objekt parametrů detektoru. Výstupem je seznam objektů třídy *Code*, tedy detekované QR kódy. Nabízí i možnost vracení objektů třídy *CodeCandidate*, vizualizace kroků detektoru a detekce světlých QR kódů na tmavém podkladě. První dvě možnosti jsou přítomny pro jednodušší nastavování parametrů, případně i opravy chyb.

Poslední hlavní funkcí je **c)**. Proces který vykonává, je popsán v podsekci 4.3.3. Vstupy do této funkce jsou seznam objektů třídy *Code*, vnitřní matice kamery a koeficienty zkreslení obrazu, případně i parametr umožňující odhadovat polohu QR kódů bez definovaných rozměrů. Tato funkce nemá výstup, ukládá totiž své výsledky do příslušných instancí objektu třídy *Code*. Jsou jimi vektory rotace a translace.

Knihovna poskytuje ještě další dodatečné funkce, pro znázorňování výsledků detekce, či odhadu polohy. Jsou znázorněny na Obr. 4.6. Funkce *qar.drawCodes()* vykresluje do obrazu rohy QR kódů, jejich ID a informaci o rozměru. Další funkce *qar.drawCandidateEvaluation()* vykresluje část detekčního procesu, pomocí seznamu objektů třídy *CodeCandidate*. Poslední znázorňovací funkcí je *qar.drawAxes()*, jež se stará o vizualizaci souřadnicových systémů jednotlivých QR kódů, stejně tak i jejich ID, informací o rozměru a rozích. Obraz se zřejmě nejužitečnější vizualizací pro nastavení parametrů detektoru, je možné dostat z funkce *qar.detectCodes()*. To přepnutím instance *visualizeDetectorSteps* objektu třídy *qar.DetectorParameters* na hodnotu True. Zmíněná funkce pak vrací i obraz s vizualizací značné části detekčního procesu. Všechny zmíněné lze i libovolně kombinovat. Těmto funkcím je vhodné předat nikoli základní snímek z kamery, ale jeho kopii. Předchází se tak možným problémům. To neplatí při použití vizualizace skrze funkci *qar.detectCodes()*, ta si kopii vytváří sama.



Obrázek 4.6: Možnosti vizualizace výsledků detekce.

Chce-li uživatel změnit nastavení různých parametrů detektoru, poslouží mu k tomu již několikrát zmíněný objekt třídy *qar.DetectorParameters*. Ten je třeba definovat před vstupem do hlavní smyčky programu a, po definici přepsat zvolené instance, jež jsou parametry jednotlivých kroků detektoru. Takto nastavené parametry jsou pak při detekci použity, vloží-li se jako přídavný parametr funkci *qar.detectCodes()*.

Knihovna, za dobrého osvětlení, pracuje na dané hardware konfiguraci až při 19 snímcích za sekundu, s vykreslováním funkcí *qar.drawAxes()*. Dokáže detekovat a odhadovat polohu vícero QR kódů v záběru, což je ukázáno i na Obr. 4.7.

4.3.5 Omezení a mozná vylepšení

Jelikož je tato knihovna prvním větším projektem autora, dá se očekávat, že některé programovací postupy nejsou zvládnuty zcela správně. Testování vytvořeného řešení bylo ovlivněno použitou kamerou, jejíž fungování není optimální (více v sekci 4.2). Velkou slabinou představeného řešení je využití dodatečné knihovny pro čtení QR kódu. Předvedeným způsobem probíhá, při jejím použití, více opětovného zpracování obrazu. Tento proces je však proveden pouze několikrát (ideálně jednou) pro každý QR kód. Navíc i s menší velikostí obrazu (výřez). Další velkou slabinou je použití jednoduchého sledovače objektů, jež pracuje s poměrně vysokým rizikem záměny ID. Úspěšnost detekce je závislá na nepoškozených pozičních značkách a na zachování tiché zóny. Zároveň je nutné pečlivě nastavit parametry detektoru. Nedostatky jsou i v odhadu souřadnic rohu B, stejně jako v pixelové nepřesnosti souřadnic všech použitých bodů. Postup vůbec nepoužívá značky zarovnání. Celá knihovna by pak zřejmě pracovala lépe, po přepsání do jazyka C++.

Obrázek 4.7: Detekce a vizualizace více QR kódů v obraze.

4.4 Porovnání přesnosti

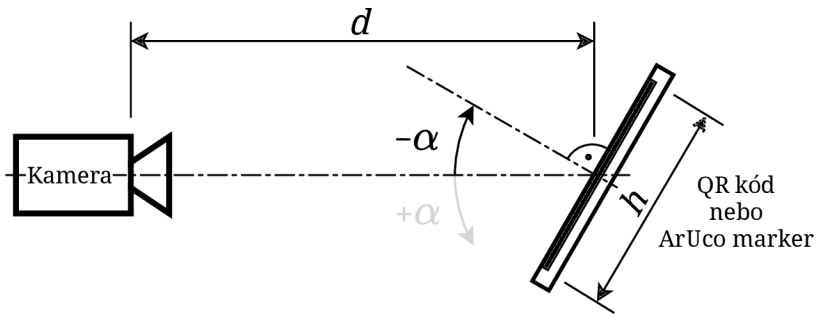
V sekci 2.7 již bylo měření krátce nastíněno. Dále byl cíl s ním spojený upřesněn v kapitole 3. Problémy s používanou kamerou byly nastíněny v sekci 4.2. Půjde tedy o porovnání přesnosti odhadu polohy kamery řešení pro QR kódy, představeného v sekci 4.3, vůči řešení s ArUco markery stejnojmenného příspěvkového modulu knihovny, pro strojové vidění, OpenCV. Budou provedena měření:

1) vzdálenostního rozsahu detekce,

2) úhlového rozsahu detekce,

3) přesnosti odhadu polohy kamery.

Toto seřazení má svůj význam. Všechna měření využívají uspořádání měřicí sestavy na Obr. 4.8, kde d je vzdálenost mezi kamerou a QR kódem/markerem, h je délka jeho strany a α je úhel mezi optickou osou kamery a osou kolmou na plochu QR kódu/markeru. Konkrétní měřené polohy jsou stanoveny na základě měření předchozího. Krom toho, bude zužován i výběr velikostí QR kódu/markeru.



Obrázek 4.8: Uspořádání měřicí soustavy.

Pro měření byl vybrán integrovaný slovník ArUco markerů 5×5_50, kde první čísla vyjadřují velikost pole bitů a poslední číslo počet markerů ve slovníku. Tento slovník byl vybrán jako střední cesta z nabízených slovníků v příspěvkovém modulu ArUco. U QR kódů jsou zvoleny velikostní verze 3 a 4 , protože je jejich kapacita již dostatečná pro aplikace odkazující na internetové servery s dalšími daty. Dále byly pro měření vybrány hlavně rozměry QR kódů/markerů 30, 40 a 50 mm.

4.4.1 Postup měření a zpracování výsledků

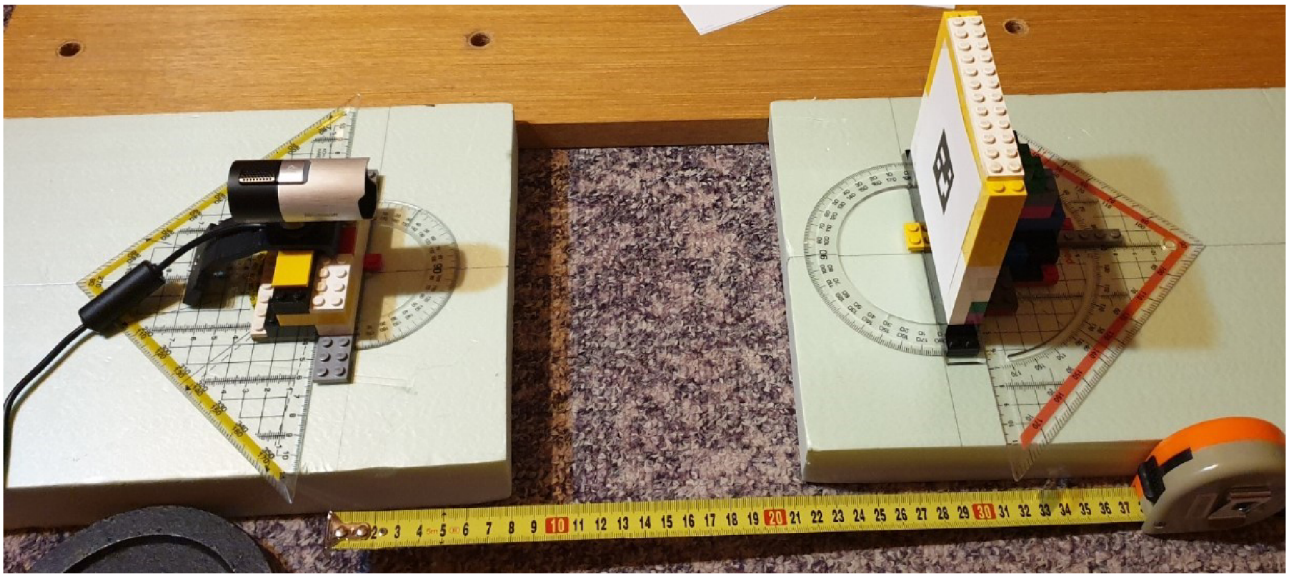
Při měření vzdálenostního a úhlového rozsahu detekce, je buďto manipulováno se vzdáleností d, nebo úhlem α a je posuzováno detekování QR kódu/markeru ve 100 po sobě následujících snímcích. Proběhne-li alespoň 50 detekcí z uvedeného počtu úspěšně, je pozice považována detekovatelnou. Uvedeným měřením je pak hledána hraniční poloha, při které ještě toto kritérium platí. Je měřeno více velikostí h QR kódů/markerů.

Měření přesnosti odhadu polohy je naopak provedeno v předem určených polohách, to znamená známých d a α. Je provedeno 100 odhadů polohy pro každou měřenou pozici. Vektor translace se pro další vyhodnocení nemusí nijak upravovat. Vektor rotace je převeden na matici a z ní je natočení vyjádřeno pomocí Eulerových úhlů. Úhly jsou převedeny z radiánů na stupně. Všechny výsledky jsou pak zpracovány, aby vyjadřovaly deviaci od skutečných hodnot. Měření je prováděno pouze pro jeden rozměr h QR kódu/markeru.

Aby byly výsledky co nejvíce porovnatelné, parametry společných kroků detektoru jsou nastaveny obdobně.

4.4.2 Provedení meřicí soustavy

Jelikož měření probíhalo v době celostátní karantény, měřicí sestava byla realizována velmi improvizovaně (viz. Obr. 4.9). Sestává z desky a dvou bloků, z nichž jeden je při měření posouván (ten nesoucí QR kóde/marker). Držáky jak QR kódů/markerů, tak kamery jsou zhotoveny pomocí stavebnice LEGO a oboustrannou lepící páskou připevněny ke spojené dvojici pravoúhlého pravítka a úhloměru. To pomohlo hlavně k zajištění souososti, ale i stejné výšky. Papír s vytisknutým QR kódem/markerem, je připevněn na příslušný držák tak, aby optická osa kamery byla, při nulovém úhlu α, k němu kolmá. Zároveň i aby protínala střed měřeného QR kódu/markeru. Vzdálenosti jsou měřeny pomocí svinovacího metru a úhly podkladným úhloměrem.



Obrázek 4.9: Realizace měřicí soustavy.

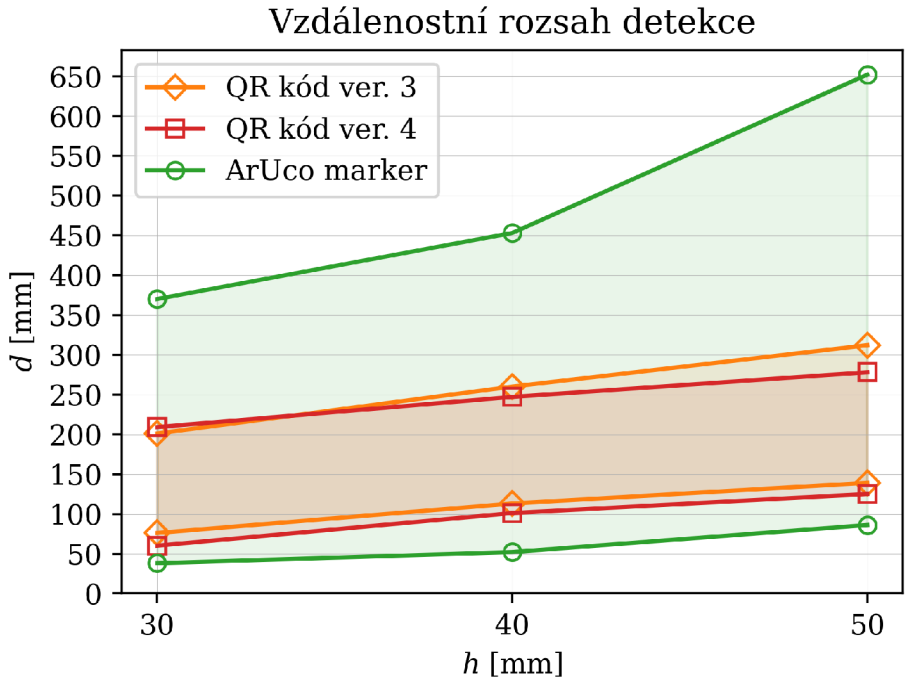
Všechna měření byla pod symetrickým umělým osvětlením prováděna v zatemněné místnosti. Jelikož je optika webkamery (tedy i její souřadnicový systém) zapuštěna do jejího těla, bylo nejprve, pomocí několika odhadů polohy markeru, určeno nutné předsazení. O to byla poté kamera na svém držáku posunuta. Přesnost podélných posuvů d je odhadována v řádech milimetrů a přesnost úhlů α v řádech stupňů.

4.4.3 Měření vzdálenostního rozsahu detekce

Základ postupu uvedený v podsekci 4.4.1. Při měření vzdálenostního rozsahu detekce, je $\alpha = 0°$ a mění se pouze d. Nejprve se zhruba hledá hraniční poloha, která je poté dále zpřesňována. Takto jsou nalezeny minimální d_{min} a maximální d_{max} detekovatelné vzdálenosti. Tímto způsobem byla naměřena data v Tab. 4.1. Pro lepší představu byl ještě vytvořen Obr. 4.10, znázorňující vzdálenostní rozsah detekce měřených QR kódů/markerů v závislosti na jejich rozměrech h.

Tabulka 4.1: Výsledky měření vzdálenostního rozsahu detekce.

h [mm]	Měřeno	d_{min} [mm]	d_{max} [mm]
10	ArUco ID 1	31	263
30	QR kód ver. 3	76	201
30	QR kód ver. 4	60	209
30	ArUco ID 3	38	370
40	QR kód ver. 3	113	260
40	QR kód ver. 4	101	247
40	ArUco ID 4	52	453
50	QR kód ver. 3	139	312
50	QR kód ver. 4	125	278
50	ArUco ID 5	86	652



Obrázek 4.10: Graf z naměřených dat pro lepší znázornění vzdálenostního rozsahu detekce. Body spojeny kvůli přehlednosti.

Při měření bylo zjištěno, že počet úspěšně detekovaných snímků v blízkosti hranice strmě klesá na velmi malé oblasti (cca 2 mm). Z výsledků lze pozorovat, že představené řešení pro QR kódy má především užší, zhruba poloviční (s rostoucím h i méně), vzdálenostní rozsah detekce. Ten závisí především na velikosti pozičních značek. Pro důkaz byl měřen i ArUco marker o $h = 10$ mm. Strana poziční značky QR kódu verze 3 o $h = 40$ mm, má rozměr 9,7 mm. Porovnají-li se naměřené d_{max} těchto dvou vzorků, jde vidět, že si jsou velmi podobné. ArUco markery mají i menší d_{min}, to je dáno více průchody adaptivním prahováním, což je vysvětleno v podsekci 2.6.1. Tento postup vytvořená knihovna nevyužívá. U větší verze QR kódu zabírají poziční značky menší plochu vůči celkové. Jak je vidět z Obr. 4.10, detekční rozsah se v takovém případě pouze posune níže. Samozřejmě záleží na nastavení detekčních parametrů, tudíž výsledky platí především za stejného nastavení.

4.4.4 Měření úhlového rozsahu detekce

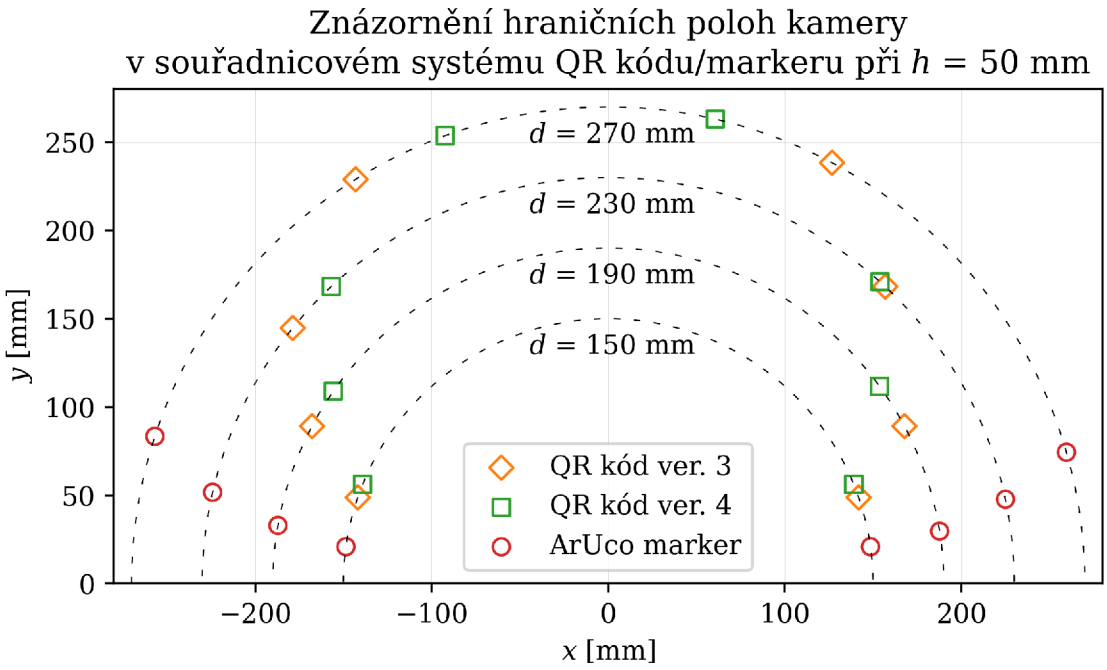
Základ postupu je uveden v podsekci 4.4.1. Pro měření úhlového rozsahu detekce bylo vybráno několik vzdáleností d na základě předchozího měření. V těchto vzdálenostech pak byl měněn úhel α podobným způsobem, jako v měření předchozím, pro zjištění hraniční polohy obou směrů natočení. Uvedeným způsobem byla naměřena data v Tab. 4.2 a 4.3. Pro lepší představu pak byl vytvořen Obr. 4.11, znázorňující hraniční polohy kamery v souřadnicovém systému QR kódů/markerů při $h = 50$ mm.

Tabulka 4.2: Výsledky měření úhlového rozsahu detekce. Část první.

	d [mm]:	**150**		**170**		**190**	
h [mm]	**Měřeno:**	**$+\alpha$ [°]**	**$-\alpha$ [°]**	**$+\alpha$ [°]**	**$-\alpha$ [°]**	**$+\alpha$ [°]**	**$-\alpha$ [°]**
30	QR kód ver. 3	64	62	55	44	39	37
30	QR kód ver. 4	58	53	47	45	31	33
30	ArUco ID 3	83	82	81	80	76	77
40	QR kód ver. 3	68	66	-	-	56	54
40	QR kód ver. 4	65	62	-	-	48	38
40	ArUco ID 4	82	83	-	-	79	80
50	QR kód ver. 3	71	71	-	-	62	62
50	QR kód ver. 4	68	68	-	-	55	54
50	ArUco ID 5	82	82	-	-	80	81

Tabulka 4.3: Výsledky měření úhlového rozsahu detekce. Část druhá.

	d [mm]:	**230**		**270**	
h [mm]	**Měřeno:**	**$+\alpha$ [°]**	**$-\alpha$ [°]**	**$+\alpha$ [°]**	**$-\alpha$ [°]**
30	QR kód ver. 3	-	-	-	-
30	QR kód ver. 4	-	-	-	-
30	ArUco ID 3	69	70	66	59
40	QR kód ver. 3	33	27	-	-
40	QR kód ver. 4	25	20	-	-
40	ArUco ID 4	73	70	67	67
50	QR kód ver. 3	51	43	32	28
50	QR kód ver. 4	43	42	20	13
50	ArUco ID 5	77	78	72	74



Obrázek 4.11: Graf z naměřených dat, znázorňující hraniční polohy kamery v souřadnicovém systému QR kódu/markeru, vycházející z naměřených dat, pro lepší představu.

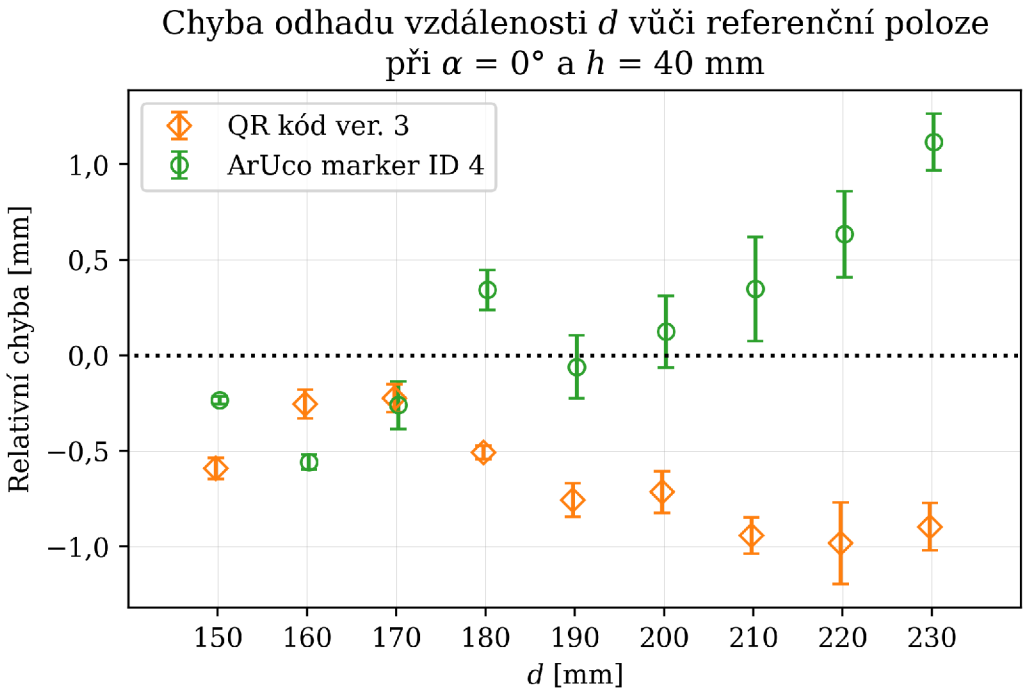
Při měření bylo opět zjištěno, že počet úspěšně detekovaných snímků v blízkosti hranice strmě klesá na oblasti asi 2°. Z výsledků lze pozorovat, že představené řešení pro QR kódy má znatelně horší úhlový rozsah detekce. To souvisí především se vzdáleností pozičních značek od osy rotace a jejich rozměrů. Mírně lze pozorovat i skutečnost, že některé úhly záběru QR kódu, jsou detekovatelné hůře. Pokud by byl QR kód umístěn na držáku diagonálně, byl by rozdíl zřetelnější. Právě i kvůli tomu, není QR kód vhodná referenční značka pro AR.

4.4.5 Měření přesnosti odhadu polohy kamery

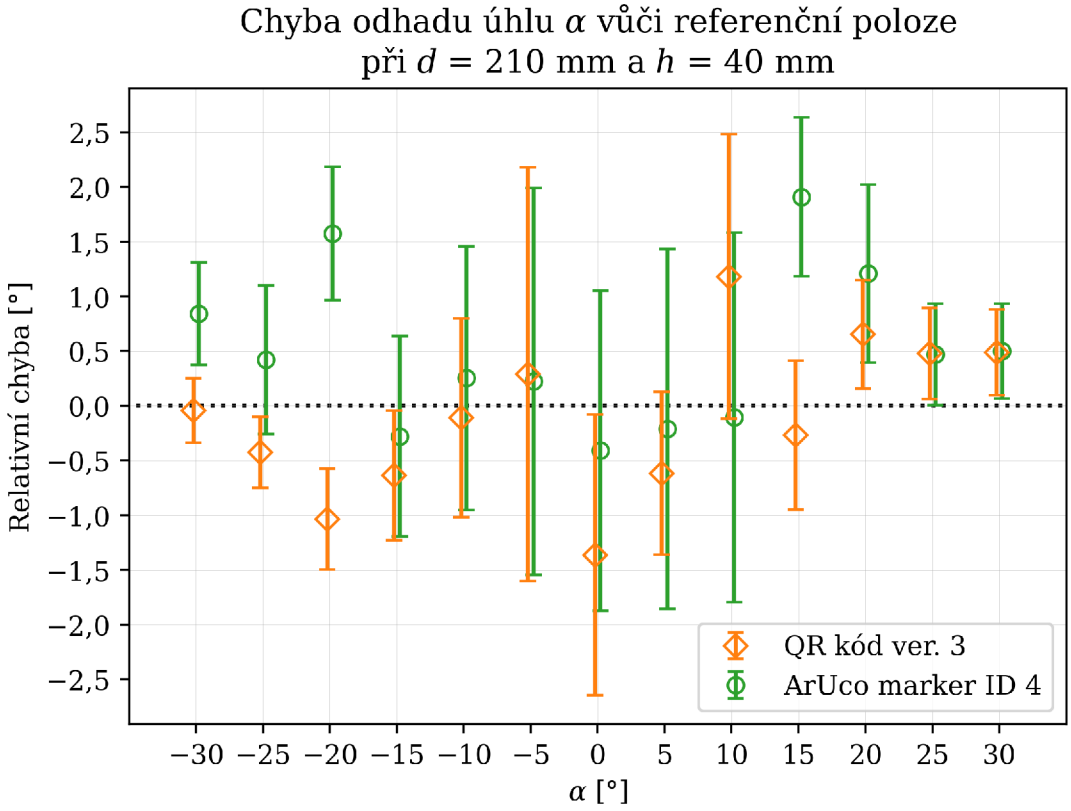
Základ postupu je uveden v podsekci 4.4.1. Z časových důvodů byl pro měření vybrán pouze QR kód verze 3 o $h = 40$ mm, jakožto zástupce s dostatečnou datovou kapacitou a ArUco marker o stejných rozměrech. Konkrétní měřené polohy jsou založeny na výsledcích předchozích dvou měření a uvedeny v Tab. 4.4, kde α_{abs} označuje absolutní hodnotu úhlu. Výsledky jsou naměřeny v obou směrech natočení. Dále pak zpracovány, jako průměry deviací od referenčních poloh a jejich směrodatné odchylky. Příklady takových výsledků jsou vyneseny na Obr. 4.10 a 4.11. Vzhledem k velkému množství naměřených dat, jsou uvedeny jen závěry z nich vyvozené.

Tabulka 4.4: Polohy pro měření přesnosti.

d [mm]	α_{abs} [°]									
150	0	5	10	15	20	25	30	35	40	45
160	0									
170	0	5	10	15	20	25	30	35	40	
180	0									
190	0	5	10	15	20	25	30	35		
200	0									
210	0	5	10	15	20	25	30			
220	0									
230	0	5	10	15	20	25				



Obrázek 4.12: Graf, znázorňující chybu odhadů polohy kamery vůči referenční, z výsledků měření po statistickém zpracování. Odhad vzdálenosti d.



Obrázek 4.13: Graf, znázorňující chybu odhadů polohy kamery vůči referenční, z výsledků měření po statistickém zpracování. Odhad úhlu α.

Přesnost odhadu polohy kamery vytvořené knihovny a příspěvkového modulu ArUco, je po již uvedeném statistickém zpracování v podstatě stejná. To je dáno hlavně velkou podobností použitých postupů. Průměrná velikost chyby ze všech odhadů d a α je u obou posuzovaných do 1 mm u vzdáleností a do 1° u úhlů. Co se však týče jednotlivých odhadů (v jednom snímku), největší chyba QR kódu byla 2 mm a 7°, AruCo markeru 12 mm a 8°. Takto velké chyby však nastávají jen výjimečně. Odhad polohy QR kódu je stabilní jen díky přichycení rohu B k vnější kontuře, což je popsáno v podsekci 4.3.1. K tomu je třeba mít vhodně nastavené parametry detektoru a zajistit dostatečně velkou tichou zónu. Toto měření bylo provedeno za takřka ideálních podmínek, vyjma problémů s kamerou, a tudíž nemusí zcela reprezentovat přesnost odhadu při reálném využití.

4.4.6 Celkové zhodnocení měření

Měření bylo značně ovlivněno webkamerou, konkrétně problémem s hloubkou ostrosti. Přesto šlo vytvořené řešení využívající QR kódy porovnat vůči přídavnému modulu ArUco v OpenCV a jeho markerům. Je však nutné brát v potaz, že nebyl testován široký rozsah poloh, a tudíž se nemusely projevit některé možné nedostatky vytvořeného řešení. Také provedení měřicího sestavení vnášelo do výsledků chyby. Z měření vyplývá, že řešení s QR kódy má výrazně menší oblast detekovatelných poloh, ale samotný odhad polohy je přesností srovnatelný s druhým zmíněným řešením. Vytvořené řešení by se tedy dalo užít spíše v aplikacích, kde není důležitý velký detekční rozsah, nebo kde výhody QR kódů převažují nad touto nevýhodou. Zároveň je možné s detekčním rozsahem manipulovat nastavením parametrů detektoru. To platí pro obě uvedené řešení.

4.5 Ukázková aplikace

Dle cíle v zadání má ukázková aplikace využívat principy AR a ukazovat výhody využití QR kódů oproti referenčním značkám. Jednou z hlavních výhod je, že konkrétní QR kódy není nutné předem definovat. Nicméně obsahuje-li odkaz na internetový server, definice konkrétního QR kódu stejně musí nastat na straně serveru (v nějakém souboru). Další výhodou je možnost datovou oblast do určité míry poškodit, v závislosti na zvoleně úrovni korekce. Kvůli postupu detekce popsaném v sekci 4.3, však nesmí být poškozeny poziční značky a kolem kódu musí být zachována dostatečná tichá zóna. Tou hlavní výhodou je právě možný datový obsah, kdy stačí aby QR kód odkazoval na příslušný soubor internetového uložiště a dále program využíval data a instrukce uložené právě na něm. Možné je i využití zkracovačů adres či jiných forem přesměrování. S tím se pojí i značné bezpečnostní riziko, na které je nutno brát ohledy.

4.5.1 Myšlenka aplikace

Nápad na aplikaci vznikl po konzultaci s vedoucím práce. Spočívá v natištění, či jiném připevnění QR kódu na Desku Plošných Spojů (DPS). V něm je zakódován odkaz na textový soubor, uložený na lokálním serveru. Ten pak obsahuje instrukce pro vizualizaci součástek a odkazy na příslušné dokumentace. Po kliknutí kurzoru myši na součástku se v prohlížeči otevře její dokumentace. Kamera bude umístěna do vzdálenosti, která zaručí ostrost obrazu, pro co nejlepší výsledky.

4.5.2 Provedení aplikace

Jak již bylo zmíněno, základ aplikace je postaven na vytvořené knihovně (více k ní v sekci 4.3). Téměř všechny důležité procesy zastává knihovna OpenCV. Další funkce jsou zastávány základními moduly programovacího jazyka Python. Pro čtení snímků z kamery je využito kódu z [27], důvod je popsán v podsekci 2.6.4. Jedná se o třídu *WebCamVideoStream*, která byla upravena pro užitou kameru. Knihovna OpenCV umožňuje projekci kontur do souřadnicového systému QR kódu, užitím vektorů rotace a translace získaným z odhadu polohy a modelu kamery. Takto je možné vykreslovat i text. Zároveň lze pomocí stejné knihovny zpracovávat polohu a stisk kurzoru myši v okně s obrazovým výstupem. Bylo usouzeno, že tyto jednoduché funkcionality budou postačovat a není nutné rozšiřovat množství užitých knihoven.

Základem je spuštění lokálního serveru v jehož souborovém systému se nachází textový soubor s instrukcemi. Pro tyto instrukce byl vytvořen přesný způsob jejich zápisu i čtení. Syntaxe umožňuje i zápis poznámek, které se ve vizualizaci neprojeví. Tento systém má několik příkazů jež jsou popsány v přiloženém textovém souboru *demo_syntax*. V QR kódu je pak vždy odkaz na konkrétní textový soubor.

Použití vytvořené knihovny již bylo popsáno v podsekci 4.3.4, akorát jsou pozměněna některá nastavení detektoru. To tak, aby bylo možné využít světlý QR kód na tmavém podkladu a detekce byla co nejspolehlivější. Dále byly vytvořeny tři třídy, které zajišťují důležité funkcionality.

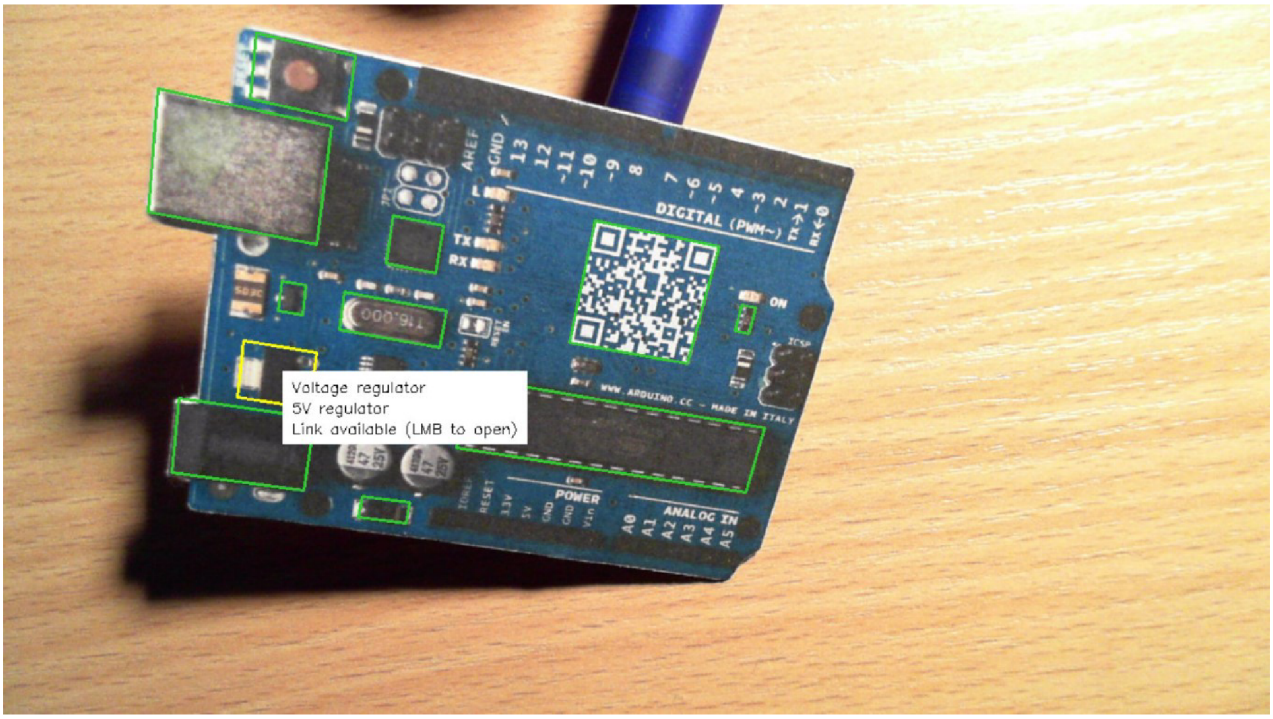
První je třída *Part*, která popisuje součástku. Její instance jsou jméno, specifikace (slouží spíše pro poznámky), kontura a odkaz. Objekt této třídy je vytvořen na základě instrukcí v textovém souboru. Kontura se v tomto souboru dá definovat dvěma způsoby, buďto bod po bodu, nebo určením souřadnic středu, šířky a výšky obdélníku. Všechny souřadnice se uvažují v souřadnicovém systému QR kódu. Další třídou je *PartsOrigin*, která přiřazuje k objektu QR kódu všechny objekty třídy *Part* jemu náležící. Poslední důležitou třídou je *PartMemory*, která pracuje velmi podobně jako třída *qar.Memory*, avšak uchovává seznamy objektů třídy *Part* přiřazené k ID ze sledovače.

Oproti normálnímu použití vytvořené knihovny se musí před vstupem do hlavní programové smyčky inicializovat i objekt třídy *PartMemory*. Dále budou popisovány procesy v každém snímku. Nejprve je použita funkce *qar.detectCodes()*. Pokud byl QR kód úspěšně dekódován a ještě nejsou data zapsaná v objektech pamětí, je zkontrolováno, zda obsahuje odkaz na textový soubor. Pokud ano, instrukce jsou přečteny, seznamy objektů třídy *Part* vytvořeny a přiřazeny k danému QR kódu. Pokud má QR kód obsaženou i informaci o svých rozměrech, je možné její hodnotu přepsat skrze instrukce v textovém souboru. Dále je použita funkce *qar.estimatePose()*. Poté následuje postupné vykreslení všech informací do obrazu, kdy se bere v potaz i pozice kurzoru myši v okně obrazu. Pokud se kurzor nachází nad vizualizovaným prvkem, jsou u kurzoru vypsány informace z instrukcí včetně indikace dostupnosti odkazu na dokumentaci. Je-li dostupná, stiskem levého tlačítka myši je otevřen internetový prohlížeč s odkazem na příslušnou dokumentaci.

Na Obr. 4.14 je snímek z běžící ukázkové aplikace, užitá DPS je Arduino Uno. Jelikož však zrovna nebyla žádná obdobná deska k dispozici, byla pouze upravena a vytisknuta její fotka [36], ve skutečné velikosti.

4.5.3 Zhodnocení a použitelnost aplikace

Uvedená aplikace byla vytvořena pouze pro účel ukázky možného použití a není zcela vyladěná. Např. při načítání instrukcí z textového souboru, je možné pozorovat zásek obrazu. Samotné prostředí pro koncového uživatele by potřebovalo snad jen grafické vylepšení. Pro rychlejší a snadnější tvorbu instrukcí by dále bylo vhodné vytvořit samostatné grafické prostředí. Aplikace je vytvořena tak, aby zvládla zpracovat i více obdobných QR kódů v obraze. Ke správnému fungování vyžaduje kalibrovanou kameru a s tím spojené poskytnutí její vnitřní matice a koeficientů zkreslení obrazu.



Obrázek 4.14: Snímek z běžící ukázkové aplikace.

5 Závěr

Cílem této práce bylo posoudit využití dvourozměrných čárových kódů pro odhad polohy v aplikacích rozšířené reality. Nejprve byl na základě rešerše zvolen, jako nejvhodnější dvourozměrný čárový kód pro tento účel, QR kód. Dále byla pro tvorbu algoritmu zvolena knihovna počítačového vidění OpenCV, společně s programovacím jazykem Python.

Kvůli snaze o neplýtvání výpočetním výkonem, vznikla myšlenka rozdělení procesů detekce a dekódování. Vzhledem k tomu, byly předmětem rešerše i další knihovny určené pouze pro čtení (dvourozměrných) čárových kódů. Většina takových knihoven však neumožňuje zpracovávat více QR kódů v obraze, nebo neumožňuje zmíněné rozdělení. Proto bylo přistoupeno k vlastní realizaci detekce a identifikace jednotlivých rohů. Kvůli rozdělení bylo nutné implementovat i jednoduchý sledovač objektů v obraze. Výsledkem tak je, že každý detekovaný QR kód v obraze, musí být úspěšně dekódován pouze jednou a poté už pouze sledován. Vždy jsou mu pak přiřazeny i jeho data. Pro dekódování byla zvolena knihovna ZBar, která obstojí i při neideálních obrazových podmínkách. Této knihovně je vždy poskytnut pouze výřez obrazu s jedním QR kódem. Zmíněný postup je součástí detekční funkce, vytvořené knihovny, pro využití QR kódů v aplikacích rozšířené reality.

Již uvedená vytvořená knihovna, je inspirována příspěvkovým modulem ArUco, knihovny OpenCV. Konkrétně základem detekčního procesu a jednoduchostí jeho použití pro odhad polohy kamery vůči ArUco markerům. Vlastní knihovna byla proto vytvořena tak, aby její použití záviselo pouze na třech funkcích a důležité parametry detektoru bylo možné komfortně přenastavit. Soubor knihovny je přiložen pod názvem *qr_code_augmented_reality.py*, dalším přiloženým souborem je *qar_pose_estimation_sample.py*, jež představuje ukázku jejího použití. Užití jednoduchého sledovače objektů přináší riziko záměny ID jednotlivých QR kódů. Na to je třeba myslet, při zvažování využití uvedené knihovny.

Vytvořené řešení, pro QR kódy, bylo porovnáno vůči řešení příspěvkového modulu ArUco, pro stejnojmenné markery, třemi druhy měření. Všechny byly provedeny pomocí kamery s hloubkou ostrosti přibližně 85-125 mm, což bylo dáno zřejmě její vadou. Z výsledků měření vzdálenostního a úhlového rozsahu detekce vyplývá, že vytvořené řešení, má možný pracovní prostor, vůči ArUco markerům stejné velkosti, znatelně menší. Vzdálenostní rozsah je zhruba poloviční a úhlový klesá rychleji než u protějšku. Přesnost obou řešení je pak, zejména kvůli podobnosti použitých postupů, stejná, což vyplývá z měření přesnosti odhadu polohy kamery. Výsledky je však nutné brát s rezervou, jelikož v důsledku hloubky ostrosti kamery, bylo možné proměřit jen vcelku malý rozsah poloh.

Nakonec byla vytvořena ukázková aplikace, zamýšlená jako intuitivní pomůcka pro práci s DPS. Spočívá v natištění, či jiném připevnění, QR kódu na desku, jeho načtení kamerou a následnou vizualizací součástek. Tento QR kód odkazuje pouze na textový soubor, s uvažovaným umístěním na lokálním serveru, obsahující instrukce k této vizualizaci. Těmi jsou: jméno, ohraničující rámeček, popis a odkaz na dokumentaci konkrétní součástky. Využitou výhodou je možnost poškodit část datové oblasti, beze ztráty dat. Další výhodu je (potenciálně) velké množství možných textových souborů.

Využití QR kódu pro odhad polohy kamery, je vhodné jen za určitých okolností. Především jsou to situace, kdy je potřebné obsáhnout větší množství dat a zároveň nepřichází v úvahu kombinace QR kódu s referenční značkou. Případně kde převažují výhody řešení s QR kódy nad jeho nevýhodami. Vždy je nutné mít na paměti možná omezení.

Seznam použitých zdrojů

[1] OWEN, C.B., FAN XIAO a P. MIDDLIN. What is the best fiducial? In: *The First IEEE International Workshop Agumented Reality Toolkit* [online]. IEEE, 2002 [cit. 2020-06-10]. DOI: 10.1109/ART.2002.1107021. ISBN 0-7803-7680-3. Dostupné z: `http://ieeexplore.ieee.org/document/1107021/`

[2] Hiro pattern. In: *Github* [online]. [cit. 2020-06-10]. Dostupné z: `https://github.com/artoolkit/ARToolKit5/blob/master/doc/patterns/Hiropattern.pdf`

[3] Apriltag tag36_11-00000. In: *Github* [online]. [cit. 2020-06-10]. Dostupné z: `https://github.com/AprilRobotics/apriltag-imgs/blob/master/tag36h11/tag36_11_00000.png`

[4] ArUco 4x4_1000-0. *Online ArUco markers generator* [online]. [cit. 2020-06-10]. Dostupné z: `https://chev.me/arucogen/`

[5] ARToolKit Home Page. *ARToolKit* [online]. Washington [cit. 2020-06-11]. Dostupné z: `http://www.hitl.washington.edu/artoolkit/`

[6] AprilTag. *APRIL Laboratory* [online]. Michigan [cit. 2020-06-11]. Dostupné z: `https://april.eecs.umich.edu/software/apriltag`

[7] *Applications of Artificial Vision* [online]. Córdoba [cit. 2020-06-11]. Dostupné z: `http://www.uco.es/investiga/grupos/ava/node/1`

[8] GARRIDO-JURADO, S, R MUÑOZ-SALINAS, F.J MADRID-CUEVAS a M.J MARÍN-JIMÉNEZ. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition* [online]. Elsevier, 2014, **47**(6), 2280-2292 [cit. 2020-06-11]. DOI: 10.1016/j.patcog.2014.01.005. ISSN 0031-3203. Dostupné z: `http://www.sciencedirect.com/science/article/pii/S0031320314000235`

[9] KATO, Hiroko, Keng T. TAN a Douglas CHAI. *Barcodes for mobile devices*. Cambridge: Cambridge University Press, 2010. ISBN 978-0-521-88839-4.

[10] *Free Online Barcode Generator* [online]. Steyr: TEC-IT [cit. 2020-06-11]. Dostupné z: `https://barcode.tec-it.com/en`

[11] BHASKAR, Raj. *Bar codes: Technology and implementation*. Tata McGraw-Hill Education, 2001. ISBN 978-0-07-463849-1.

[12] *Barcode Information Tutorials: FAQs & Reference Guides* [online]. Tampa: IDAutomation.com [cit. 2020-06-12]. Dostupné z: `https://www.barcodefaq.com/`

[13] Aztec Code. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2020-06-11]. Dostupné z: `http://en.wikipedia.org/wiki/AztecCode`

[14] *Software for Barcodes, 2D Codes, Label Printing, Data Collection* [online]. Steyr: TEC-IT [cit. 2020-06-12]. Dostupné z: https://www.tec-it.com/en/start/Default.aspx

[15] QRcode.com [online]. Denso Wave [cit. 2020-06-12]. Dostupné z: https://www.qrcode.com/en/

[16] *OpenCV: OpenCV Documentation* [online]. [cit. 2020-06-12]. Dostupné z: https://docs.opencv.org/master/

[17] PnP. In: *OpenCV: OpenCV Documentation* [online]. [cit. 2020-06-12]. Dostupné z: https://docs.opencv.org/master/pnp.jpg

[18] KAEHLER, Adrian a Gary R BRADSKI. *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. Sebastopol: O'Reilly, 2016, 990 s. ISBN 978-1-4919-3799-0.

[19] *OpenCV* [online]. [cit. 2020-06-13]. Dostupné z: https://opencv.org/

[20] *BoofCV* [online]. [cit. 2020-06-13]. Dostupné z: https://boofcv.org/index.php?title=Main_Page

[21] Github: ZXing. *Github* [online]. [cit. 2020-06-14]. Dostupné z: https://github.com/zxing/zxing

[22] Github: ZBar. *Github* [online]. [cit. 2020-06-14]. Dostupné z: https://github.com/mchehab/zbar

[23] Github: OpenCV contrib. *Github* [online]. [cit. 2020-06-14]. Dostupné z: https://github.com/opencv/opencv_contrib

[24] LOJAYA, Iksandi. OpenCV: QR Code detection and extraction. *DsynFLO* [online]. 25 October 2014 [cit. 2020-06-14]. Dostupné z: http://dsynflo.blogspot.com/2014/10/opencv-qr-code-detection-and-extraction.html

[25] QR Code Structure Example 3. In: *Wikipedia* [online]. 9 April 2013 [cit. 2020-06-14]. Dostupné z: https://commons.wikimedia.org/wiki/File:QR_Code_Structure_Example_3.svg

[26] ROSEBROCK, Adrian. Simple object tracking with OpenCV. *PyImageSearch* [online]. 23 July 2018 [cit. 2020-06-15]. Dostupné z: https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/

[27] TAYAL, Rachit. Github: Multithread frame reading. *Github* [online]. 18 Apr 2019 [cit. 2020-06-15]. Dostupné z: https://github.com/rktayal/multithreaded_frame_reading

[28] ČEPL, Miroslav. *Návrh a implementace autonomního dokování mobilního robotu*. Brno, 2019. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/116771. 69 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky. Vedoucí práce Jiří Krejsa.

[29] KAN, Tai-Wei, Chin-Hung TENG a Wen-Shou CHOU. Applying QR code in augmented reality applications. In: *Proceedings of the 8th International Conference on Virtual Reality Continuum and its Applications in Industry - VRCAI '09* [online]. New York, New York, USA: ACM Press, 2009, 2009, s. 253- [cit. 2020-06-15]. DOI: 10.1145/1670252.1670305. ISBN 9781605589121. Dostupné z: `http://portal.acm.org/citation.cfm?doid=1670252.1670305`

[30] AGUSTA, Gia M., Khodijah HULLIYAH, ARINI a Rizal Broer BAHAWERES. Applying Merging Convetional Marker and Backpropagation Neural Network in QR Code Augmented Reality Tracking. *International Journal on Smart Sensing and Intelligent Systems* [online]. 2013, **6**(5), 1918-1948 [cit. 2020-06-15]. DOI: 10.21307/ijssis-2017-620. ISSN 1178-5608. Dostupné z: `https://www.exeley.com/in_jour_smart_sensing_and_intelligent_systems/doi/10.21307/ijssis-2017-620`

[31] ZHANG, Huijuan, Chengning ZHANG, Wei YANG a Chin-yin CHEN. Localization and navigation using QR code for mobile robot in indoor environment. In: *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)* [online]. IEEE, 2015, s. 2501-2506 [cit. 2020-06-15]. DOI: 10.1109/ROBIO.2015.7419715. ISBN 9781467396745. Dostupné z: `https://ieeexplore.ieee.org/document/7419715`

[32] KATANACHO, Manuel, Wladimir DE LA CADENA a Sebastian ENGEL. Surgical navigation with QR codes. *Current Directions in Biomedical Engineering* [online]. 2016, **2**(1), 355-358 [cit. 2020-06-15]. DOI: 10.1515/cdbme-2016-0079. ISSN 2364-5504. Dostupné z: `https://www.degruyter.com/view/journals/cdbme/2/1/article-p355.xml`

[33] QR Code generator library. *Project Nayuki* [online]. [cit. 2020-06-16]. Dostupné z: `https://www.nayuki.io/page/qr-code-generator-library`

[34] *Microsoft LifeCam Studio: Full Information* [online]. [cit. 2020-06-15]. Dostupné z: `https://igotoffer.com/microsoft/lifecam-studio`

[35] WEISSTEIN, Eric W. Line-Line Intersection. *MathWorld: A Wolfram Web Resource* [online]. [cit. 2020-06-16]. Dostupné z: `https://mathworld.wolfram.com/Line-LineIntersection.html`

[36] ADAFRUIT INDUSTRIES. Arduino Uno R3 (Atmega328 - assembled). In: *Flickr* [online]. 8 November 2013 [cit. 2020-06-19]. Dostupné z: `https://live.staticflickr.com/7411/10748949904_4dc8440a8b_b.jpg`

Seznam použitých zkratek

AR Augmented reality, Rozšířená realita

GPL General Public License, Obecná veřejná licence

BSD Berkeley Software Distribution

QR Quick Response, Rychlá odezva

PnP Perspective n-Point

ID Identifikační číslo

DPS Deska Plošných Spojů

Seznam obrázků

Seznam tabulek

Seznam příloh

1. Přiložené soubory
2. Poznámky k použití přiložených souborů

Přílohy

Přiložené soubory

\ZIP

- **\server_files**
  - *demo_syntax.txt*
    - ... Vysvětlení syntaxe instrukcí pro ukázkovou aplikaci.
  - *uno.txt*
    - ... Instrukce pro přiložené foto. Využitelné ukázkovou aplikací.
- *camera_matrix.npy*
  - ... Vnitřní matice kamery ve formě NumPy pole tvaru (3, 3).
- *distortion_coefficients.npy*
  - ... Koeficienty zkreslení obrazu ve formě NumPy pole tvaru (1, 5).
- *qar_demo_app.py*
  - ... Ukázková aplikace. Nejprve je nutné spustit lokální server (níže).
- *qar_pose_estimation_sample.py*
  - ... Ukázka použití vytvořené knihovny s vykreslením pomocí *qar.drawAxes()*.
- *qr_code_augmented_reality.py*
  - ... Vytvořená knihovna.
- *simple_local_server.py*
  - ... Jednoduchý lokální server zajišťující fungování ukázkové aplikace.
- *photo_for_demo_A4_print_ready.pdf*
  - ... Upravené foto [36] k ukázkové aplikaci. Připraveno k tisku na A4.

Poznámky k použití přiložených souborů

V ukázkách je nastavený index kamery na hodnotu **1**. Pokud systém disponuje pouze jednou webkamerou, měl by být přepsán na **0**. Je definován jako *CAM_INDEX*. To se týká souborů *qar_demo_app.py* (řádek č. 499) a *qar_pose_estimation_sample.py* (řádek č. 69).

Pro vyzkoušení ukázkové aplikace je nejdříve nutné spustit soubor *simple_local_server.py*, až poté *qar_demo_app.py*. Aplikace pracuje pouze se světlými QR kódy na tmavém podkladu, viz. příklad v *photo_for_demo_A4_print_ready.pdf*.

Ukázka *qar_pose_estimation_sample.py* pracuje s tmavými QR kódy na světlém podkladu. Lze s ní odzkoušet detekci více kódů v obraze.

V souboru knihovny *qr_code_augmented_reality.py* jsou komentovány nastavitelné parametry detektoru. Konkrétně je třída *DetectorParameters* definována a komentována od řádku č. 187.