Knihovna předzpracování obrazu z více kamer

Lukáš Gazdík

Bakalářská práce
2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Lukáš Gazdík**
Osobní číslo:	**A19026**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Knihovna předzpracování obrazu z více kamer**
Téma práce anglicky:	**Multiple Camera Stream Preprocessing Library**

Zásady pro vypracování

1. Prostudujte zkreslení obrazu vznikající záznamem scény běžnou webovou kamerou.
2. Seznamte se s požadavky platformy ARK na obrazový vstup.
3. Navrhněte postup kalibrace obrazu z více kamer za účelem jeho sloučení.
4. Navrhněte architekturu knihovny pro automatizované zpracování obrazu z více kamer.
5. Implementujte jednotlivé kroky kalibrace obrazu v podobě knihovny jazyka C++.
6. Vytvořte dokumentaci a podpůrnou aplikaci pro prvotní kalibraci.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. KAEHLER, Adrian a Gary R. BRADSKI. Learning OpenCV 3: computer vision in C++ with the OpenCV library. Sebastopol: O´Reilly, 2017. ISBN 978-1491937990.
2. BRAHMBHATT, Samarth. Practical OpenCV (Technology in Action). New Yorku: Apress, 2013. ISBN 978-1430260790.
3. MEYERS, Scott. Effective modern C++. Sebastopol, CA: O´Reilly, [2015]. ISBN 978-1491903995.
4. ČUKIĆ, Ivan. Functional Programming in C++. Shelter Island: Manning Publications, 2019. ISBN 978-1617293818.
5. SZELISKI, Richard. Computer Vision: Algorithms and Applications. London: Springer, 2010. Texts in computer science. ISBN 978-1848829343.
6. PRINCE, Simon J. D. Computer vision: models, learning, and inference. New York: Cambridge University Press, 2012. ISBN 978-1107011793.

Vedoucí bakalářské práce: **Ing. Peter Janků, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **3. prosince 2021**
Termín odevzdání bakalářské práce: **23. května 2022**

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 24. ledna 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 23.05.2022

Lukáš Gazdík v. r.
podpis studenta

ABSTRAKT

V tejto bakalárskej práci je cieľom navrhnúť postup akým bude možné spojiť obraz z viacerých kamier do jedného celku. Takýto proces zlučovania obrazu zahŕňa kalibráciu kamier, zistenie vzťahu medzi kamerami a transformáciu výsledného živého prenosu. Zadania je realizované formou knižnice jazyku C++. Využité sú aj algoritmy a metódy z oblasti počítačového videnia, ktoré sú dostupné z open source knižnice Open CV. Súčasť práce je aj vytvorenie počítačovej aplikácie, ktorá bude demonštrovať použitie kalibračnej knižnice. V teoretickej časti sú popísané postupy a teória v oblasti kalibrácie kamery a algoritmy počítačového videnia použité pri vývoji knižnice. V praktickej časti je zdokumentovaná architektúra knižnice, proces zlúčenia obrazu, a popis aplikácie.

Klíčová slova: Kalibrácia, detekcia, zlučovanie, skreslenie, Open CV, C++

ABSTRACT

In this bachelor thesis, the aim is to propose a procedure by which it will be possible to merge a video stream from several cameras into one unit. This image merging process involves calibration of the cameras, finding the relationship between the cameras, and transforming the resulting live broadcast. The assignment is realized in the form of a library in the C ++ language. Algorithms and methods from the field of computer vision, which are available from the open-source library Open CV, are also used. Part of the work is also the creation of a computer application that will demonstrate the use of the calibration library. The theoretical part describes the procedures and theories in the field of calibration of cameras and computer vision algorithms used in the development of the library. The practical part documents the architecture of the library, the process of the process of image merging, and a description of the application.

Keywords: Calibration, detection, key point, stitching, distortion, Open CV, C++

Chcem sa na tomto mieste poďakovať vedúcemu práce pánovi Ing. Petru Janků, PhD. za pomoc s návrhom, ochotu, inšpiráciu a aj za spätnú väzbu pri vypracovaní tejto práce.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Počítačové videnie je oblasť ktorá má v dnešnej dobe využitie takmer všade, kde je prítomný mobilný telefón, webkamera kamera, alebo iné zariadenie na snímanie obrazu. Jednou z platforiem, ktorá sa zaoberá rozvojom a aplikáciu počítačového videnia je Open CV. Algoritmy a funkcie, ktoré táto knižnica ponúka sa dajú použiť na prácu s obrazom od jednoduchej úpravy farieb na snímke z fotoaparátu, cez detekciu tvári a objektov z video záznamu, až po zložitú 3D rekonštrukciu objektov, alebo tvorbu panorámy z viacerých snímok. V prípade tvorby mozaiky z obrazového vstupu nemusí ísť len o statické snímky, ale občas máme situáciu, kedy dvomi a viac kamerami snímame tú istú plochu. Rozdiel je len v tom, že každá kamera pozoruje scénu z mierne odlišného uhlu. V tejto situácií sa objavuje príležitosť zábery z týchto kamier nejakým spôsobom spojiť do jedného celku. Tým by sa napríklad zjednodušilo používanie takéhoto systému. Príkladom tejto situácie môže práca s Kilobotmi, konkrétne platforma ARK, ktorá využíva 4 kamery na snímanie arény, v ktorej sa Kiloboti testujú.

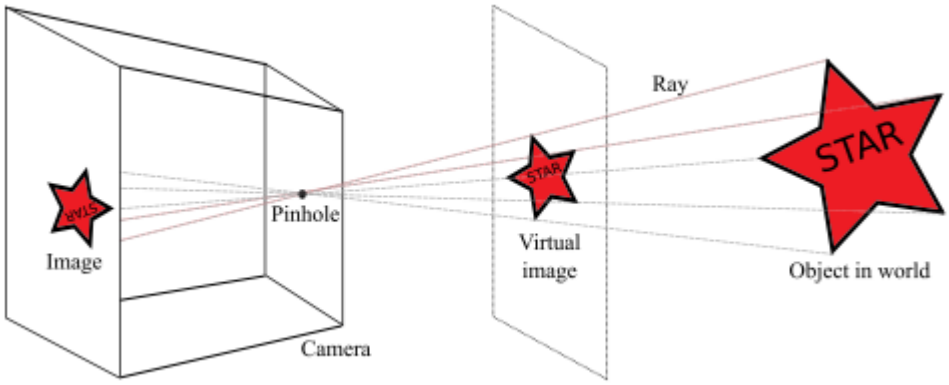
Cieľom tejto práce je navrhnúť a zostaviť algoritmus, ktorý túto problematiku zlúčenia záznamu z viacerých kamier vyrieši. Spojenie realtime záznamu z viacerých vstupov je proces, ktorý si vyžaduje najprv úpravu kamerového obrazu na vstupe. To znamená kalibrácia kamier a následná oprava radiálneho a perspektívneho skreslenia, ktoré sa objavujú pri každom type kamery. Takto upravený obrazový vstup z viacerých vstupných zariadení môžeme pomocou matematických výpočtov upraviť, transformovať a v konečnom dôsledku spojiť do jedného celku. Celý tento postup zlučovania obrazu bude implementovaný formou knižnice v programovacom jazyku C++ s využitím algoritmov, ktorú ponúka knižnica Open CV.

I. TEORETICKÁ ČASŤ

1 PARAMETRE KAMERY

1.1 Pin Hole Model

Ide o čisto geometrický model, ktorý popisuje proces, pri ktorom sa 3D body vo svete premietajú na 2D obrazovú plochu. Tento model udáva projekciu 3D bodu $\boldsymbol{w} = [u, v, w]^T$ do 2D bodu definovaného ako $\boldsymbol{x} = [x, y]^T$. Parametrami týchto rovníc sú vnútorné a vonkajšie parametre kamery. Virtuálny obraz je týmto spôsobom vytvorený na rovine obrazu, ktorá je posunutá od optického stredu pozdĺž osi *w* alebo aj optickej osi. V reálnom živote sa dierková kamera skladá z komory s malým otvorom v prednej časti a lúče odrážajúce sa z predmetu vo svete prechádzajú cez tento otvor a vytvárajú prevrátený obraz na zadnej strane komory.



Obrázok 1. Princíp Pin Hole modelu [1]

Takýto obraz je však otočený je hore nohami, ako je ukázané na Obrázku 1. Samotná dierka (bod, v ktorom sa lúče zbiehajú) sa nazýva optický stred. Bod, v ktorom optická os naráža na rovinu obrazu, je známy ako hlavný bod. Vzdialenosť medzi hlavným bodom a optickým stredom (t. j. vzdialenosť medzi rovinou obrazu a dierkou) je známa ako ohnisková vzdialenosť. [1]

1.1.1 Vonkajšie parametre kamery

Vonkajšie parametre kamery (camera extrinsic) závisia od umiestnenia a orientácie kamery v priestore a s vnútornými parametrami nemajú nič spoločné. Tieto parametre je možné zapísať do transformačnej matice. Tá mapuje body zo svetového súradnicového systému na súradnicový systém kamery. Matica týchto parametrov je kombináciu rotačnej matice *R* a prekladovej matice *T*, ktorá udáva polohu kamery vo fyzickom priestore. [2] [3]

$$EM = [\, R \mid T \,] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \end{bmatrix}$$

1.1.2 Vnútorné parametre kamery

Medzi vnútorné parametre (camera intrinsic) patrí ohnisková vzdialenosť, clona, zorné pole, rozlíšenie atď., ktoré riadia vnútornú maticu modelu fotoaparátu. Tieto vnútorné parametre kamery závisia od konštrukcie kmery a každá kamera má tieto parametre fixne dané. Použitie kalibračného vzoru alebo sady značiek je jedným zo spoľahlivejších spôsobov odhadu vnútorných parametrov. Matica týchto parametrov je transformačná matica, ktorá prevádza body zo súradnicového systému kamery na súradnicový systém pixelov. V tomto prípade je v rovnici *f* ohnisková vzdialenosť, *x* , *y* sú rozmery senzoru v milimetroch a *w* , *h* sú rozmery obrázku uvedené v pixeloch. [2] [3] [4]

$$IM = \begin{bmatrix} \frac{fw}{xs} & 0 & w/2 \\ 0 & \frac{fh}{ys} & h/2 \\ 0 & 0 & 0 \end{bmatrix}$$

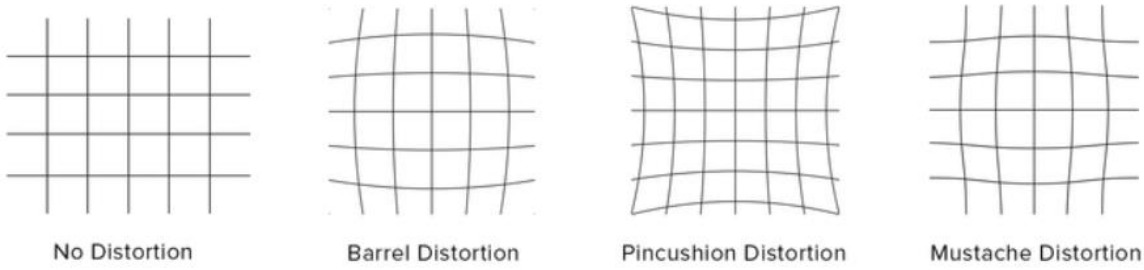
2 SKRESLENIA

Existujú dva typy skreslenia: optické a perspektívne. Oboje má za následok určitú deformáciu obrazu, buď jemnú, alebo veľmi výraznú. Zatiaľ čo optické skreslenia na Obrázku 2 sú spôsobené optickou konštrukciou šošoviek (a preto sa často nazýva „skreslenie šošovky“), skreslenie perspektívy je spôsobené polohou fotoaparátu vzhľadom na objekt alebo polohou objektu v rámci záberu. [5]

Obrázok 2. tri základné druhy skreslení, súdkovité, poduškové, kombinované [6]

2.1 Optické skreslenie

V počítačovom videní je možné označiť aj ako optická aberácia, ktorá deformuje a ohýba fyzické priame čiary a spôsobuje, že sa na obrázkoch javia zakrivené. Toto skreslenie sa preto bežne označuje aj ako „krivkové“. K optickému skresleniu dochádza v dôsledku optického dizajnu, keď sa používajú špeciálne prvky šošovky na zníženie sférických a iných skreslení. Optické skreslenie je v jednoduchosti chyba objektívu. Existujú tri známe typy optického skreslenia – konvexné (súdkovité), konkávne (poduškové) a komplexné (známe aj ako vlnité a zložité). „Dokonalé“ šošovky bez akéhokoľvek optickej vady je veľmi obťažné vyrobiť, preto väčšina šošoviek trpí aspoň jedným druhom skreslenia. Veľmi dobré šošovky majú prvky, ktoré výrazne znižujú skreslenie tam, kde to nie je pre naše oči viditeľné. Mnohé objektívy s veľkým priblížením, najmä fotoaparáty a kamery so super priblížením, trpia viacerými typmi skreslenia, ako sú súdkové a poduškové pri rôznych ohniskových vzdialenostiach.[5]



Obrázok 3. Optické aberácie [7]

2.1.1 Súdkové skreslenie (barrel distortion)

Širokouhlé kamery sa rozsiahle používajú pri navigácii, bezpečnostných kamerách alebo najmä v medicíne. Širokouhlé fotoaparáty, vybavené širokouhlým objektívom dokážu zachytiť väčšie pole na jedinom obrázku, ktorý tým zachytí viac informácií na jednej snímke. Avšak snímky zachytené týmito kamerami vykazujú súdkovité priestorové skreslenie spôsobené širokouhlou konfiguráciou šošovky fotoaparátu. Súdkovité skreslenie nastáva, keď sú rovné čiary zakrivené dovnútra v tvare suda. K súdkovitému skresleniu, dochádza preto, že zorné pole objektívu je oveľa širšie ako veľkosť obrazového snímača, a preto ho treba „stlačiť“, aby sa zmestil na pole snímača. Výsledkom je, že priame línie sú viditeľne zakrivené dovnútra, najmä smerom k extrémnym okrajom rámu. Čiary sa zobrazujú rovno v samom strede rámu a začínajú sa ohýbať smerom ďalej od stredu. Je to preto, že obraz je rovnaký v optickej osi (v strede šošovky), ale smerom k rohom sa jeho zväčšenie zmenšuje.

Súdkovité skreslenie je zvyčajne prítomné na väčšine širokouhlých objektívov s malou ohniskovou vzdialenosťou a mnohých objektívoch s priblížením s relatívne krátkymi ohniskovými vzdialenosťami. Miera skreslenia sa môže líšiť v závislosti od vzdialenosti fotoaparátu od objektu. Sudové skreslenie je použitím kompenzačných optických prvkov možné výrazne znížiť, ale úplné odstránenie takéhoto skreslenia je takmer nemožné. Niektoré objektívy majú množstvo prvkov na kompenzáciu skreslenia, ktoré výrazne zvyšujú hmotnosť aj veľkosť objektívu. To je dôvod, prečo sú širokouhlé šošovky zvyčajne väčšie a ťažšie ako štandardné šošovky. Čo sa týka opravy súdkovitého skreslenia ide zvyčajne o pomerne jednoduchý proces. Softvér na následné spracovanie môže ľahko vyriešiť problémy so súdkovým skreslením, pokiaľ má objektív v databáze podporný profil. Keďže každá šošovka je iná, takéto údaje o profile šošovky sa musia starostlivo otestovať a vytvoriť v laboratórnom prostredí. [5] [6] [8]

2.1.2 Poduškové skreslenie (pincushion distortion)

Poduškové skreslenie je presným opakom súdkovitého skreslenia, čiže rovné čiary sú zakrivené smerom von od stredu. Tento typ skreslenia sa bežne vyskytuje na teleobjektívoch a vyskytuje sa v dôsledku zväčšovania obrazu smerom k okrajom rámu od optickej osi. Tentoraz je zorné pole menšie ako veľkosť obrazového snímača, a preto je potrebné ho „natiahnuť“, aby sa zmestil na pole snímača. V dôsledku toho sa zdá, že rovné čiary sú v rohoch vytiahnuté nahor. Tento typ skreslenia je veľmi bežný najmä na objektívoch s veľkým priblížením. Kvalitné snímače s prvotriednym objektívom mávajú kompenzačné prvky, ktoré môžu výrazne znížiť poduškové skreslenie na zanedbateľnú úroveň.

Poduškové skreslenie môže byť veľmi silné aj na spotrebiteľských šošovkách, čo je na obraze rýchlo možné si všimnúť. Väčšina objektívov s priblížením, ktoré prechádzajú zo širokouhlých na štandardné, zvyčajne trpí súdkovým skreslením na najkratších ohniskových vzdialenostiach, ktoré postupne prechádza do poduškového skreslenia smerom k najdlhšiemu koncu. Rovnako ako súdkové skreslenie, aj poduškové skreslenie možno jednoducho opraviť v softvéri na následné spracovanie.[5] [6]

2.1.3 Kombinované - Fúzové skreslenie (moustache distortion)

Najhorším typom radiálneho skreslenia je fúzové skreslenie, alebo aj „vlnité“. Ide v podstate o kombináciu súdkového skreslenia a poduškového skreslenia. Priame čiary sa zdajú byť zakrivené dovnútra smerom k stredu rámu a potom zahnuté smerom von v rohoch. To je dôvod, prečo sa fúzové skreslenie často označuje ako „komplexná“ deformácia, pretože jej charakteristiky sú zložité a môže byť náročné sa s nimi vysporiadať. Aj keď tento typ skreslenia možno potenciálne opraviť, často si vyžaduje špecializovaný softvér. Pri pokuse vysporiadať sa s takým skreslením, je problém, že oprava súdkovitého skreslenia, spôsobí oveľa väčšie zakrivenie extrémnych rohov. Pri pokuse kompenzovať poduškové skreslenie, môže nastať ešte silnejšie súdkovité skreslenie smerom k stredu. Fúzovím skreslením trpia mnohé staršie, ako aj niektoré moderné šošovky.[5] [6]

2.2 Perspektívne skreslenie

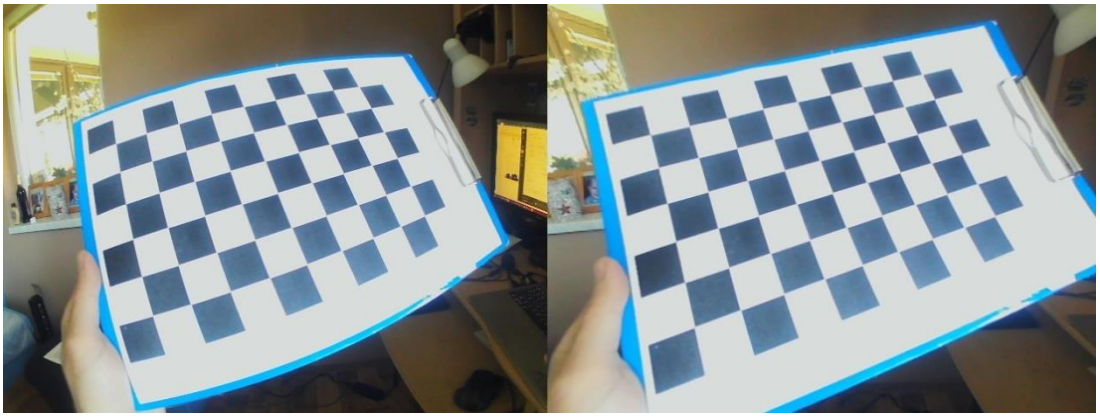
Ďalším typom skreslenia, ktorý sa často vyskytuje na obrázkoch, je perspektívne skreslenie. Na rozdiel od optického skreslenia nemá nič spoločné s optikou objektívu, a teda nejde o chybu objektívu. Ak sa pri premietaní trojrozmerného priestoru do dvojrozmerného obrazu nachádza objekt príliš blízko fotoaparátu, môže sa zdať neúmerne veľký alebo skreslený v

porovnaní s objektmi na pozadí (obrázok 4). To isté sa môže stať pri fotografovaní akéhokoľvek objektu vrátane ľudí. Perspektíve skreslenie je výsledkom fotografovania scény alebo objektu objektívom, ktorý má ohniskovú vzdialenosť alebo zorné pole, ktoré sa výrazne líši od nášho normálneho videnia. Je to vlastne funkcia vzdialenosti od fotoaparátu k objektu a spôsobuje, že objekt pôsobí neprirodzene. Pokiaľ ide o perspektívne skreslenie, ovplyvňuje ho iba poloha roviny obrazu (kamery).[4] [5]

Obrázok 4. Rozdiel v perspektíve vzhľadom na vzdialenosť objektu od kamery [4]

3 KALIBRÁCIA

Kalibrácia kamier bola vždy dôležitou úlohou v počítačovom videní a príslušnému výskumu sa už dlhé roky venuje značné úsilie. Úlohou kalibrácie je určiť vnútorné a vonkajšie parametre kamery, ktoré kvantitatívne popisujú vzťah medzi priestorovým 3D geometrickým umiestnením a príslušnými 2D obrazovými bodmi. Rôzne techniky na kalibráciu kamier možno rozdeliť hlavne do troch typov: lineárna kalibrácia, nelineárna kalibrácia a viacstupňová kalibrácia. Technika lineárnej kalibrácie využíva metódu najmenších štvorcov na získanie transformačnej matice, ktorá spája 3D body s ich 2D projekciami. Nelineárna kalibrácia sa týka skreslenia šošovky a na získanie optimálneho výsledku sa používa iteračný algoritmus. Viacstupňová kalibrácia je kombináciu oboch metód. [6] [9]



Obrázok 5. Hore záber pred kalibráciu a dole zábery po kalibrácií

Techniky kalibrácie sa delia na dve skupiny.

3.1.1 Kalibrácia pomocou referenčného objektu

Táto kalibrácia je založená na pozorovaní objektu v 3D svete, ktorého geometriu poznáme a vieme ju zmerať a parametrizovať. Ide väčšinou o jeden alebo viacero rovinných útvarov s jasne rozpoznateľným vzorom, ktoré sú kolmo natočené na kameru. Kvalita tejto kalibrácie potom závisí aj od použitého kalibračného vzoru. Ide o zložitejší techniku, pretože je potrebné ku kalibrácií zabezpečiť aj technickú výbavu alebo aparatúru. [10]

3.1.2 Samo kalibrácia

Pri tejto technike nie je potrebné používať kalibračný objekt. Kalibrácia sa vykonáva napríklad pohybom kamery v priestore a vytvorením snímok tohto nemenného priestoru z viacerých uhlov. Z poznávacích znakov, ktoré sú pri týchto snímkach rovnaké potom dokážeme vypočítať vonkajšie aj vnútorné parametre kamery. Táto technika je flexibilnešia, nie je však vždy úplne spoľahlivá.[10]

3.2 Homografia

V počítačovom videní je homografia definovaná ako projektívne mapovanie z jednej roviny do druhej. Je to technika hľadania rovnakých bodov medzi dvomi rovinnými plochami. Príkladom planárnej homografie môže byť mapovanie bodov z dvojrozmernej roviny reálneho sveta na povrch zobrazovača fotoaparátu, môže ísť aj o mapovanie dvoch obrazových rovín. Toto zobrazenie je možné vyjadriť pomocou maticového násobenia. [11]

$$H_{3x3} = \begin{pmatrix} H_{11} & H_{12} & H_{31} \\ H_{21} & H_{22} & H_{32} \\ H_{31} & H_{32} & H_{33} \end{pmatrix}$$

$$\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} \sim H \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

Najjednoduchší spôsob ako vyjadriť homografiu je s maticou 3x3 a pevnou mierkou. Takáto matica zahŕňa časti matíc vonkajších aj vnútorných parametrov danej kamery. Homografia mapuje [u, v], pixely z jednej roviny, na [u′, v′], pixely na druhý roviny. Na to, aby sme dokázali vytvoriť maticu homografie, potrebujeme poznať minimálne štyri body jednej roviny a tie isté príslušné body druhej roviny. Vytvorená matica homografie môže byť použitá na perspektívnu transformáciu obrazu, kedy je obraz deformovaný tak aby sa jeho koordináty zhodovali s koordinátami iného obrazu, čo spôsobí, že obraz vyzerá ako by bol vytvorený z rovnakého pozorovacieho uhlu. Taktiež je možné podľa bodov, ktoré majú obrazy spoločné nájsť zhody medzi obrazmi. To sa využíva na detekciu objektov.[3]

3.3 Transformácie

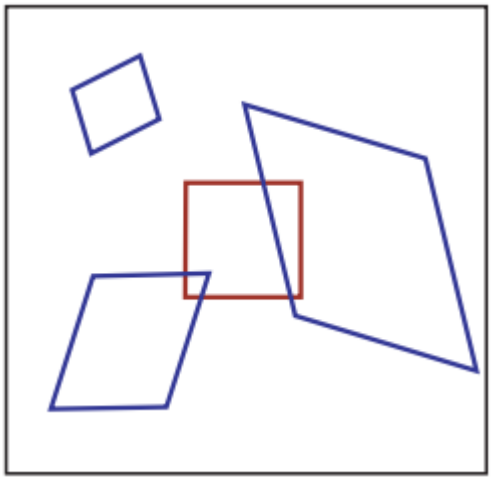
Geometrické transformácie obrázkov sú len transformácie obrázkov, ktoré sa riadia geometrickými pravidlami. Vlastnosťou týchto transformácií je, že sú všetky lineárne a teda môžu byť vyjadrené ako matice a transformácia obrazu sa rovná násobeniu matíc. Pri manipulácií s obrazom existujú 2 varianty geometrických transformácií: afinné

transformácie, to sú také, ktoré používajú maticu 2 × 3, a perspektívne alebo homografické transformácie, ktoré sú založené na matici 3 × 3.

Najjednoduchšou z geometrických transformácií je otáčanie a zmena mierky obrázka. Existovať samozrejme aj iné komplikovanejšie geometrické transformácie. Geometrické transformácie obrazu sú dôležitou súčasťou všetkých programov počítačového videnia, ktoré sa zaoberajú skutočným svetom, pretože vždy existuje perspektívna transformácia medzi svetom a obrazovou rovinou kamery, ako aj medzi rovinami obrazu v dvoch polohách kamery. [11] [12]

3.3.1 Afinné Transformácie

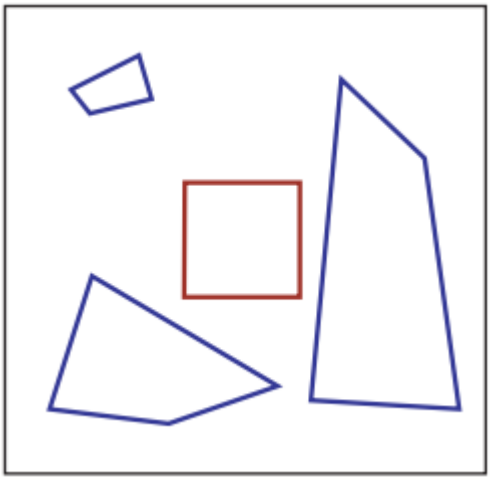
Afinná transformácia je akákoľvek lineárna transformácia, ktorá môže byť vyjadrená vo forme násobenia matice, po ktorej nasleduje pridanie vektora. V Open CV je štandardným štýlom reprezentácie takejto transformácie matica 2 × 3. Tento typ transformácie zachováva „paralelnosť“ čiar po transformácii. To znamená že tiež zachováva body ako body, priamky ako priamky a pomer vzdialeností bodov pozdĺž priamok. Nezachováva však uhly medzi rovnými čiarami. Afinné transformácie zahŕňajú všetky typy rotácie, translácie a zrkadlenia snímky. Týmito transformáciami je napríklad možné previesť obdĺžniky na rovnobežníky, stlačiť alebo rozšíriť tvar, môžu objekt otáčať alebo meniť jeho mierku. [11]



Obrázok 6. Aplikácia afinných transformácií [1]

3.3.2 Perspektivné transformácie

Perspektívne transformácie sú všeobecnejšie ako afinné transformácie. Nie vždy zachovávajú „rovnobežnosť“ línií. Ale, keďže sú všeobecnejšie, sú aj praktickejšie – takmer všetky transformácie, s ktorými sa stretávame v pri natáčení videa, alebo fotografovaní, sú perspektívne transformácie. Knižnica Open CV má na aplikáciu tohto typu transformácie definovanú funkciu *cv::warpPerspective*.



Obrázok 7. Aplikácia perspektívnych transformácií [1]

3.4 Detekcia kľúčových bodov

Na to aby sme dokázali zlúčiť viacero obrazových vstupov do jedného, potrebujeme najprv nájsť body alebo elementy, ktoré majú tieto obrázky spoločné. Ku tomuto účelu slúži techniky „Feature detection“ a „Feature matching“ a sú základnou súčasťou mnohých aplikácií počítačového videnia. Obrázok 8 znázorňuje vyhľadávanie takýchto elementov. Uvažujme 2 páry obrázkov. Prvý pár zachytáva rovnakú plochu, len jeden obrázok je viac posunutý na jednu stranu, a druhý pár zachytáva jeden objekt, napríklad dom, ale každý obrázok je vytvorený snímaním z iného uhlu. Pri prvom páre možno budeme chcieť zarovnať dva obrázky tak, aby sa dali hladko spojiť do zloženej mozaiky. Pre druhý pár môžeme chcieť vytvoriť hustú množinu bodov, aby bolo napríklad možné skonštruovať 3D model. [1] [6]

Obrázok 8. Kľúčové body sú označené farebnými krúžkami

Je zrejmé, že ak vyberieme bod na veľkej prázdnej ploche, potom nebude ľahké nájsť ten istý bod v ďalšej snímke videa. Ak sú všetky body na ploche veľmi podobné alebo rovnaké, bude problém nájsť ten istý bod v nasledujúcich snímkach. Na druhej strane, ak si vyberieme bod, ktorý je jedinečný, tak máme celkom dobrú šancu, že tento bod opäť nájdeme. V praxi by bod alebo prvok, ktorý vyberieme, mal byť jedinečný alebo takmer jedinečný a mal by byť parametrizovateľný (taký aby sme mohli určiť jeho polohu, alebo rotáciu vzhľadom ku okolitému prostrediu) tak, aby ho bolo možné porovnať s inými bodmi na inom obrázku. Bod, ku ktorému je priradená silná derivácia, môže byť na nejakej hrane, no stále môže vyzerať ako všetky ostatné body pozdĺž tej istej hrany. Ak sa však v blízkosti pozorujú silné deriváty v dvoch rôznych smeroch, potom môžeme dúfať, že tento bod bude s väčšou pravdepodobnosťou jedinečný.

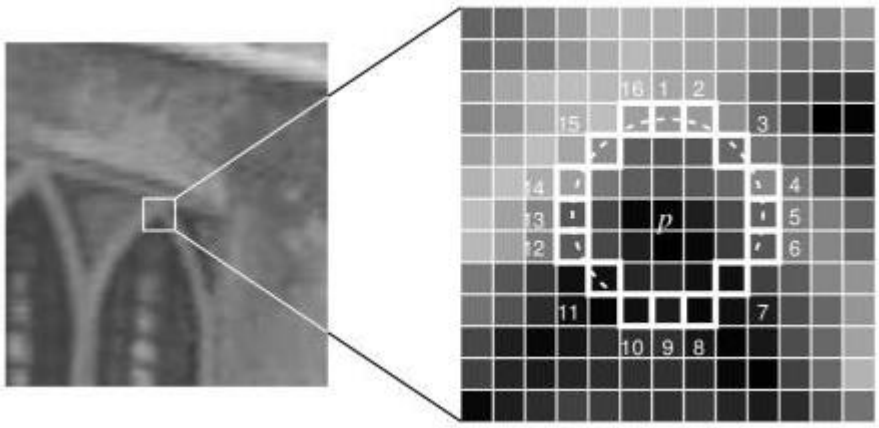
Tieto druhy lokalizovaných prvkov sa často nazývajú kľúčové body alebo záujmové body (alebo dokonca rohy) a sú často opísané výskytom políčok pixelov obklopujúcich umiestnenie bodu. Rohy sú viac intuitívne ako hrany. Väčšinou obsahujú dostatok informácií na to, aby sa dali vybrať na jednom aj druhom snímku. Ďalšou triedou dôležitých prvkov sú hrany. Tie možno priradiť na základe ich orientácie a miestneho vzhľadu (profily hrán) a môžu byť tiež dobrými indikátormi hraníc objektov v sekvenciách viacerých obrázkov. Hrany môžu byť zoskupené do dlhších kriviek a priamych línií, ktoré možno priamo

porovnávať alebo analyzovať, aby sa našli súbežníky, a teda interné a externé parametre kamery. Hrany a čiary poskytujú informácie, ktoré dopĺňajú kľúčové body aj deskriptory založené na regiónoch a sú vhodné na popis hraníc objektov a objektov vytvorených človekom. Open CV ponúka niektoré funkcie na jednoduché nájdenie týchto kľúčových bodov, ktoré sú vhodnými kandidátmi na sledovanie a párovanie. Najznámejšie sú ORB, BRISK, SIFT, SURF, FAST, KAZE, AKAZE iné. [6] [11] [13] [14]

3.4.1 ORB

Pre mnohé aplikácie je rýchlosť detekcie kľúčových bodov nielen užitočná, ale aj nevyhnutná. Platí to najmä pre úlohy, pri ktorých sa očakáva, že budú prebiehať v reálnom čase na video dátach, ako sú aplikácie rozšírenej reality alebo robotiky. Z tohto dôvodu bola vyvinutá funkcia ORB (Oriented FAST and Rotated BRIEF) s cieľom poskytnúť rýchlejšiu alternatívu k SIFT alebo SURF. Jeho slávou je extrémne rýchla prevádzka, pričom na presnosť výkonu obetuje veľmi málo.

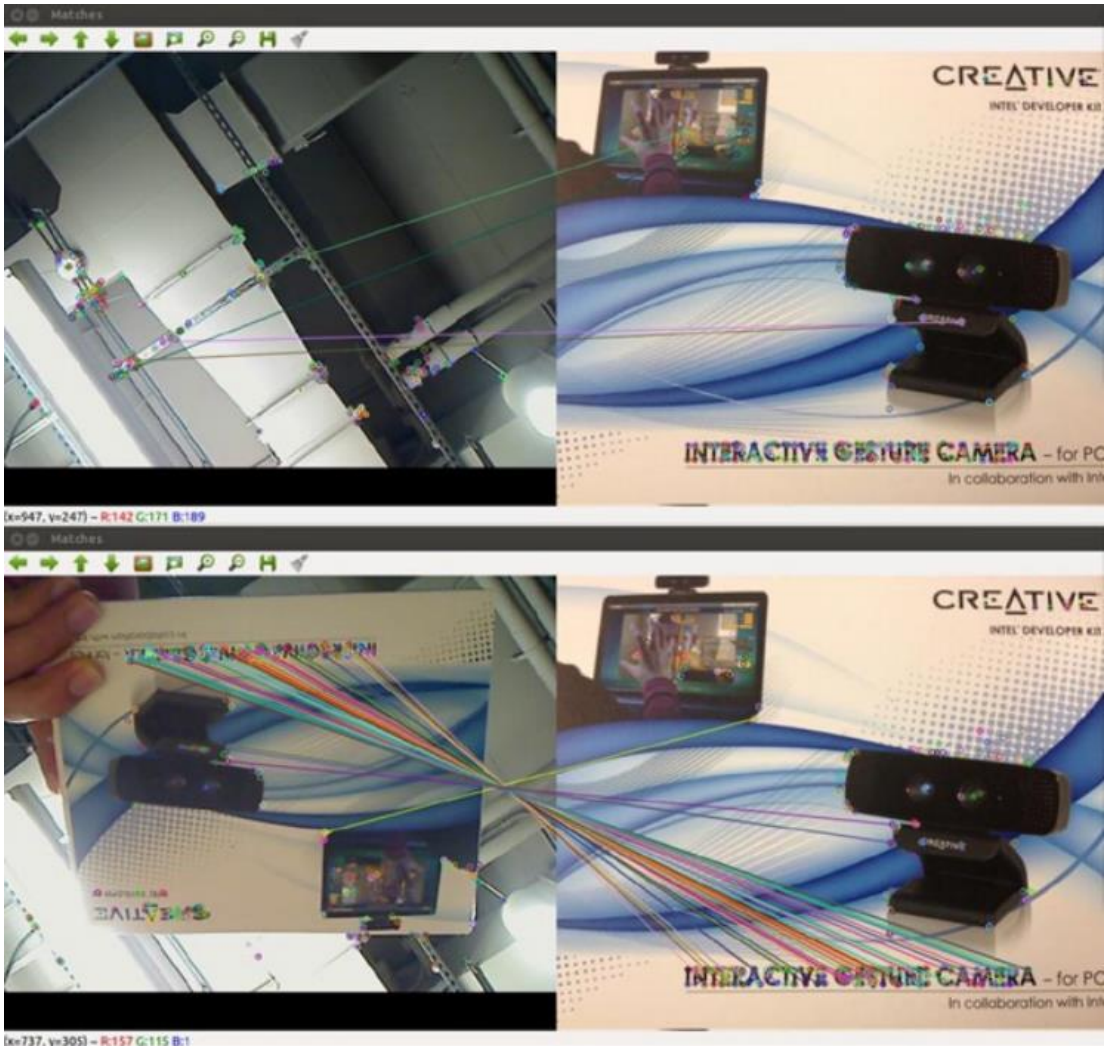
Algoritmus je kombináciou detekcie kľúčových bodov FAST (Features from Accelerated Segment Test) s orientáciou pridanou do algoritmu deskriptora kľúčových bodov BRIEF (Binary Robust Independent Elementary Features) upraveného tak, aby zvládal orientované kľúčové. Pôvodný detektor kľúčových bodov FAST testuje 16 pixelov v kruhu okolo pixelu, ukážka na Obrázku 9.



Obrázok 9. *p* je stredový bod vyhľadávania [15]

Ak je stredový pixel tmavší alebo jasnejší ako prahový počet pixelov zo 16, ide o roh. Aby sa tento postup zrýchlil, na rozhodnutie o efektívnom poradí kontroly 16 pixelov sa používa prístup strojového učenia. Prispôsobenie FAST v ORB deteguje rohy vo viacerých mierkach

vytvorením mierkovej pyramídy obrazu a pridáva orientáciu do týchto rohov nájdením ťažiska intenzity. BRIEF je založený na filozofii, že kľúčový bod na obrázku možno dostatočne opísať sériou testov binárnej intenzity pixelov okolo kľúčového bodu. To sa vykonáva výberom párov pixelov okolo kľúčového bodu a potom porovnaním dvoch intenzít. Test vráti hodnotu 1, ak je intenzita prvého pixelu vyššia ako intenzita druhého, a v opačnom prípade vráti hodnotu 0. [11] [12]



Obrázok 10. Detekcia a párovanie kľúčových bodov algoritmom ORB [12]

3.4.2 SIFT

SIFT (Scale invariant feature transform) je pravdepodobne najznámejší a široko implementovaný algoritmus na detekciu a popis kľúčových bodov, ktorý je dnes k dispozícii. Metóda popisu extrahuje okolie 16x16 okolo každého zisteného prvku a ďalej segmentuje oblasť do podblokov, čím sa celkovo vykreslí 128 hodnôt bin. SIFT je výrazne invariantný ku rotácií a mierke obrázku. To znamená, že algoritmus rozpozná objekty, ktoré majú

rovnaký vzhľad, ale väčšiu alebo menšiu veľkosť na testovacom obrázku v porovnaní s tréningovým objektom, ktorý je otočený (približne v mierke kolmej na obrázok) v porovnaní s cvičným objektom, alebo sú kombináciu týchto dvoch variant.

Je to preto, že kľúčové body extrahované SIFT majú priradenú veľkosť a orientáciu. Mierka sa vzťahuje na pojem veľkosť, pri ktorej je objekt videný, alebo ako ďaleko je kamera od objektu. Väčšia mierka znamená, že objekt vyzerá menší (pretože kamera sa vzďaľuje) a naopak. Vyššia mierka sa zvyčajne dosiahne vyhladením obrazu a prevzorkovaním. Nižšia mierka sa dosiahne podobným procesom vyhladzovania a vzorkovania. Hlavnou nevýhodou tohto algoritmu sú vysoké výpočtové nároky. [12] [13]

3.4.3 BRISK

Tento deskriptor je snahou vylepšiť algoritmus BRIEF, lepšou organizáciu binárnych pozorovaní, aby sa zvýšila robustnosť poznávacích znakov. Používa vlastný detektor kľúčových bodov AGAST (zrýchlená verzia FAST) zameraný na vyhľadávanie rohov a vylepšený tak, aby identifikoval veľkosť aj rotáciu kľúčového bodu. BRISK identifikuje veľkosť poznávacieho bodu tak, že najprv vytvorí obrázkovú pyramídu s fixným počtom veľkostí tohto obrázka. Použitím detektora AGAST potom hľadá poznávacie znaky na obrázkoch tejto pyramídy a následne odfiltruje body, ktorých skóre nie je najväčšie zo susedných bodov, a tým ostanú iba najlepšie kľúčové body.

BRISK je tvorený sériou kruhov okolo stredového bodu. Na týchto kruhoch sa porovnáva jas v dvoch podskupinách nazývaných páry na krátku vzdialenosť a dlhú vzdialenosť. Tieto páry sú určené na základe minimálnej a maximálnej vzdialenosť. Páry s krátkym dosahom tvoria deskriptor, zatiaľ čo páry s dlhým dosahom sa používajú na výpočet orientácie. Tento algoritmus je tiež všeobecné výpočtovo nenáročný. [11] [13]

4 OPEN CV

Open CV bol spustený v auguste 1999 na konferencii Computer Vision and Pattern Recognition. Gary Bradski založil Open CV v spoločnosti Intel so zámerom urýchliť výskum a využitie reálnych aplikácií počítačového videnia. Open CV má dnes takmer viac ako 3 000 funkcií, má viac ako 14 miliónov stiahnutí, trend výrazne presahuje 200 000 stiahnutí za mesiac a denne sa používa v miliónoch mobilných telefónov, rozpoznáva čiarové kódy, spája panorámy a zlepšuje obrázky pomocou výpočtovej fotografie. Open CV funguje v robotických systémoch, rozpoznáva predmety na dopravných pásoch, pomáha samo riadiacim autám vidieť, spravuje mapovanie v systémoch virtuálnej a rozšírenej reality, pomáha vykladať nákladné autá a palety v distribučných reťazcoch, používa sa aj v aplikáciách, ktoré podporujú bezpečnosť v baniach, spracúvajú mapy Google a snímky Streetview a implementujú robotiku Google X a mnohých ďalších.

Knižnica Open CV bola prerobená z C na moderný, modulárny C++ kompatibilný s STL a Boost. Knižnica bola prispôsobená moderným štandardom vývoja softvéru s distribuovaným vývojom na Git, robotmi na nepretržité zostavovanie, unit testami Google, komplexnou dokumentáciou a tutoriálmi. Open CV mal byť multiplatformový od začiatku, keď zahŕňal Windows, Linux a Mac OS X. Pokračuje v aktívnej podpore týchto desktopových operačných systémov, ale teraz pokrýva aj mobilné verzie s Androidom a iOS. Má optimalizované verzie pre architektúry Intel, ARM, NVidia GPU a čipy Movidius, ale funguje aj s Xylinx Zync FPGA. Okrem efektívneho zdrojového kódu C++ má rozsiahle rozhrania v Pythone (kompatibilné s NumPy), Java a MATLAB. Open CV tiež pridalo novú, nezávislú sekciu spravovanú používateľmi – Open CV Contrip. V tejto verzií, sú všetky rutiny samostatné a dodržiavajú štýl a dokumentáciu Open CV. S opencv_contrib drží Open CV krok s najnovšími algoritmami a aplikáciami v oblasti počítačového videnia.[11] [12] [16]

4.1.1 Mat

Základným dátovým typom je *cv::Mat*, okolo ktorého sa točí celá implementácia knižnice Open CV v C++. Prevažná väčšina funkcií v knižnici Open CV sú členmi triedy *cv::Mat*, a táto trieda *cv::Mat* sa používa na reprezentáciu hustých polí ľubovoľného počtu rozmerov. Hustý v tomto kontexte znamená, že pre každý záznam v poli je v pamäti uložená hodnota údajov zodpovedajúca tomuto záznamu, aj keď je tento záznam nula. Väčšina obrázkov je napríklad uložená ako husté polia. Alternatívou by bolo riedke pole. V prípade riedkeho poľa

sa zvyčajne ukladajú iba nenulové položky. To môže viesť k veľkej úspore úložného priestoru, ak je veľa položiek v skutočnosti nulových, ale môže to byť veľmi nehospodárne, ak je pole relatívne husté. [11] [12]

4.1.2 VideoCapture

Na zachytávanie a snímanie videa v Open CV slúži objekt *cv::VideoCapture*. Tento objekt môže otvárať a zatvárať video súbory rôznych video formátov. Tento objekt zachytenia dostane reťazec obsahujúci cestu, alebo názov súboru videa, ktoré sa má otvoriť. Po otvorení bude objekt zachytávania obsahovať všetky informácie o prečítanom video súbore vrátane informácií o stave. Takýmto spôsobom sa je možné napojiť na práve snímajúcu video kameru, alebo načítať video súbor z počítača. Z tohoto objektu sa dá inštancia snímky priamo uložiť do *cv::Mat* a pracovať s touto snímkou ako s obrázkom. [11]

5 C++ PROGRAMOVACÍ JAZYK

C++ je programovací jazyk vyvinutý Bjarne Stroustrupom v roku 1979 v Bell Labs. C++ sa považuje za jazyk strednej úrovne, pretože obsahuje kombináciu funkcií jazyka na vysokej a nízkej úrovni. C++ sa zrodilo ako rozšírenie programovacieho jazyka C, ktoré umožňuje programátorom písať objektovo orientovaný kód. Je teda nad množina jazyka C a prakticky každý legálny program v jazyku C je legálnym programom C++. C++ beží na rôznych platformách, ako sú Windows, Mac OS a rôzne verzie UNIX. C++ je otvorený jazyk štandardizovaný podľa ISO. Istý čas nemalo C++ žiadnu oficiálnu normu a bolo udržiavané štandardom, avšak od roku 1998 je C++ štandardizovaný výborom ISO.

Jedná sa o kompilovaný jazyk, čo znamená, že sa kompiluje priamo do natívneho kódu stroja, čo mu umožňuje, ak je dobre optimalizovaný, byť jedným z najrýchlejších jazykov na svete. Je to strongly-typed jazyk. Tento koncept je používaný na označenie programovacieho jazyka, ktorý presadzuje prísne obmedzenia na miešanie hodnôt s rôznymi dátovými typmi. C++ je jazyk, ktorý od programátora očakáva, že bude vedieť, čo robí, no vo výsledku umožňuje neuveriteľné množstvo kontroly. Od najnovšieho štandardu podporuje C++ manifestné aj odvodené písanie, čo umožňuje flexibilitu a zjednodušenie tam, kde je to potrebné. Podporuje statickú aj dynamickú typovú kontrolu. Umožňuje kontrolu typových konverzií buď v čase kompilácie, alebo v čase spustenia, čo opäť ponúka ďalší stupeň flexibility. Väčšina typových kontrol v C++ je však statická.

Výhodou je aj jeho prenosnosť. Ako jeden z najčastejšie používaných jazykov na svete a ako otvorený jazyk má C++ širokú škálu kompilátorov, ktoré bežia na mnohých rôznych platformách, ktoré ho podporujú. Kód, ktorý používa výhradne štandardnú knižnicu C++, pobeží na mnohých platformách s malými alebo žiadnymi zmenami. C++ je jazyk, ktorý priamo stavia na C. Je kompatibilný s takmer všetkým C kódom a môže používať knižnice C s niekoľkými alebo žiadnymi úpravami kódu knižníc. Ponúka tiež veľa možností paradigmy, čo znamená že má pozoruhodnú podporu pre procedurálne, funkčné, generické aj objektovo orientované programovacie paradigmy. [17] [18]

Procedurálne programovanie, niekedy aj ako imperatívne programovanie, je paradigma ktorá využíva lineárny prístup. Procedurálny programovací jazyk podporuje koncepciu procedúr a podprogramov (známych aj ako funkcie alebo metódy).

Funkcionálne programovanie je podmnožinou deklaratívneho programovania, ktoré sa snaží vyjadriť problémy pomocou matematických rovníc a funkcií. Snaží sa vyhnúť

konceptom stavov a meniteľných premenných, ktoré sú bežné v imperatívnych jazykoch. Hlavnou filozofiou funkčného programovania je, aby sa programátor nezaoberal tým, ako by mal program fungovať, ale skôr tým, čo by mal robiť.[17]

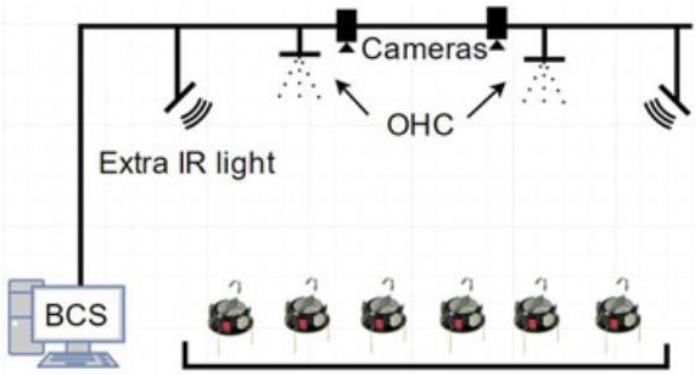
Generické programovanie sa zameriava na písanie kostrových algoritmov z hľadiska typov, ktoré budú špecifikované, keď sa algoritmus skutočne použije, čo umožňuje určitú zhovievavosť voči programátorom, ktorí sa chcú vyhnúť prísnym pravidlám pre písanie. Ak je dobre implementovaná, môže to byť veľmi silná paradigma.

Objektovo orientované programovanie je podmnožina štruktúrovaného programovania, ktoré vyjadruje programy v termínoch „objektov“, ktoré sú určené na modelovanie objektov v reálnom svete. Takáto paradigma umožňuje opätovné použitie kódu pomocou dedičnosti a má byť ľahko zrozumiteľná. [18] [19]

6 PLATFORMA ARK

ARK - rozšírená realitu pre Kilobotov, je systém, ktorý rozširuje možnosti Kilobotov tým, že umožňuje robotom pracovať v rozšírenej realite. Tento systém umožňuje Kilobotom prístup k prispôsobeným informáciám na základe ich polohy a stavu. Na druhej strane môžu upravovať svoje virtuálne prostredie, ktoré potom môžu vnímať iné roboty. ARK navyše značne uľahčuje a skracuje čas potrebný na obsluhu veľkých rojov automatizáciou niekoľkých nevyhnutných krokov pri nastavovaní experimentov, ako je polohovanie, kalibrácia motora a priradenie jedinečného ID. Nakoniec ARK poskytuje funkcie na logovanie a zaznamenávanie experimentálnych údajov na následnú analýzu.

ARK sa skladá z troch komponentov. Prvým je kamerový sledovací systém, ktorý poskytuje údaje o polohe a stave robota v reálnom čase. Druhý je upravený vysielač vysielajúci infračervené (IR) signály na komunikáciu s Kilobotmi. A tretím komponentom je základná riadiaca stanica na koordináciu systému a simuláciu virtuálnych prostredí. Systém ARK sa odlišuje od ostatných alternatívnych technológií tým, že ponúka ďalšie funkcie za podstatne nižšiu cenu, čo bolo preukázané v škálovaných robotických experimentoch. Ukážka architektúry na Obrázku 12. [20]



Obrázok 11. Architektúra ARK platformy [20]

6.1 Požiadavky ARK na kamerový vstup

Sledovanie rojov Kilobotov sa vykonáva pomocou 4 kamier, ktoré sa spätne pripájajú do BCS na spracovanie. Počet kamier je zvolený ako minimum, ktoré dosiahne pokrytie veľkej arény 2,2×2,2 metra. To je pri montáži vo výške 160 cm nad povrchom arény. Je možné používať aj viac kamier ak je softvéru ARK upravený. V systéme ARK sú použité USB

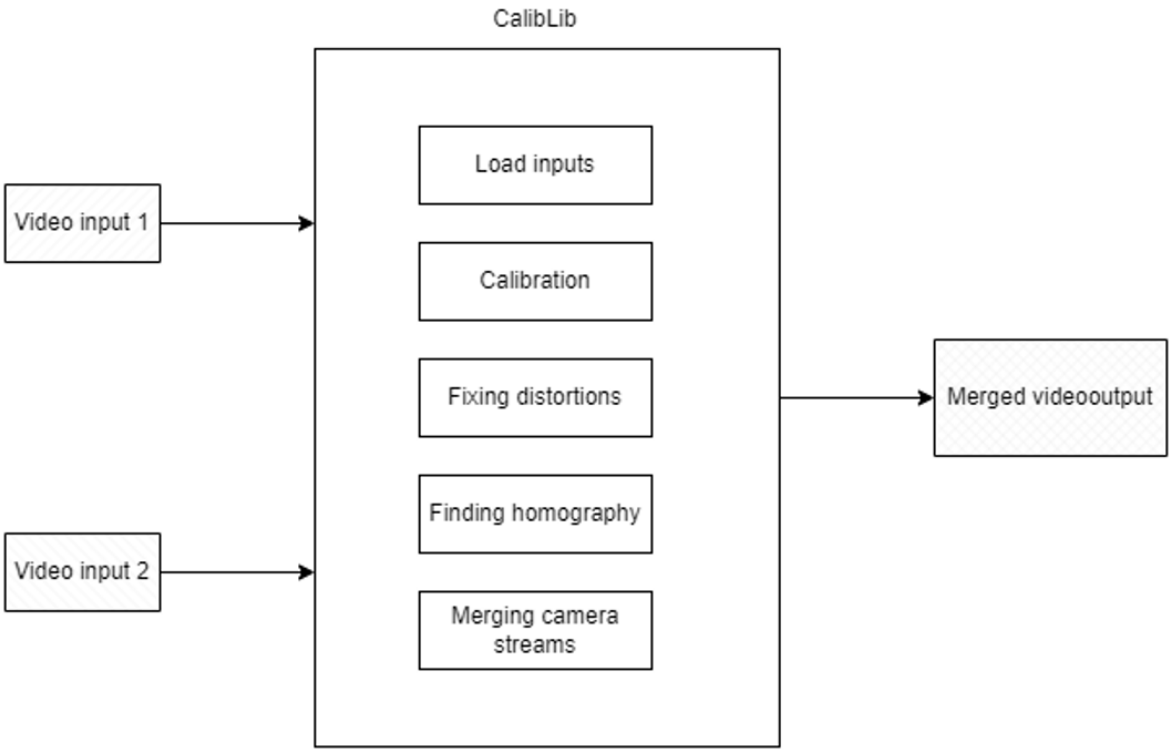
kamery e-CAM51 s maximálnym rozlíšením 2592 × 1944 s rýchlosťou snímania maximálne 30 snímok za sekundu. Avšak je možné použiť akúkoľvek kameru s porovnateľným rozlíšením a snímkovou frekvenciou, pokiaľ je kompatibilná so základovou stanicou. Hľadanie Kilobotov sa vykonáva pomocou Open CV a miesta nájdených Kilobotov sa používajú na vytvorenie inštancií triedy Kilobot. Získavanie obrazu z každej zo štyroch kamier pokrývajúcich arénu je riešené v samostatnom vlákne. Tieto vlákna používajú jednotlivé prúdy Open CV CUDA na deformáciu obrázkov a zmenu ich veľkosti pri príprave na zošívanie. To umožňuje súbežné vykonávanie. Deformácia využíva parametre určené samostatným programom, ktorý vykonáva automatickú kalibráciu na štyroch kamerových snímkach, aby ich transformoval do jedného pohľadu pomocou modulu na zošívanie panorámy Open CV.

Výkon v reálnom čase umožňuje použitie vlákien na umožnenie chodu na viacerých jadrách CPU a použitie knižníc GPU Open CV CUDA na spracovanie obrazu na zošívanie obrazu, sledovanie a farbu LED. GUI je vytvorené pomocou Qt Library Framework. Pretože Kiloboty sú lacné, zjednodušené roboty, sú vybavené iba jedným senzorom schopným detekovať okolité svetlo. Povolenie virtuálnych prostredí, a teda aj virtuálnych senzorov, otvára široké spektrum uskutočniteľných experimentov, ktoré predtým nebolo možné realizovať iba pomocou hardvéru Kilobot. ARK umožňuje implementáciu jedného virtuálneho snímača s vysokým rozlíšením alebo viacerých snímačov s nižším rozlíšením. Voľba je daná obmedzeným počtom bitov dostupných v každej správe. [20]

II. PRAKTICKÁ ČASŤ

7 NÁVRH KNIŽNICE

Zlučovanie dvoch a viacerých obrázkov, v jednoduchosti je proces ktorého cieľom je nájsť spoločné body týchto obrázkov, miesta na ktorých sa prekrývajú, tento vzťah matematicky vyjadriť a na základe toho, pomocou algoritmu tieto obrázky spojiť tak, aby tvorili jeden súvislý celok. Tento postup sa bežne využíva napríklad pri zlučovaní obrázkov pri vytváraní panoramatických snímok. Tento proces je pri návrhu zhrnutý do 5 základných bodov.



Obrázok 12. Návrh postupu zlúčenia obrazov

7.1.1 Vstupy

Prvým krokom je identifikovať aké sú vstupy do programu. To sú v tomto prípade 2 štandardné webové kamery. Program však nie je obmedzený iba na vstup z webových kamier, vie prijať vstup z akéhokoľvek snímacieho zariadenia pripojeného k počítaču. Obraz z týchto vstupov by mal snímať konkrétnu scénu, ako napríklad arénu s Kilobotmi, alebo akúkoľvek inú plochu. Je ideálne aby snímaná plocha bola rovina, kolmo otočená smerom ku strednému bodu medzi kamerami. Kamery túto plochu snímajú z rôznych uhlov, avšak určitú plochu snímajú naraz. To je miesto v ktorom sa obraz prekrýva a je podstatný na to aby sme na základe neho obraz spojili.

Do knižnice sa posiela živý prenos obrazu z týchto kamier, ak pri prenose nastane chyba, proces je potrebné opakovať. Dôležité je, hlavne pri prvotnej práci s programom, mať fyzickú verziu rovinného útvaru s jednoducho rozpoznateľným vzorom, napríklad čierno biela šachovnica. Je veľmi dôležité aby bol tento vzor na pevnom povrchu, a nebol žiadnym spôsobom deformovaný, v opačnom prípade by takáto chyba spôsobila že kalibrácia by skreslenie neodstránila, ale ešte viac ho zvýraznila.

7.1.2 Kalibrácia

Obraz každej kamery, v dôsledku konštrukcie šošoviek, je nejakým spôsobom skreslený. Pred pokusom zlúčiť obrazy týchto kamier, je potrebné toto skreslenie odstrániť, aby bol obraz prirodzenejší a aby program vedel lepšie rozoznať body, ktoré majú oba obrazy spoločné. Forma kalibrácie v tomto programe je založená na Zhangovom algoritme, tu bude dôležitá kalibračná šachovnica, spomínaná vyššie.

Cieľom tohto bodu je pripraviť a exportovať znovu použiteľný súbor s parametrami, ktoré budú ďalšie funkcie používať pri oprave skreslení. Túto kalibráciu teda prevedieme pri prvom spustení aplikácie, ďalej tento bod nie je nutné opakovať. Takúto kalibráciu je potrebné opakovať len pri zmene používanej kamery, alebo daného obrazového vstupu. Táto funkcia programu je voliteľná, pokiaľ je skreslenie vstupných obrazov už vyriešená externe, je možné tento bod vynechať.

7.1.3 Oprava skreslení

Výstupom kalibrácie sú súbory, ktoré obsahujú matice vnútorných a vonkajších parametrov kamier. Open CV ponúka funkcie, ktoré už opravu skreslenia obrazu riešia, jediné čo na opravu potrebujú sú práve parametre z vygenerovaných súborov. Týmto spôsobom vieme každý záznam z kamier opraviť hneď pri ich načítaní a ďalej používať opravenú verziu obrazového vstupu. Tento medzi krok nie je úplne nutný, aby zlúčenie vstupných obrazov prebehlo správne, napríklad ak už boli kamery pred napojením do aplikácie skalibrované. V opačnom prípade je oprava skreslení nevyhnutná pri ďalšej práci s obrazovom.

7.1.4 Nájdenie spoločných bodov

Ďalším krokom po oprave skreslení na kamerových vstupoch, je zistiť vzájomný vzťah medzi danými obrazmi z kamier. Zlúčenie obrazu bude fungovať iba v prípade, že obraz z kamier sa bude v určitých miestach prekrývať. Open CV obsahuje funkcie, ktoré túto podobnosť obrazov dokážu identifikovať. Cieľom tejto časti je zistiť vzťah medzi týmito

obrazmi, túto spojitosť zaznamenať a vygenerovať súbor, ktorý túto spojitosť matematicky popisuje. Výsledný súbor bude znovu možné exportovať/importovať a opakovane používať. Túto funkciu je potrebné zavolať vždy, keď chceme prvotne zistiť spojitosť medzi kamerami, alebo keď sa zmení poloha a uhol kamier.

7.1.5 Spojenie obrazu

Ak načítanie vstupného obrazu prebehne správne, sú opravené skreslenia kamier a poznáme spojitosť medzi obrazmi, môžeme tieto obrazy spojiť do jedného. Spojenie prebieha spôsobom, že jeden zo vstupných obrazov je použitý ako hlavný a obraz z druhej kamery sa naň pomocou príslušných funkcií namapuje. Ide teda o perspektívne skreslenie druhého obrazového vstupu tak, aby bol perspektívne zhodný s prvým obrazovým vstupom. Ďalej je potrebné vytvoriť väčšie „okno“ tak aby sa tieto oba obrazy, hlavný a perspektívne upravený, do neho vošli a dali sa ďalej premietať. Do tohto väčšieho okna vložíme prvý hlavný obraz, a prekryjeme ho druhým upraveným obraz v mieste, ktoré je pre oba obrazy spoločné. Tým sa plocha snímaná dvomi kamerami premietne do jedného výstupu tak, aby bol prechod medzi obrazmi minimálne viditeľný. V tomto bode máme obrazy z viacerých kamier zlúčené do jedného obrazu, s ktorým sa dá ďalej pracovať.

7.2 Účel knižnice

Účel knižnice je získať obraz z dvoch alebo viacerých kamier, upraviť ho a tieto obrazy spojiť do jedného obrazu. Táto knižnica je primárne určená na použitie v Kilobot aréne ARK, dá sa však použiť na spracovanie akéhokoľvek podobného systému kde sa vyskytujú dve kamery snímajúce rovnakú plochu z rôznych uhlov. Knižnica je vytvorená ako trieda v jazyku C++. Hlavné zásady pri vytváraní knižnice sú:

7.2.1 Univerzálnosť vstupov

Vstup do programu nemusí byť nutne obraz z webovej kamery. Cieľ je aby sa dal použiť hocijaký video formát, napríklad kamera telefónu, alebo iné snímacie zariadenie. Pri spustení aplikácie je možné špecifikovať o aký vstup ide.

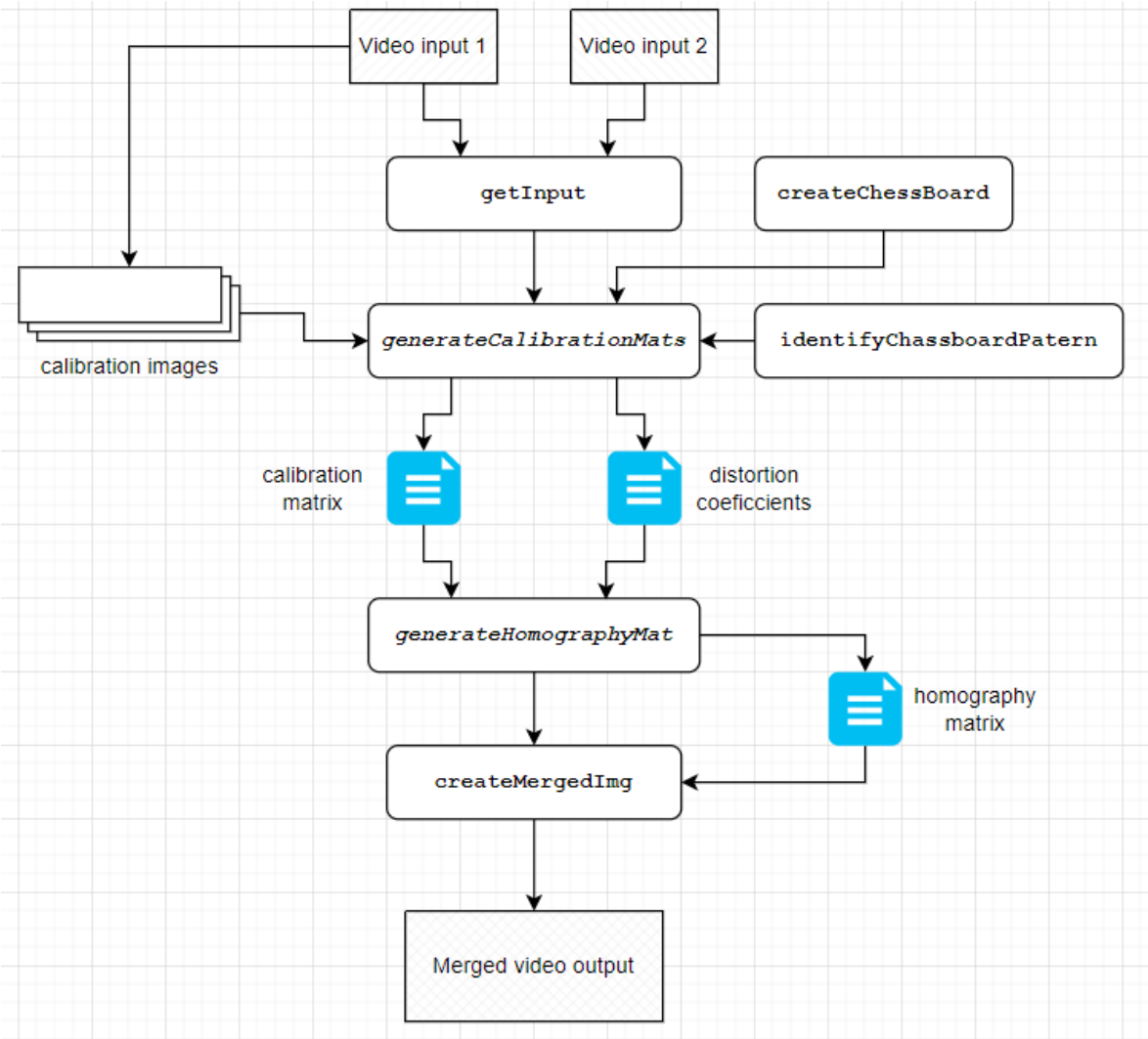
7.2.2 Znovu použiteľnosť

Dôležitou súčasťou programu je, aby pri prevedení výpočtov na úpravu a spojenie obrazov, nebolo potrebné tieto výpočty opakovať pri každom spustení aplikácie. Kalibráciou obrazu sa vygeneruje súbor, ktorý slúži na opravu obrazu z kamier a pokiaľ sa počas používania

nezmení typ kamery, je tento súbor možné opakovane importovať a používať. Zistením homografie medzi obrazmi sa vygeneruje súbor, ktorý zase udáva vzťah medzi obrazmi. Ten je možné importovať a používať, dokiaľ sa nezmení poloha kamier alebo uhol akým snímajú sledovanú plochu. Ak pri prvom spustení aplikácie použitím knižnice vytvoríme spojený obraz a celý set kamier a snímanej plochy ostane rovnaký bez zmeny, stačí aplikáciu kedykoľvek spustiť a načítaním všetkých potrebných údajov z existujúcich súborov hneď získať spojený obraz, tak ako bol vytvorený pri prvom spustení.

8 IMPLEMENTÁCIA KNIŽNICE

Knižnica obsahuje funkcie, ktoré na seba nadväzujú. Tieto funkcie slúžia na manipuláciu s obrazovými vstupmi, a tiež sa pomocou nich volajú metódy knižnice Open CV. Určité premenné, ktoré sa počas programu používajú, sú využívané opakovane na viacerých miestach, sú preto zdieľané v celej knižnici. Ide napríklad o rozmery kalibračného vzoru, kalibračné matice, alebo práve zachytené snímky z kamier, ku ktorým pristupuje viac funkcií naraz.



Obrázok 13. Architektúra knihovne

8.1 Časť načítania vstupov

Načítanie obrazu pomocou knižnice Open CV funguje spôsobom, že objekt *cv::VideoCapture* pomocou špecifikácie cesty ku kamere v konštruktore, alebo pomocou

funkcie *cv::read* prečíta danú snímku z video streamu. Aby sme túto snímku mohli používať a spracovávať, je potrebné ju uložiť do premennej čo je v našom prípade *cv::Mat.* Pri ukladaní je možné špecifikovať rozmery, farebné kanály, aj formát snímky. Na zobrazenie snímky v samostatnom okne sa v Open CV používa funkcia *cv::imshow,* ktorej parametrami sú názov okna a premenná, v ktorej je snímka uložená.

Týmto spôsobom však program zachytí len jednu snímku v čase, kedy je funkcia volaná. Preto musí byť funkcia ukladania snímky zo streamu do premennej umiestnená v nekonečnom cykle, počas ktorého sa bude volať a tak sa budú snímky z videového vstupu plynule načítavať. Po tom ako snímku uložíme, môžeme ju hneď aj spätne zobraziť. Takýto prístup má však nevýhodu. Pretože cyklus na zachytávanie je väčšinou nekonečný, napríklad formátu *while(true),* program nemôže vykonávať inú funkcionalitu dokiaľ cyklus manuálne neukončíme, alebo nenastane chyba pri uložení snímky, kedy sa cyklus ukončí. Na to je v každom cykle v programe podmienka, v ktorej sa zisťuje či užívateľ stlačil príslušný znak na klávesnici, a tým sa ukončí cyklus a aj funkcia, v ktorej sa načítanie vstupu z kamery vykonáva.

Jedným riešením tohto problému je použiť v C++ vlákna. *std::thread* umožní programu funkciu za snímanie obrazu spustiť v samostatnom vlákne, čím oddelí tento proces od zvyšku programu. Tak je možné pracovať s programom a zároveň získavať vstup z kamier.

```
void CalibLib::showCameraInput(int i, string w)
{
    namedWindow(w,WINDOW_NORMAL);
    Mat frame;
    VideoCapture video(0);
    string w = „window 1“;
    while (true){
        video >> frame;
        if(frame.empty())
            return;
        imshow(w,frame);
        if(waitKey(20) == 27){
            destroyWindow(w);
            return;
        }
    }
}
```

8.2 Časť kalibrácie

8.2.1 createChessBoard

Vstupom do tejto funkcie je dátový typ knižnice Open CV *cv::Size* ktorý obsahuje veľkosť kalibračnej šachovnice. Prednastavené hodnoty sú 6 x 9 štvorcov. Druhý parameter je fyzická dĺžka hrany jedného štvorca na šachovnici v milimetroch. Hodnoty týchto parametrov je dôležité nastaviť podľa rozmerov reálne používanej šachovnice, inak môže nastať problém pri výpočtoch neskôr v programe, ktoré by skreslenie po kalibrácií mohli naopak ešte zhoršiť. Účelom tejto funkcie je vytvoriť programovú reprezentáciu šachovnice, čo je v tomto prípade vektor bodov typu *cv::Point3f* ktorý má rozmery rovnaké ako reálna šachovnica. Tento zoznam bodov bude neskôr potrebný pri kalibrácií.

8.2.2 identifyChassboardPatern

Aby sme vedeli kamery správne skalibrovať, je dôležité aby obrázky kalibračného vzoru, v našom prípade šachovnice, ktoré predložíme programu boli kvalitné a dobre rozpoznateľné. Preto je dobré mať možnosť vidieť či, ak sa daný kalibračný vzor na video vstupe ukáže, dokáže ho program rozoznať. Tu je použitá pomocná funkcia knižnice Open CV *cv::findChessboardCorners*, ktorá zo vstupného obrazu dokáže extrahovať daný vzor. Ak je vzor rozoznaný uloží výstup do premennej, ktorá je vektor bodov *cv::Point2f*.

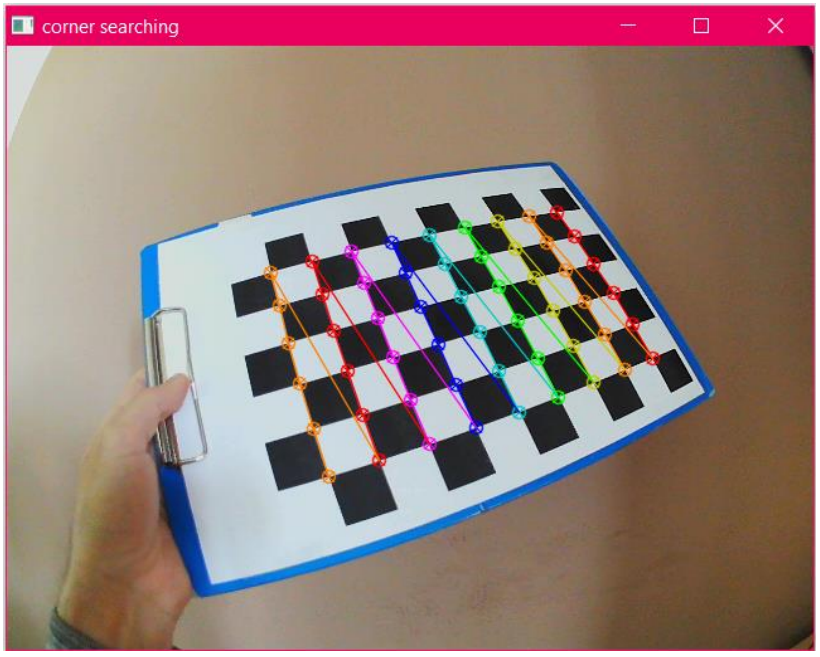
Na to slúži táto funkcia, ktorá má dve varianty. Prvý variant slúži na zachytenie vzoru z obrazového vstupu v reálnom čase. Vstupný parameter cv::Mat je v tomto prípade aktuálna snímka z video vstupu. Túto funkciu je potrebné volať vo *While,* alebo *For* cykle. S každou novou snímkou videa sa rozpoznaný vzor uloží nanovo, a prednastavenou funkciu *cv:: drawChessboardCorners* tento vzor spätne vykreslí na práve zobrazovanú snímku videa. Týmto spôsobom vieme, či program rozoznáva daný vzor, ako je to na obrázku 13 . Ak sa vzor spätne nevykreslí, program ho na snímke neidentifikoval správne. To môže byť spôsobené nedostatočným osvetlením scény, slabým rozlíšením kamery, alebo príliš veľkým odklonením vzoru šachovnice od kamery.

```
void CalibLib::identifyChassboardPatern(Mat img, vector<vector<Point2f>>
cornersAll)
{
    vector<Point2f> points;
    bool found = findChessboardCorners(img,
        chessboardSize,
        points,CALIB_CB_ADAPTIVE_THRESH |
        CALIB_CB_NORMALIZE_IMAGE |
        CALIB_CB_FAST_CHECK);
```

```
    if(found)
        cornersAll.push_back(points);

    drawChessboardCorners(img, chessboardSize, points, found);
    imshow("corner searching",img);
}
```

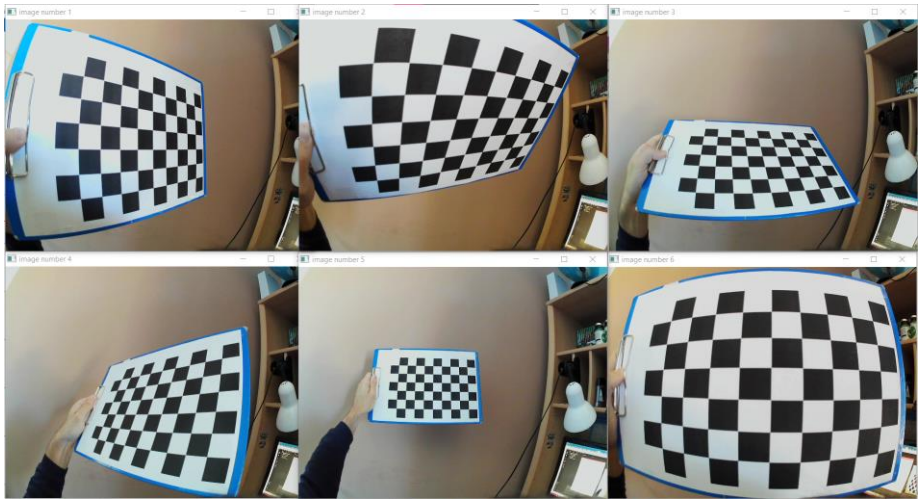
Druhý variant, ktorý je preťažením funkcie je určený na vnútorné použitie pri konkrétnej kalibrácií. Vstupom tu už nie je len daná snímka, ale zoznam snímok, na ktorých je kalibračný vzor jasne rozpoznateľný. Funkcia je upravená tak, aby v cykle prešla každou snímku zo zoznamu, extrahovala vzor, a tieto vzory uložila do zoznamu. V tomto bode máme zo strany obrazového vstupu všetko potrebné pre kalibrovanie.



Obrázok 14. Realtime detekcia a vykreslenie kalibračného vzoru

8.2.3 saveImg

Táto funkcia je spôsob ako vytvoriť zoznam snímok, ktoré budú odoslané na spracovanie do kalibračnej funkcie. Funkciu je ideálne volať počas toho ako program pri snímaní spätne vykresľuje šachovnicový vzor. Maximálny počet snímok nie je obmedzený, ideálne však je mať minimálne 10 snímok. Pri menšom počte sa môže stať, že program nebude mať pri výpočtoch dostatočné množstvo dát. Je preto potrebné aby kalibračný vzor, teda šachovnica bola na snímkach zachytená z čo najviac uhlov a vzdialeností. Ukážka na Obrázku 16.



Obrázok 15. Ukladanie snímok s kalibračným vzorcom

8.2.4 generateCalibrationMats

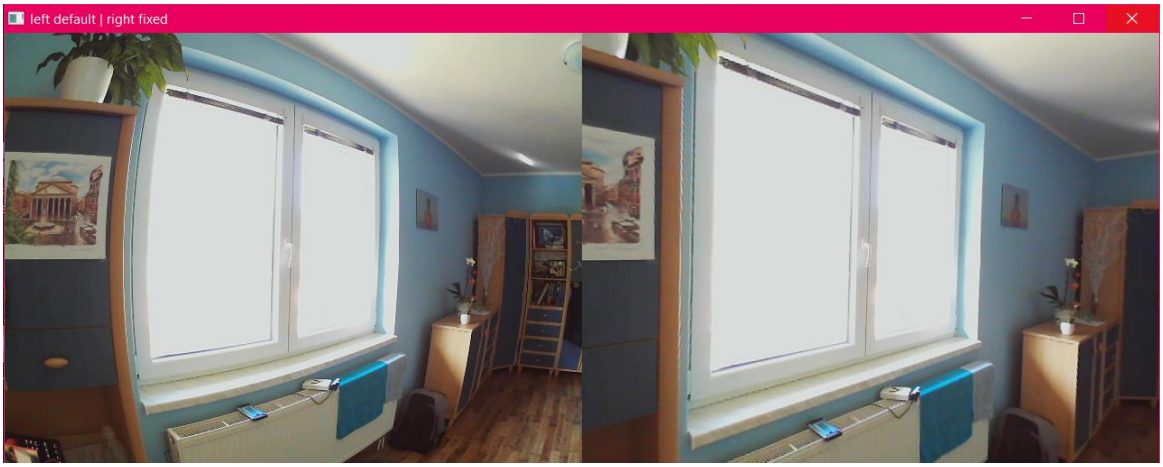
Vstupom do tejto funkcie sa predpokladá, že kamery majú určité skreslenie, ktoré treba odstrániť. Cieľom je vytvoriť dve matice, z ktorých jedna je 3x3 kalibračná matica dátového typu *cv::Mat*. Druhá potrebná matica je zoznam koeficientov, ktoré udávajú vnútorné parametre danej kamery. Open CV poskytuje niekoľko algoritmov, ktoré nám pomáhajú vypočítať tieto vnútorné parametre. Skutočná kalibrácia sa vykoná pomocou Open CV funkcie *cv::calibrateCamera()*. Vstupom do funkcie sú aj referencie na premenné, do ktorých sa uložia kalibračné matice. Na základe zvolených rozmerov šachovnice si funkcia túto šachovnicu vytvorí pomocou metódy *createChessBoard() a* následne si funkcia zo snímok vyextrahuje body na základe zvoleného vzoru, čím dostaneme štruktúru, ktorá má veľa individuálnych a identifikovateľných bodov. Ukážka na kóde nižšie.

```
vector<Point3f> CalibLib::createChessBoard(Size boardS, double squareS)
{
    vector<Point3f> c;
    for (int i = 0; i < boardS.height; i++) {
        for (int j = 0; j < boardS.width; j++) {
            c.push_back(Point3f(j * squareS,i * squareS, 0.0f));
        }
    }
    return c;
}
```

Pri pohľade na túto štruktúru z rôznych uhlov a porovnaní s vytvorenou šachovnicou môžeme potom vypočítať (relatívnu) polohu a orientáciu kamery v čase každého obrázka, ako aj vnútorné parametre kamery. Vygenerovaná kalibračná matica a koeficienty skreslenia

sú následne uložené do príslušných súborov. Tým sa zabezpečí, že funkciu nebude potrebné volať pri každom spustení aplikácie a kalibrovať celú zostavu nanovo, ale jednoducho sa matice načítajú zo súborov a budú pripravené na použitie. Matice, ktoré týmto spôsobom dostaneme sú naviazané na konkrétnu kameru, preto pri zmene zariadení na snímanie obrazu je nutné takúto kalibráciu zopakovať, pretože pôvodné matice by nefungovali správne.

Reálne odstránenie skreslenia kamerového záznamu sa najjednoduchšie aplikuje funkciou *cv::Undistort,* ktorá použitím kalibračných matíc a algoritmov skreslenia odstraňuje a vracia späť opravenú snímku, viď Obrázok 17. Pri živom snímaní obrazu kamerou, je táto funkcia volaná po uložení danej snímky, čím je snímka opravená a pripravená na ďalšie spracovanie.



Obrázok 16. Oprava skreslení obrazu, originálna snímka vľavo, upravená vpravo

8.3 Časť zlučovania obrazu

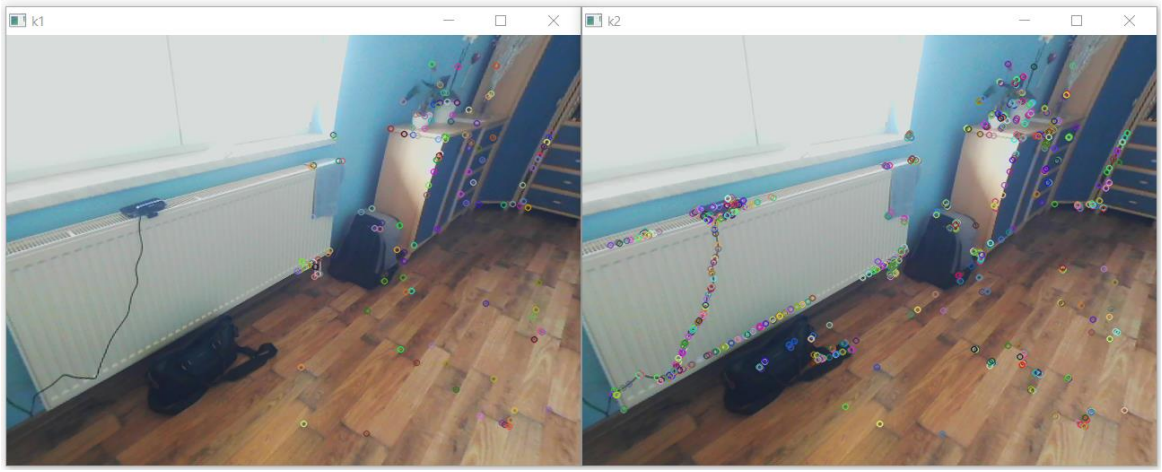
Táto časť sa stará už priamo o vytvorenie mozaiky zo snímok, ktoré sú na vstupe. Funkcie je možné volať úplne samostatne hneď po spustení aplikácie, ak už sú vygenerované matice na odstránenie skreslenia.

8.3.1 generateHomographyMat

Na spojenie dvoch akýchkoľvek obrazov potrebujeme najprv zistiť či majú nejakú časť spoločnú. Ak sa takýmto spôsobom zhodujú, existuje medzi nimi spojitosť. Úlohou tejto funkcie je túto spojitosť matematicky popísať a uložiť do formátu, s ktorým vieme pracovať. Vstupom do funkcie sú referencie na premenné typu *cv::Mat*, ktoré reprezentujú snímky z kamier. Základom výpočtu matice homografie, ktorá udáva vzťah medzi snímkami, je

identifikovať kľúčové body spoločné pre obe snímky. Prvým krokom je vytvoriť objekt na detekciu kľúčových bodov. Pre tento projekt bol zvolený detektor kľúčových bodov ORB**,** ktorý sa z dostupných algoritmov javil ako najideálnejší, z hľadiska rýchlosti aj kvality detekcie. Tieto detektory sú potrebné dva, pre každú snímku jeden. Predvolený počet vyhľadávaných bodov je 5000, aby mal program väčšiu šancu nájsť kľúčové body aj na menej rozpoznateľnej ploche. Ďalej potrebujeme dva typy zoznamov. Prvý určuje kam sa tieto detekované kľúčové body budú ukladať. Open CV ma pre tento prípad dátovú štruktúru *vector<cv::KeyPoint>*. Tento bod je charakterizovaný svojimi 2D súradnicami, veľkosťou vzhľadom ku svojmu okoliu a orientáciou. Druhý zoznam je typu *vector<cv::Dmatch>* kam sa uložia tie kľúčové body, ktoré sú úspešne rozoznané, a zároveň spárované, čo znamená že ich majú spoločné obe snímky.

Keďže sa predpokladá, že snímky sa budú prekrývať nie na celej ploche, je pri detegovaní kľúčových bodov možné nastaviť vyhľadávanie len v určitej časti snímok. Toto sa robí vytvorením masky, ktorá sa aplikuje na snímky ešte pred výpočtami. Funkcia knižnice Open CV *cv::Rect* po zadaní rozmerov požadovaných rozmerov vytvorí štvorec, do ktorého sa dajú ukladať časti snímky. Aby sme z tohto štvorca dostali masku, vložíme doň štruktúru typu *cv::Mat,* ktorá je naplnená nulami, a má formát *CV_8UC1,* čo je označenie pre 8 bitové jedno-kanálové pole. Takto „zakryjeme“ časť snímky aby detektor kľúčových bodov v tejto oblasti body nehľadal. Vyhľadávanie je tým pádom rýchlejšie aj efektívnejšie. Obrázok 18 zobrazuje snímku vľavo s aplikovanou maskou a snímku vpravo bez masky.



Obrázok 17. Obmedzenie detekcie len na polovicu snímky

O proces detegovania kľúčových bodov sa stará funkcia detektora *cv::detectAndCompute,* ktorá do zoznamov uloží body ktoré sa podarilo detegovať. O spárovanie nájdených kľúčových bodov sa stará deskriptor triedy *cv::DescriptorMatcher*. Tento algoritmus porovná všetky body z jednej snímky so všetkými bodmi druhej snímky a nájde najlepšiu zhodu. Takýto zoznam párov však môže obsahovať aj také páry, ktoré ale nie sú priradené správne. Na to je v kóde cyklus, ktorý tieto body uloží do zoznamu. Následne je tento zoznam zaradzovaný aby najpresnejšie páry boli na vrchu zoznamu.

Funkciou *cv::drawMatches* môžeme spätne tieto páry kľúčových bodov vykresliť na zobrazovanú snímku, aby sme mali prehľad ako sa detektoru darí tieto body hľadať a párovať. To zobrazuje Obrázok 19.

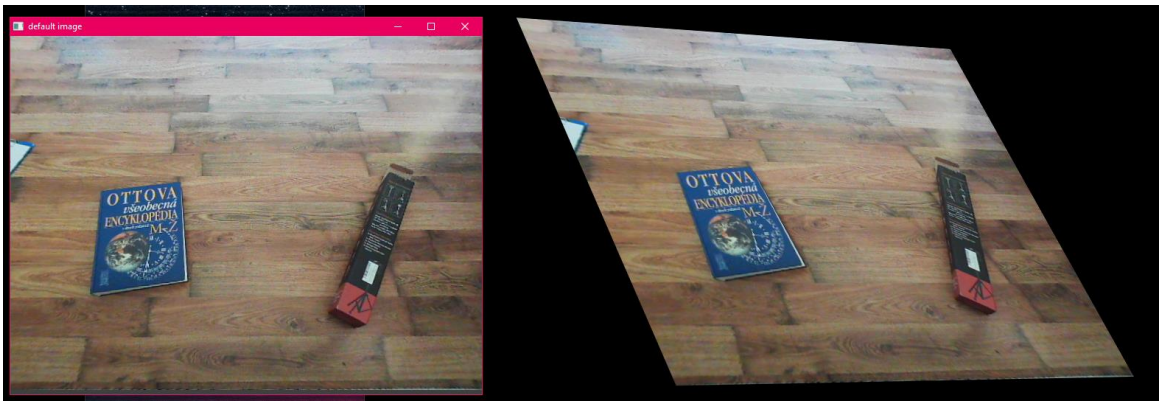
Obrázok 18. Kreslenie spárovaných bodov naprieč snímkami.

Ak je zoznam zoradený, kľúčové body z týchto párov sú priradené k daným snímkam. Poslednou časťou metódy je generovanie matice homografie medzi snímkami. Funkcia *cv::findHomography* vykoná všetky potrebné výpočty, na základe týchto získaných bodov. Využíva pri tom metódu RANSAC [21]. Takto získaná matica popisuje spôsob akým sa bude jedna snímka mapovať na druhú.

8.3.2 createMergedImg

Cieľom tejto metódy je využitím všetkých matíc vygenerovaných predošlými funkciami upraviť snímaný obraz kamier tak, aby sa viditeľne zlúčil do jedného súvislého celku. V tomto bode musí kalibračná matica, koeficienty skreslenia aj matica homografie byť načítaná v systéme. Návratový typ funkcie je *cv::Mat*, ktorá obsahuje finálnu zlúčenú

snímku. Po overení, či je vstup z kamier načítaný správne, má užívateľ možnosť zvoliť či má byť vstupný obraz z kamier upravený pomocou kalibračných matíc, aby sa odstránilo prípadné skreslenie. Proces zlúčenia prebieha v dvoch krokoch. Prvým krokom je výber snímky, ktorá bude hlavná, a ku nej v druhom kroku pripojiť druhú snímku. Počíta sa s tým, že každá kamera má iný uhoľ pohľadu, a teda aj perspektívne skreslenie každej kamery je rozdielne. Druhú snímku preto musíme transformovať, aby jej perspektívne skreslenie bolo rovnaké ako pri hlavnej snímke. To je vidieť na Obrázku 20.



Obrázok 19. Pôvodná snímka naľavo a transformovaná napravo

Takáto transformácia sa robí funkciou *cv::warpPerspective.* Vložením druhej snímky, inverznej matice homografie a určením veľkosti výstupnej snímky, funkcia druhú snímku transformuje. Následne ju vloží do väčšieho rámu, pretože veľkosť snímky sa v dôsledku transformácie môže zväčšiť a niektoré časti obrazu by sa mohli v procese stratiť. Druhým krokom je vytvoriť, v tejto výslednej snímke, plochu do ktorej nakopírujeme hlavnú snímku. Pomocou *cv::Rect* vytvoríme oblasť, ktorá bude mať rozmery hlavnej snímky. Kód nižšie zobrazuje transformáciu a prekrytie obrazu.

```
warpPerspective(secondFrame, result, homographyMAT.inv(),
                    Size(secondFrame.cols + firstFrame.cols,
                         secondFrame.rows + firstFrame.rows/2));
Mat firstFrame, secondFrame, result;
Mat cut(result, Rect(0, firstFrame.rows/6,
                     firstFrame.cols,firstFrame.rows));
firstFrame.copyTo(cut);
return result;
```

Keďže obe snímky sú v tomto bode perspektívne skreslené, snímky sa prekryjú v bodoch, ktoré sú rovnaké pre obe snímky. Nakopírovanie hlavnej snímky do výslednej prebehne funkciou *cv::Mat.copy.* Obrázok 21 ukazuje zobrazenie každého záznamu zvlášť

a Obrázok 22 ukazuje už výsledné zlúčenie záznamov z kamier. Výsledná snímka je v tomto prípade hotová a pripravená na použitie.

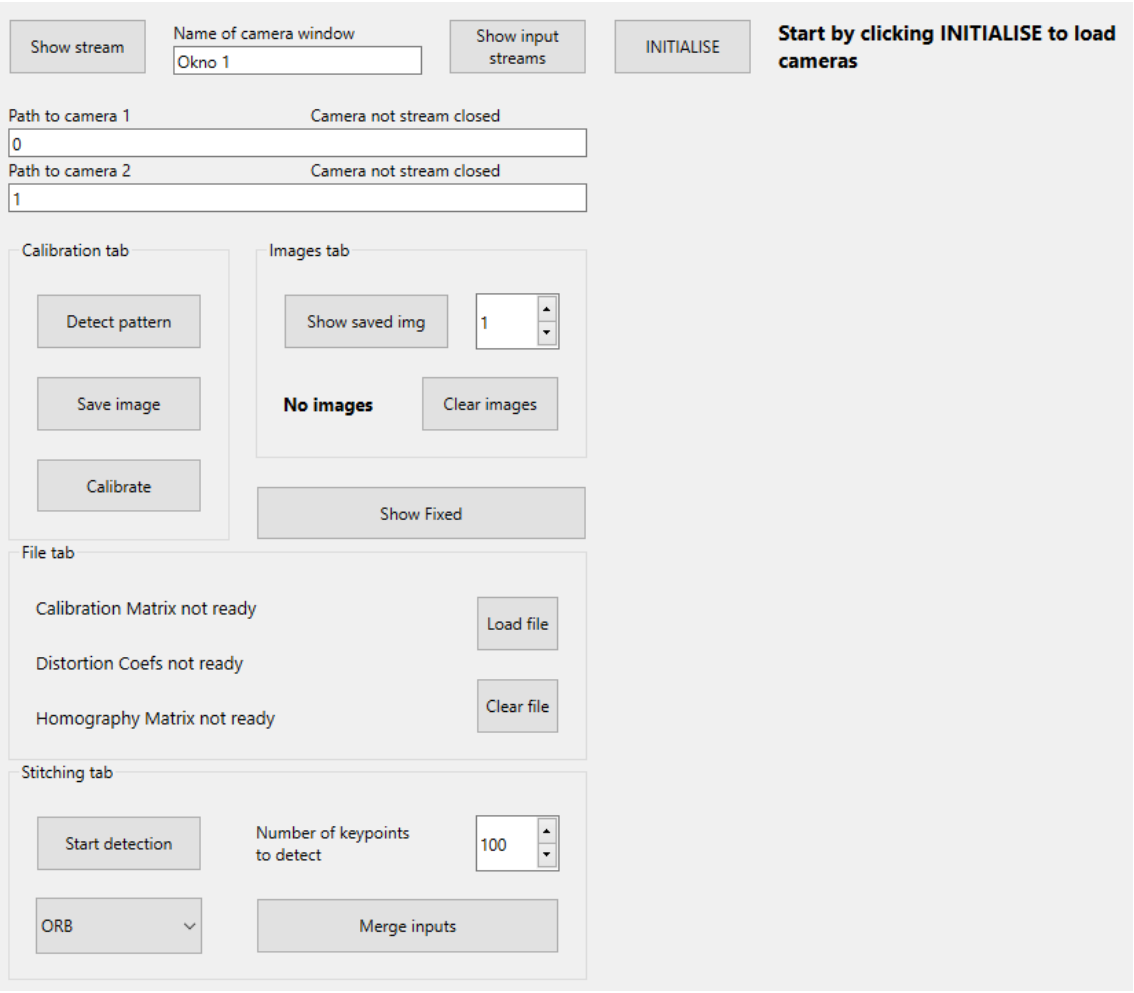
Obrázok 20. Individuálne záznamy z kamier

Obrázok 21. Výstup spojenia záznamov z kamier

9 APLIKÁCIA

9.1 Qt GUI

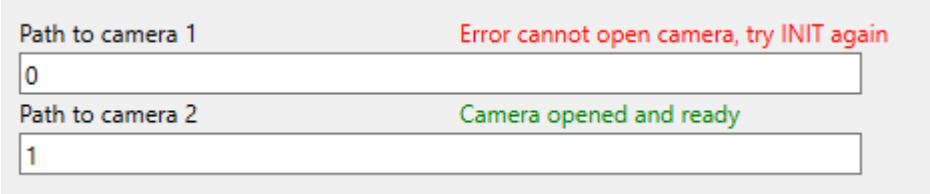
Aby sa dala knižnica na zlučovanie obrazu ľahšie používať, je ku nej vytvorená podporná aplikácia, ktorá všetky jej funkcie priamo využíva. Qt umožňuje jednoduchý návrh programu s používateľským rozhraním spustiteľným na Windows, Linux alebo Mac. Poskytuje aj výkonnú súpravu nástrojov používateľského rozhrania s názvom Qt Designer, ktorá umožňuje navrhovať používateľské rozhranie bez písania jediného riadku kódu a tiež umožňuje pokročilým používateľom prispôsobiť si komponenty používateľského rozhrania pomocou jednoduchého skriptovacieho jazyka nazývaného Qt Style Sheets. [22]



Obrázok 22. Menu aplikácie

9.1.1 Napojenie kamier, získanie obrazu

Aby sme mohli z kamier zachytávať obraz, potrebuje najprv otvoriť komunikačný kanál medzi kamerou a programom. Polia „Path to camera 1“ a „Path to camera 2“ slúžia na zadanie cesty ku danej kamere. Napríklad na notebooku je 0 predvolená webová kamera. Po zadaní ciest ku obom kamerám, tlačidlom „INITIALISE“ aktivujeme snímanie obrazu týmito kamerami. Na kontrolu, či sú kamery dostupné a pripravené je možné zobraziť snímanie tlačidlom „Show input streams“ alebo samostatne pre jednu kameru tlačidlom „Show stream“. Na Obrázku 23 je vpravo zobrazený textový stav kamier.



Obrázok 23. Pole pre pripojenie kamier

9.1.2 Detekcia vzoru a ukladanie snímok

Prvý bod pri procese kalibrácie je získať snímky, na ktorých je jednoznačne rozpoznateľný vzor šachovnice. Ideálny stav je získať 15 a viac obrázkov, z čo najviac uhlov a vzdialeností. Tlačidlom „Detect pattern“ sa aktivuje funkcia, ktorá sníma obraz zo zvolenej kamery a zároveň rozpoznáva šachovnicový vzor. Ak je vzor rozpoznaný, funkcia ho graficky vykreslí spätne do snímaného videa.

Počas tohto snímania stlačením tlačidla „Save image“ danú snímku uložíme do zoznamu obrázkov, ktorý je pripravený. Týmto spôsobom stačí šachovnicu pred kamerou polohovať a ukladať snímky dovtedy, kým nebudeme mať dostatok obrázkov zo všetkých potrebných uhlov. Tlačidlo „Show saved img“ slúži na zobrazenie konkrétneho obrázku, kde sa dá prekontrolovať, či má obrázok správnu kvalitu. V prípade, že bude potrebné kamery skalibrovať nanovo, alebo uložené obrázky nie sú dostatočne kvalitné, tlačidlom „Clear images“ zoznam obrázkov vymažeme. Týmto spôsobom sa zo systémových premenných odstránia už aj existujúce matice skreslenia. Obrázky sú zmazané aj pri ukončení aplikácie.

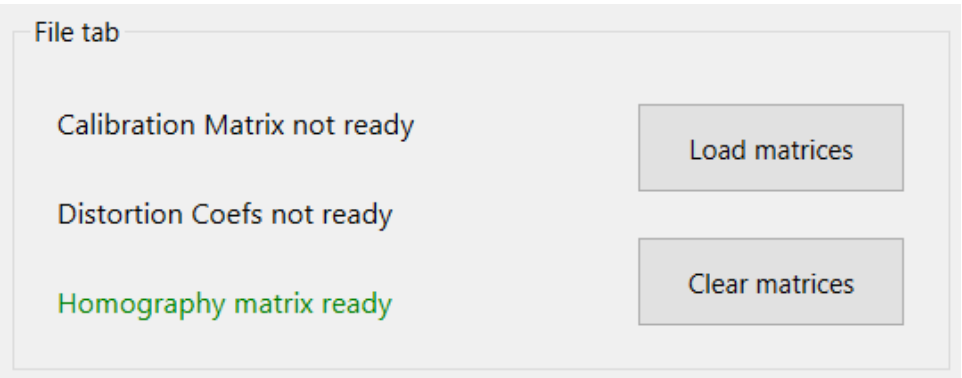
9.1.3 Kalibrácia

Ak máme dostatok obrázkov, program je pripravený na kalibráciu. Tlačidlom „Calibrate“ vyvoláme funkciu, ktorá tento proces riadi. Výsledkom tejto operácie je vygenerovanie kalibračných matíc, ich uloženie do systémových premenných, ako aj do samostatných súborov formátu XML. To či prebehla táto operácia úspešne sa dá overiť tlačidlom „Show Fixed“. Tým sa spustí prenos z kamery v originálnom formáte a zároveň ten istý prenos, na ktorý je aplikovaná úprava skreslenia. Ak sú obe pripojené kamery rovnakého typu, bude oprava skreslení fungovať rovnako na každej z nich.

Tým, že sú tieto matice znovu použiteľné, nebolo by efektívne kalibrovať obraz pri každom spustení aplikácie. Funkcia preto na konci svojho behu tieto matice uloží do samostatných súborov, aby sa pri ďalšom spustení aplikácie dali načítať do príslušných premenných a priamo používať. V prípade zmeny kamier je nutné proces kalibrácie vykonať nanovo, aby sa matice napasovali na parametre danej kamery.

9.1.4 Import/Export súborov

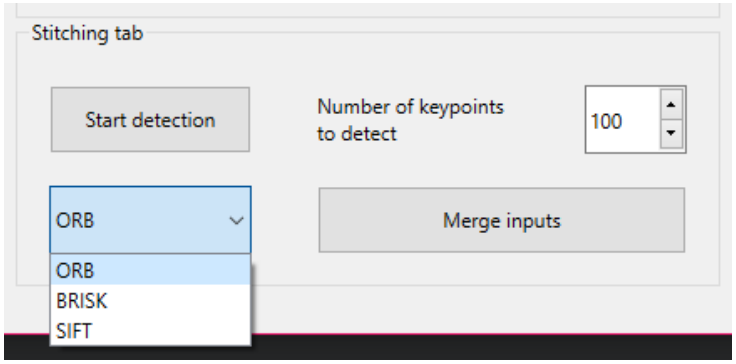
Časť „File tab“ slúži na kontrolu v akom stave sú kalibračné matice. Matica kalibrácie a parametre skreslenia sú vytvorené a uložené do systému aj súborov funkciou Calibrate. Ak takto súbory existujú v súborovom priečinku aplikácie, je možné ich načítať priamo pri spustení aplikácie tlačidlom „Load matrices“. V prípade, že bude potrebné kalibráciu alebo spojenie obrazu prepočítať nanovo, dajú sa tieto matice vynulovať tlačidlom „Clear matrices“.



Obrázok 24. Načítanie parametrov zo súborov.

9.1.5 Stitching tab

V tejto časti tlačidlo „Start detection“ spustí premietanie z oboch kamier. Zároveň sa pomocou funkcie *generateHomographyMat* spätne na záznam vykresľujú spárované kľúčové body. Počas tohto snímania je dôležité systému poskytnúť čo najviac referencií, z ktorých môže detegovať kľúčové body. V číselnej kolónke je možné zadať aké množstvo spárovaných kľúčových bodov má funkcia na výpočet matice homografie použiť. Funkcia vyžaduje minimálne 4 body aby výpočet fungoval. Algoritmus automaticky zaradzuje páry bodov zostupne podľa najlepších zhôd. Vždy budú teda pri výpočte použité tie najvhodnejšie páry. Na Obrázku 25 sú vidieť na výber 3 algoritmy na vyhľadávanie kľúčových bodov.



Obrázok 25. Výber deskriptora.

Stlačením klávesy ESC sa snímanie bodov ukončí a funkciou je vrátená matica homografie vypočítaná z danej snímky. Funkcia tiež vráti na ukážku výslednú snímku už s upraveným spojeným obrazom. Môže sa stať, že spracovanie homografie nie je úplne správne, čo môže nastať, keď nie je upravené skreslenie kamier, alebo ak je plocha snímaná každou kamerou príliš odlišná. Preto je občas potrebné aktivovať detekciu viac krát, dokiaľ výsledok spojenia obrazu nie je ideálny. Snímaný obraz by mal v ideálnom prípade byť voľná plocha, aby sa predišlo vytvoreniu mŕtveho uhlu medzi spojenými zábermi. Čo je oblasť medzi kamerami mimo ich zorného poľa. Tlačidlo „Merge inputs“ následne spustí premietanie tohto spojeného obrazu.

ZÁVER

Účelom tejto bakalárskej práce bolo vytvoriť program, ktorý dokáže zjednodušiť používanie systému, kde sa sníma jedna plocha viacerými kamerami. To znamená spojiť obraz dvoch kamier do jedného obrazu a premietať tento obraz v reálnom čase. Väčšina algoritmov na tvorbu panorámy alebo mozaiky iba jednorazovo spojí snímky do jedeného veľkého obrazu, alebo naopak zložitejšie algoritmy dynamicky spracovávajú a počítajú vzťah medzi zábermi neustále pred každým vykreslením. Pokiaľ by sa stalo, že takýto program neidentifikuje spojitosť medzi snímkami, algoritmus zlyhá a tým program zlyhá. V práci som chcel dosiahnuť, aby sa týmto prípadom predišlo, čo znamená, aby bolo výpočty potrebné urobiť iba na začiatku, a ďalej len na základe nich spojený obraz premietať.

Cieľom teoretickej časti bolo popísať aké javy ovplyvňujú snímanie obrazu bežnou kamerou. Sú tu vysvetlené základné konštrukčné nedokonalosti kamier a aj postupy ako tieto nedokonalosti matematicky popísať a pomocou algoritmov počítačového videnia opraviť. Taktiež je popísaný spôsob akým sa dá zistiť vzťah medzi kamerami, ktoré snímajú rovnakú plochu, na základe čoho je možné modelovať algoritmy na 3D rekonštrukciu, alebo tvorbu panorámy.

Praktická časť je zameraná na popis a návrh procesu akým spôsobom sa dá statické spojenie kamerového obrazu riešiť. Následne je vysvetlené fungovanie C++ knižnice a jej funkcií, ktoré boli vytvorené k tomuto účelu. Aby bolo použitie knižnice jednoduchšie, je vytvorená aj samostatná aplikácia s užívateľským rozhraním, za pomoci ktoérho sa dá proces riadiť užívateľom. Dokumentácia zahŕňa použitie a vysvetlenie jednotlivých elementov aplikácie. Aplikácia je navrhnutá na opakované použitie, po vytvorení kalibračných súborov, ich pri opätovnom spustení stačí načítať a spustiť premietanie spojeného obrazu. Súčasťou projektu je open source knižnica počítačového videnia Open CV, ktorá ponúka obrovské množstvo algoritmov z oblasti počítačového videnia a strojového učenia. Knižnica aj aplikácia bola testovaná s pomocou dvoch webový kamier ACR010 QHD Webcam. Funkcionalita programu nie je limitovaná iba na použitie v ARK platforme, ale zlúčený obraz z kamier je možné premietať s ľubovoľnými kamerami, ktoré sú kompatibilné so systémom, napríklad kamera telefónu alebo tabletu, ktorá bola pri testovaní tiež použitá.

ZOZNAM POUŽITEJ LITERATÚRY

[1] PRINCE, Simon J. D. *Computer vision: models, learning, and inference*. New York: Cambridge University Press, 2012. ISBN 978-1-107-01179-3.

[2] ANWAR, Aqeel. What are Intrinsic and Extrinsic Camera Parameters in Computer Vision? *Towards Data Science* [online]. 2022. Dostupné z: https://towardsdatascience.com/what-are-intrinsic-and-extrinsic-camera-parameters-in-computer-vision-7071b72fb8ec

[3] SHANKAR, Yalda. Homography Estimation. *Towards Data Science* [online]. 2021. Dostupné z: https://towardsdatascience.com/estimating-a-homography-matrix-522c70ec4b2c

[4] OVCHAR, Illya. Lenses Don't Cause Perspective Distortion and 'Lens Compression'. *PetaPixel* [online]. 2. srpen 2021. Dostupné z: https://petapixel.com/2021/08/02/lenses-dont-cause-perspective-distortion-and-lens-compression/

[5] MANSUROV, Nasim. What is Lens Distortion? *photographylife* [online]. 5. červenec 2020. Dostupné z: https://photographylife.com/what-is-distortion

[6] SZELISKI, Richard. *Computer vision: algorithms and applications*. London ; New York: Springer, 2011. Texts in computer science. ISBN 978-1-84882-934-3.

[7] VALERGA, Ana, Rocio JIMENEZ-RODRIGUEZ a Sergio FERNANDEZ-VIDAL. *Photogrammetry as an Engineering Design Tool*. 2020. ISBN 978-1-83968-212-4.

[8] PAN, Minghua a Guoli ZHU. A Novel Method for the Distortion Modification of Camera Lens. In: *2010 International Conference on Optoelectronics and Image Processing (ICOIP)*: *2010 International Conference on Optoelectronics and Image Processing* [online]. Haiko, Hainan, China: IEEE, 2010, s. 92–95 [vid. 2022-05-13]. ISBN 978-1-4244-8683-0. Dostupné z: doi:10.1109/ICOIP.2010.234

[9] ZHU, Qiuyu. A technique of camera calibration using single planar calibration image. In: *2012 5th International Congress on Image and Signal Processing (CISP)*: *2012 5th International Congress on Image and Signal Processing* [online]. Chongqing, Sichuan, China: IEEE, 2012, s. 824–827 [vid. 2022-05-17]. ISBN 978-1-4673-0964-6. Dostupné z: doi:10.1109/CISP.2012.6469808

[10] ZHANG, Z. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. 2000, **22**(11), 1330–1334. ISSN 01628828. Dostupné z: doi:10.1109/34.888718

[11] KAEHLER, Adrian a Gary R. BRADSKI. *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. First edition, Second release. Sebastopol, CA: O'Reilly Media, 2017. ISBN 978-1-4919-3799-0.

[12] BRAHMBHATT, Samarth. *Practical OpenCV*. New York, NY: Apress, 2013. Technology in action. ISBN 978-1-4302-6079-0.

[13] TAREEN, Shaharyar Ahmed Khan a Zahra SALEEM. A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK. In: *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*: *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)* [online]. Sukkur: IEEE, 2018, s. 1–10 [vid. 2022-05-14]. ISBN 978-1-5386-1370-2. Dostupné z: doi:10.1109/ICOMET.2018.8346440

[14] ZUJUN YU, HONGTAO ZHANG, BAOQING GUO, a LIQIANG ZHU. A mosaic method for large perspective distortion image. In: *2012 International Conference on Measurement, Information and Control (MIC)*: *Proceedings of 2012 International Conference on Measurement, Information and Control* [online]. Harbin, China: IEEE, 2012, s. 506–510 [vid. 2022-05-18]. ISBN 978-1-4577-1604-1. Dostupné z: doi:10.1109/MIC.2012.6273352

[15] SPHINX. *OpenCV 3.0.0-dev documentation* [online]. 10. listopad 2014. Dostupné z: https://docs.opencv.org/3.0-beta/index.html#

[16] OPENCV TEAM. *About Open CV* [online]. Dostupné z: https://opencv.org/about/

[17] ČUKIĆ, Ivan. *Functional programming in C++*. Shelter Island, New York: Manning Publications Company, 2019. ISBN 978-1-61729-381-8.

[18] RICHARD, Paul. What is C++ programming language? *tutorialspoint* [online]. 14. únor 2018. Dostupné z: https://www.tutorialspoint.com/What-is-Cplusplus-programming-language

[19] ALBATROSS. *An Overview of Programs and Programming Languages* [online]. Dostupné z: https://www.cplusplus.com/info/description/

[20] REINA, Andreagiovanni, Alex J. COPE, Eleftherios NIKOLAIDIS, James A. R. MARSHALL a Chelsea SABO. ARK: Augmented Reality for Kilobots. *IEEE Robotics and Automation Letters* [online]. 2017, **2**(3), 1755–1761. ISSN 2377-3766, 2377-3774. Dostupné z: doi:10.1109/LRA.2017.2700059

[21] FISCHLER, Martin A. a Robert C. BOLLES. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* [online]. 1981, **24**(6), 381–395. ISSN 0001-0782, 1557-7317. Dostupné z: doi:10.1145/358669.358692

[22] ENG, Lee Zhi. *Qt5 C++ GUI programming cookbook: use Qt5 to design and build a graphical user interface that is functional, appealing, and user-friendly for your software application*. Birmingham Mumbai: Packt Publishing, 2016. Quick answers to common problems. ISBN 978-1-78328-027-8.

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK

CV	Computer vision.
STL	Standard Template Library.
ISO	International Organization for Standardization.
ARK	Augmented Reality for Kilobots.
BCS	Base Control Software Structure
OHC	Over-head Controller
CUDA	Compute Unified Device Architecture
ORB	Oriented FAST and rotated BRIEF.
BRIEF	Binary Robust Independent Elementary Features.
BRISK	Binary Robust Invariant Scalable Keypoints
SIFT	Scale-invariant feature transform.
SURF	Speeded up robust features.
CV_8UC1	špecifikácia typu matice v Open CV, označuje 8-bitové jednokanálové pole.
RANSAC	Random sample consensus.
GUI	Graphical user interface.
XML	eXtensible Markup Language.
FAST	Features from Accelerated Segment Test.

ZOZNAM OBRÁZKOV

ZOZNAM PRÍLOH

PRÍLOHA P I. - CD

PRÍLOHA P I. - CD

- fulltext.pdf
- LG_project_BP.zip