**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE**
**FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-5382-97817

# SOFTVÉROVÉ RIEŠENIE KAMEROVÉHO SYSTÉMU PRE 3D VIDEO KONFERENČNÝ PRENOS

**BAKALÁRSKA PRÁCA**

**2021**

**Marek Kačmár**

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE**
**FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-5382-97817

# SOFTVÉROVÉ RIEŠENIE KAMEROVÉHO SYSTÉMU PRE 3D VIDEO KONFERENČNÝ PRENOS

**BAKALÁRSKA PRÁCA**

| | |
|---|---|
| Študijný program : | Aplikovaná informatika |
| Názov študijného odboru: | Informatika |
| Školiace pracovisko: | Ústav multimediálnych informačných a komunikačných technológií |
| Vedúci záverečnej práce: | prof. Ing. Gregor Rozinaj, PhD. |

**2021** **Marek Kačmár**

**Slovenská technická univerzita v Bratislave**
**Ústav multimediálnych informačných a komunikačných technológií**

**Fakulta elektrotechniky a informatiky**
**2020/2021**

# ZADANIE BAKALÁRSKEJ PRÁCE

| | |
|---|---|
| Autor práce: | Marek Kačmár |
| Študijný program: | aplikovaná informatika |
| Študijný odbor: | informatika |
| Evidenčné číslo: | FEI-5382-97817 |
| ID študenta: | 97817 |
| Vedúci práce: | prof. Ing. Gregor Rozinaj, PhD. |
| Miesto vypracovania: | Ústav multimediálnych informačných a komunikačných technológií |
| Názov práce: | **Softvérové riešenie kamerového systému pre 3D video konferenčný prenos** |
| Jazyk, v ktorom sa práca vypracuje: | slovenský jazyk |
| Špecifikácia zadania: | Súčasné systémy pre audiovizuálny prenos poskytujú iba obmedzené možnosti voľby smeru pohľadu na mieste prenosu a väčšinou neumožňujú človeku vnímať okolité 3D prostredie stereoskopicky. Súčasné technologické možnosti by pri vhodnom usporiadaní kamier mohli poskytnúť sledovanie vzdialeného priestoru na diaľku s multismerovým 3D pohľadom, zobrazenie 3D objektu na vzdialenom mieste alebo možnosť sledovania vzdialeného priestoru z viacerých miest.<br>Úlohy:<br>1. Naštudujte možnosti rozšírenej funkcionality 3D viacsmerového video prenosu.<br>2. Navrhnite a podľa možností aj zrealizujte systém na viacsmerový 3D video prenos.<br>3. Vyhodnoťte navrhnuté riešenie. |
| Rozsah práce: | min 1AH, max 2AH |

Literatúra:

- KAĽAVSKÝ, E. -- ROZINAJ, G. Stitching images for 3D view. In MINÁRIK, I. -- ROZINAJ, G. *Redžúr 2019*. Bratislava: Vydavateľstvo Spektrum STU, 2019, s. 35--38. ISBN 978-80-227-4931-2.

| | |
|---|---|
| Dátum zadania: | 15. 02. 2021 |
| Dátum odovzdania: | 04. 06. 2021 |

**Marek Kačmár**
študent

**prof. Ing. Gregor Rozinaj, PhD.**
vedúci pracoviska

**Dr. rer. nat. Martin Drozda**
garant študijného programu

# SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

| | |
|---|---|
| Študijný program : | Aplikovaná informatika |
| Bakalárska práca: | Softvérové riešenie kamerového systému pre 3D video konferenčný prenos |
| Autor: | Marek Kačmár |
| Vedúci záverečnej práce: | prof. Ing. Gregor Rozinaj, PhD. |
| Miesto a rok predloženia práce: | Bratislava 2021 |

Počítačové spájanie obrazu je nesmierne výpočtovo náročný proces, ktorý stále aj po mnohých rokoch predstavuje veľkú výzvu v oblasti počítačového videnia. Pri zošívaní obrazu sa vytvára 2D panoráma kombináciou obrazu z viacerých kamier, ktorý sa prekrýva, čím sa vytvorí rozšírené zobrazenie okolitého priestoru. Využíva sa v medicíne, pri vytváraní satelitných snímkov alebo ako aj súčasť rôznych aplikácii v smartfónoch. Cieľom tejto práce je vytvorenie 2 priestorov – panorám, ktoré sa budú zaznamenávať prostredníctvom kamerového systému, spájať a vysielať do ľavého a pravého oka, čím sa vytvorí dojem tzv. "virtuálneho teleportu" – efektu, ktorým sa v mozgu vytvorí ilúzia, že sa osoba, ktorej sa do očí vysiela prenos z kamier, nachádza v miestnosti, kde je umiestnený kamerový systém. V tejto práci si predstavíme proces spájania obrazu, navrhneme možné riešenie systému pre 3D video konferenčný prenos a podľa tohto návrhu sa ho pokúsime zrealizovať, pričom našu implementáciu overíme a zhodnotíme. Výsledkom, ku ktorému sme sa dopracovali v tejto práci, je úspešná realizácia návrhu, kde sme sa pokúsili v zjednodušenej verzii vytvoriť snímanie priestoru stereoskopickou kamerou, ktoré sme simulovali 2 dvojicami stereokamier. Snímali sme dynamickú scénu, a tú sme spájali pre ľavé a pravé oko zvlášť. Následne sme túto scénu zobrazili cez Android smartfón do okuliarov Google Cardboard, v ktorých si ju dokážeme prezerať stereoskopicky.

Kľúčové slová: obraz, 3D, kamery

# ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

| | |
|---|---|
| Study Programme: | Applied Informatics |
| Bachelor Thesis: | Software solution of camera system for 3D spherical video conference transmission |
| Autor: | Marek Kačmár |
| Supervisor: | prof. Ing. Gregor Rozinaj, PhD. |
| Place and year of submission: | Bratislava 2021 |

Computer image stitching is an extremely computationally intensive process that still poses a major challenge in the field of computer vision after many years later. In image stitching, a 2D panorama is created by combining images from multiple cameras and overlaying them to create an augmented view of the surrounding space. It is used in medicine, in the creation of satellite images or as a part of various smartphone applications. The aim of this work is to create 2 spaces - panoramas that will be recorded by the camera system, combined and transmitted to the left and right eye, creating the impression of a "virtual teleport" - an effect that creates the illusion in the brain that the person to whose eyes the transmission from the cameras is transmitted is in the room where the camera system is located. In this paper we will introduce the image fusion process, propose a possible system solution for 3D video conferencing transmission, and attempt to implement it according to this proposal, while verifying and evaluating our implementation. The result we have achieved in this work is a successful implementation of the design, where we have attempted to create a simplified version of the space capture with a stereoscopic camera, which we have simulated with 2 pairs of stereo cameras. We recorded a dynamic scene, and we fused it for the left and right eye separately. Then we displayed this scene through an Android smartphone to Google Cardboard device, in which we can view the scene stereoscopically.

Key words: image, 3D, cameras

# Vyhlásenie autora

Podpísaný Marek Kačmár čestne vyhlasujem, že som bakalársku prácu Softvérové riešenie kamerového systému pre 3D video konferenčný prenos vypracoval na základe poznatkov získaných počas štúdia a informácií z dostupnej literatúry uvedenej v práci.
Uvedenú prácu som vypracoval pod vedením pána profesora Ing. Gregora Rozinaja, PhD.

V Bratislave dňa 11.06.2021

.....
podpis autora

# Poďakovanie

Chcel by som sa poďakovať môjmu vedúcemu bakalárskej práce prof. Ing. Gregorovi Rozinajovi, PhD, za jeho odborné rady, pripomienky, usmernenie, pomoc a hlavne ochotu konzultovať počas tvorby tejto bakalárskej práce.

# Obsah


**1 Úvod ..... 1**

**2 Analýza problému ..... 2**

2.1 Spájanie obrazu ..... 2

2.1.1 Druhy spájania obrazu ..... 2

2.2 Model spájania obrazu ..... 3

2.3 Získanie snímok ..... 4

2.4 Detekcia a spájanie kľúčových bodov ..... 4

2.4.1 FAST(Features from accelerated segment test) ..... 4

2.4.2 SIFT(Scale-invariant feature transform) ..... 6

2.4.3 SURF(Speeded up robust features) ..... 8

2.4.4 ORB(Oriented FAST and rotated BRIEF) ..... 9

2.5 Odhad RANSAC ..... 10

2.6 Zakrivenie a zarovnanie obrazu ..... 12

2.6.1 Afínna transformácia ..... 12

2.6.2 Perspektívna transformácia ..... 12

2.6.3 Prokroustesova transformácia ..... 13

2.7 Spojenie obrazu ..... 14

2.7.1 Alfa zahladenie ..... 14

2.7.2 Feathering ..... 15

2.7.3 Pyramída ..... 15

2.8 Stereoskopia v počítačovom videní ..... 15

**3 Ciele, technológie, návrh a opis riešenia ..... 17**

3.1 Ciele práce ..... 17

3.2 Použité technológie ..... 17

3.3 Návrh a opis vytvoreného systému ..... 18

3.3.1 Zachytenie obrazu z kamier ..... 18

3.3.2 Spojenie obrazu ..... 19

3.3.3 Zobrazenie spojeného obrazu ..... 20

**4 Implementácia ..... 21**

4.4 Konzolová aplikácia zošívajúca obraz ... 21
4.4.1 Načítavanie obrázkov ... 21
4.4.2 Hľadanie, deskripcia a spojenie kľúčových bodov ... 22
4.4.3 Získanie rotačnej matice a zakrivenie obrazu ... 23
4.4.4 Zarovnanie a exportovanie obrázkov ... 23
4.5 Unity projekt ... 24
4.5.1 SceneVR ... 24
4.5.2 VR ... 25

**5 Testovanie a výsledky ... 27**
5.6 Test rýchlosti hľadania kľúčových bodov ... 27
5.7 Test spájania obrazu z odlišných vzdialeností ... 28

**6 Záver ... 30**

**Príloha A: Štruktúra elektronického nosiča ... A1**

**Príloha B: Návod na inštaláciu programov ... B1**

**Príloha C: Používateľská príručka ... C1**

# Zoznam obrázkov a tabuliek

Obrázok 1 Obmedzené pokrytie jednotlivých druhov snímania ..... 2
Obrázok 2 Príklad sférickej panorámy vytvorenej z 54 fotografii ..... 3
Obrázok 3 Vizualizácia postupu spájania obrazu ..... 3
Obrázok 4 Detekcia rohu pri použití FAST algoritmu ..... 5
Obrázok 5 Gausova pyramída a Gaussov rozdiel ..... 6
Obrázok 6 Transformácia gradientov obrázku na deskriptory kľúčových bodov ..... 8
Obrázok 7 Porovnanie nájdených kľúčových bodov algoritmami SIFT a SURF ..... 9
Obrázok 8 Spojenie bodov typu inlier medzi 2 obrázkami ..... 11
Obrázok 9 Porovnanie panorámy so zjavnými trhlinami s panorámou po aplikácii zahladenia obrazu ..... 14
Obrázok 10 Scéna sceneVR vytvorená v Unity ..... 25
Obrázok 11 Scéna VR zobrazujúca na rovinu obraz pre pravé a ľavé oko ..... 26
Obrázok 12 Porovnanie spojených obrázkov z malej a väčšej vzdialenosti ..... 28

Tabuľka 1 Porovnanie algoritmov ..... 27

# Zoznam skratiek a značiek

nm – nanometer

ms – milisekunda

GB – gigabajt

GHz – gigahertz

PNG – Portable Network Graphics

PDF – Portable Document Format

CUDA – Compute Unified Device Architecture

GPU – Graphics Processing Unit

CPU – Central Processing Unit

RANSAC – Random Sample Consensus

SURF – Speeded Up Robust Features

SIFT – Scale-Invariant Feature Transform

FAST – Features from Accelerated Segment Test

ORB – Oriented FAST and Rotated BRIEF

VR – Virtual Reality

SDK – Software Development Kit

RAM – Random Access Memory

API – Application Programming Interface

# 1 Úvod

Človek dokáže vnímať okolité prostredie 2 spôsobmi: panoramaticky(monokulárne) a stereoskopicky(binokulárne). Pri monokulárnom videní sú obrazové signály vysielané do mozgu len z 1 oka, pri stereoskopickom sú vysielané do mozgu signály z oboch očí. Stereoskopickým videním mozog tieto signály spracuje a spojí, čím umožní človeku vnímať 3D priestor – približnú vzdialenosť objektu, jeho tvar, či veľkosť.

Oblasť počítačového videnia sa v poslednej dobe tento vnem pokúša replikovať aj do softvérovej podoby s použitím kamerových systémov, ktoré do určitej miery dokážu napodobniť ľudský orgán – oko. Technikou spájania obrazu sa kombinuje obraz viacerých kamier do 2D panorámy, čím možno vytvoriť 1 celistvé zobrazenie priestoru. Stereo vnem je možné tak u človeka dosiahnuť vysielaním osobitných 2D priestorov spojených z kamerových systémov do pravého a ľavého oka, čím sa vytvorí dojem vnímania 3D priestoru.

V súčasnosti softvérové spájanie obrazu nadobúda nesmierny potenciál, keďže aplikáciou 5G mobilných sietí sa zvyšuje rýchlosť prenosu dát a znižuje celková latencia. Spájanie obrazu z kamier tak možno použiť nielen vo firemnom prostredí vo forme 3D videokonferencii, no aj napríklad v medicíne, či vo vesmíre pri zachytávaní satelitných snímok.

Cieľom tejto práce je vytvorenie 2 priestorov – panorám, ktoré sa budú zaznamenávať prostredníctvom kamerového systému, spájať do ľavého a pravého oka a tak vytvárať dojem tzv. “virtuálneho teleportu“ – efektu, ktorým sa vytvorí v mozgu ilúzia, že sa osoba, ktorej sa do očí vysiela prenos z kamier, nachádza v miestnosti, kde je umiestnený kamerový systém.

# 2 Analýza problému

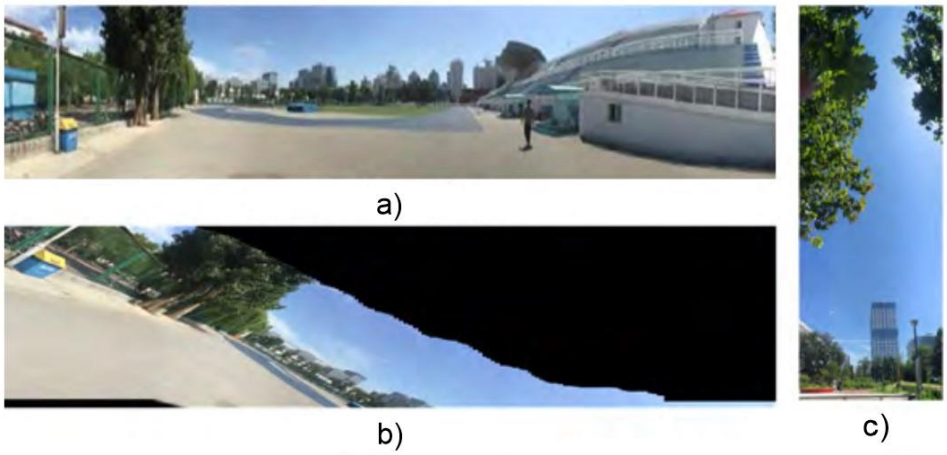
## 2.1 Spájanie obrazu

Spájanie obrazu je proces, pri ktorom sa spájajú dva alebo viac vzájomne sa prekrývajúce obrázky za cieľom vytvorenia jednotného obrazu – panorámy. Na dosiahnutie čo najlepšieho výsledku pri softvérovom spájaní obrazu je potrebné, aby sa jednotlivé obrázky prekrývali v čo najväčšej miere[4].

### 2.1.1 Druhy spájania obrazu

Spájanie obrazu sa rozdeľuje nasledovne:

- Mozaika – spojenie viacerých radov obrázkov do spojitej mozaiky za podmienky, že fotoaparát sa nerotuje okolo jedného špecifického bodu, ale bude otočená vždy kolmo k subjektu, ktorý zachytáva
- Jednoriadková panoráma – spojenie jediného radu obrázkov za podmienky, že fotoaparát sa otáča smerom vľavo alebo vpravo okolo jedného špecifického bodu v horizontálnej alebo vertikálnej rovine
- Viacriadková panoráma – spojenie viacerých radov obrázkov za podmienky, že fotoaparát sa neotáča len smerom vľavo alebo vpravo po rovine, ale môže sa otáčať aj smerom hore a dole[4]




*Obrázok 1 Obmedzené pokrytie jednotlivých druhov snímania a) horizontálne snímanie, b) šikmé snímanie, c) vertikálne snímanie[6]*

- Panoramatická kamera – spojenie koncov panorám, ktoré sú vytvorené fotoaparátom schopným vytvárať panoramatické snímky
- Sférická panoráma – spojenie ľubovoľného množstva obrázkov do sféry, od viacriadkovej panorámy sa líši tým, že zatiaľ čo pri viacriadkovej panoráme sa pri pohľade nahor a nadol nachádzajú prázdne, nespojené miesta, pri sférickej panoráme sú tieto miesta spojené a tak vytvárajú hladký prechod[4]

*Obrázok 2 Príklad sférickej panorámy vytvorenej z 54 fotografii[1]*

## 2.2 Model spájania obrazu

Model spájania obrazu pozostáva z piatich krokov: získanie snímok, detekcia kľúčových bodov a ich spájanie, odhad RANSAC, zarovnanie obrazu a spojenie obrazu. Úspešným prechodom všetkými fázami, kde sa používa viacero algoritmov, ktoré budú detailnejšie popísané v ďalších častiach, sa vytvorí výsledná panoráma[2].


*Obrázok 3 Vizualizácia postupu spájania obrazu[3]*

## 2.3 Získanie snímok

Počas prvej fázy sa získavajú obrázky z určitého zdroja alebo zdrojov, zvyčajne sa ako zdroj používa fotoaparát. Samotný proces získavania snímok vyžaduje korektné zvolenie pozície na následné zachytenie obrázkov. V závislosti od toho, akým spôsobom sú zachytené obrázky, je možno získať rôzne typy výsledných panorám, ktoré boli spomenuté v predošlej kapitole[2],[4].

## 2.4 Detekcia a spájanie kľúčových bodov

Druhá fáza, označovaná ako detekcia kľúčových bodov, je jednou z najdôležitejších fáz celého procesu spájania obrazu. V tejto fáze prebieha hľadanie kľúčových bodov v dvoch alebo viacerých obrázkoch tak, že sa na jednotlivé obrázky nepozerá, ako na celky, ale v jednotlivých obrázkoch sa vyznačia špecifické body, ktoré sa budú v ďalších fázach analyzovať. Pretože množstvo online algoritmov vyžaduje spracovanie obrázkov v reálnom čase, rýchlosť, akou sa tieto kľúčové body detegujú je kľúčová. Najdôležitejším prvkom, ktorý sa hľadá ako kľúčový bod v obrázku, je roh. Hlavným dôvodom, prečo sa v obrázkoch hľadajú rohy je fakt, že intenzita pixelov v okolí rohu sa prudko zvyšuje. Metódy hľadania podobných vlastností nehľadajú podobnosť len v určitých bodoch, ale aj čiarach a iných geometrických tvaroch. Existuje množstvo algoritmov hľadania kľúčových bodov, ktorými sú napríklad SURF, SIFT, FAST, ORB, Harrisov detektor rohov a i. Tieto algoritmy sú schopné rýchlo detegovať vzájomné vzťahy medzi obrázkami, taktiež sú odolné voči pohybu okolitej scény, čím sú ideálne pre použitie bežnými používateľmi[3].

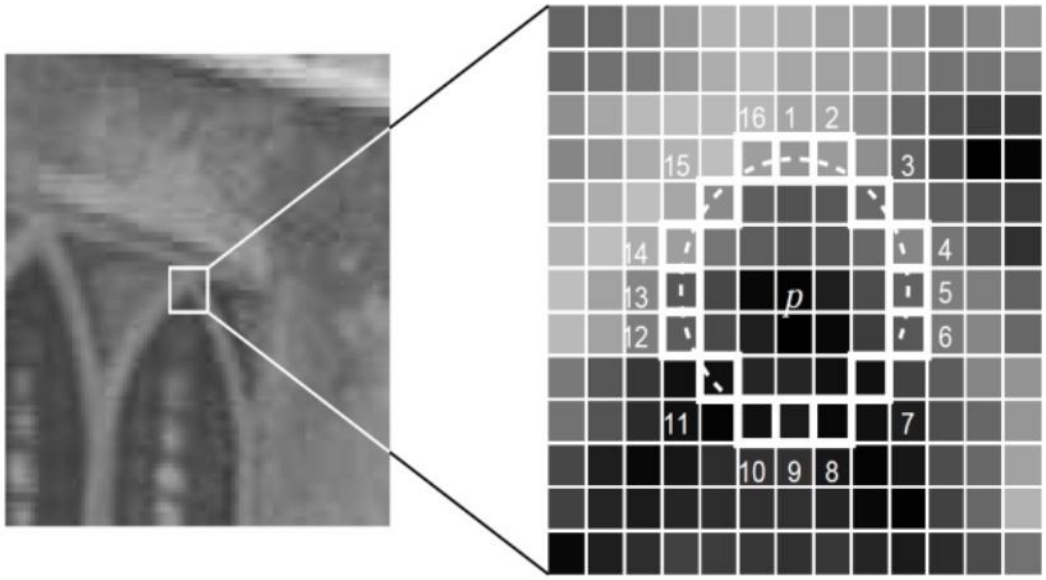
### 2.4.1 FAST(Features from accelerated segment test)

FAST sa považuje za jeden z najrýchlejších algoritmov používaných na získavanie kľúčových bodov. Princíp algoritmu spočíva v porovnávaní intenzity jasu vybraného pixelu s jeho okolím, čím sa vytvára kruhová maska. FAST určí vybraný bod ako kľúčový vtedy, ak aspoň 12 súvislo po sebe idúcich pixelov v maske je tmavších alebo svetlejších než bod samotný. Ak neplatí žiadna z podmienok zobrazených v (1), bod nemôže byť označený ako kľúčový bod predstavujúci roh[5].

$$S_x = \begin{cases} t, I_x \leq I_p - P \\ j, I_x \geq I_p + P \\ r, I_p - P < I_x < I_p + P \end{cases} \quad (1)$$

$S_x$ popisuje stav pixelu x v maske, $I_x$ popisuje hodnotu intenzity jasu pixelu x, $I_p$ predstavuje hodnotu intenzity jasu pixelu p, P je prahový parameter a písmená t(tmavší), j(jasnejší) a r(rovnaký) predstavujú skupiny bodov prislúchajúce jednotlivým podmienkam[5].

Obr. 2 zobrazuje oblasť, ktorá bude analyzovaná. Na začiatku algoritmu sa určí stredový pixel ako kandidát na roh v kruhu o veľkosti 16 pixelov. Ak hodnota aspoň 3 zo 4 hodnôt pixelov $I_1$, $I_5$, $I_9$, $I_{13}$ nie je väčšia alebo menšia ako $I_p$ + T, tak potom bod p nie je kľúčovým bodom. Ak sú aspoň 3 zo 4 hodnôt väčšie alebo menšie ako hodnota $I_p$ + T, algoritmus skontroluje či 12 zo 16 po sebe idúcich pixelov spĺňa podmienku. Táto procedúra sa zopakuje pre každý pixel nachádzajúci sa v obrázku[3].




*Obrázok 4 Detekcia rohu pri použití FAST algoritmu[3]*

### 2.4.2 SIFT(Scale-invariant feature transform)

Algoritmus detegujúci a popisujúci vektory lokálnych čŕt nachádzajúce sa v obrázkoch. Tieto črty sú nemenné voči zmenám mierky, rotácie, osvetlenia, či malým zmenám uhlu pohľadu jednotlivých obrázkov. Okrem iného sú tieto body veľmi jednoducho extrahovateľné s relatívne nízkou chybovosťou[4],[7].

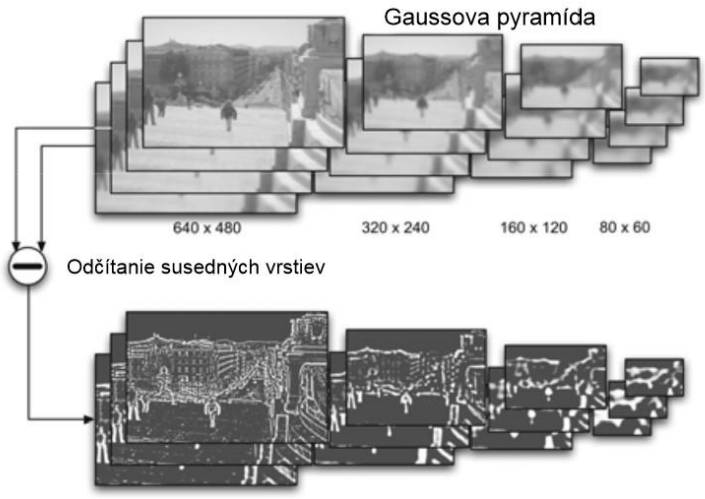
Algoritmus sa delí na 4 časti:

- Detekcia lokálnych extrémov

Algoritmus sa snaží na začiatku lokalizovať miesta, ktoré sú nemenné voči rotáciám, zväčšeniam, či deformáciám obrazu. To sa dosiahne aplikáciou Gaussovej krivky(2) na priestor zo snímky a následným získaním maxima a minima z jej rozdielu. Efektívnym výpočtom je napríklad vytvorenie obrázkovej pyramídy. Maximum a minimum sa z tejto funkcie zistí porovnávaním každého pixelu pyramídy s jeho susedom. Ak je hodnota pixela, ktorý je porovnaný s jeho ďalšími 8 susedmi maximom alebo minimom, určí sa tento pixel ako kľúčový bod[7].

$$g(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-x^2/2\sigma^2} (2)$$

Pri lokalizácii kľúčových bodov sa využíva $\sigma = \sqrt{2}$ [7].




*Obrázok 5 Gausova pyramída(vrchná časť) a Gaussov rozdiel(spodná časť)[8]*

- Odstránenie nežiadúcich čŕt

Za účelom zvýšenia spoľahlivosti algoritmu je potrebné, aby sa odstránili črty, ktoré sú určené nesprávne. Najčastejšie sú odstraňované črty, ktorých body boli chybne nájdené v dôsledku šumu na vstupnom obrázku a body ležiace na okrajoch. Nežiadúce črty sú nájdené vypočítaním Hessovej matice (3)[8].

$$H(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix} \quad (3)$$

Kde $L_{xx}(x,\sigma)$, $L_{xy}(x,\sigma)$ a $L_{yy}(x,\sigma)$ predstavujú konvolúciu parciálnej derivácie druhého rádu $\frac{\sigma^2}{\sigma x^2} g(\sigma)$ bodu $x$ v obrázku[9].

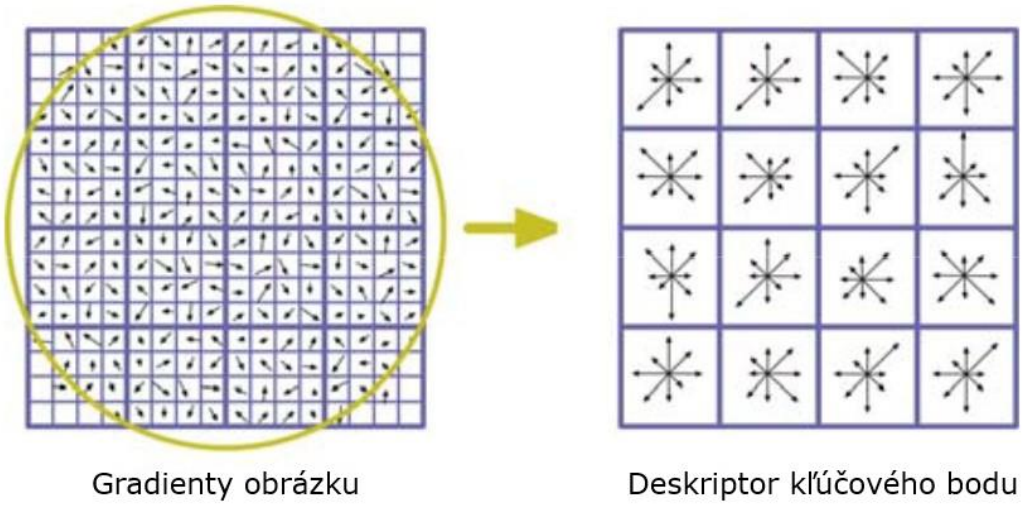
- Pridelenie orientácie

Na dosiahnutie imúnnosti voči rotáciám je potrebné kľúčovým bodom priradiť orientáciu. To sa docieli transformovaním gradientov v určitej vzdialenosti okolo každého kľúčového bodu do súradnicového systému[8].

$$m(x,y) = \sqrt{\left(\frac{\partial \mathrm{D}(\mathrm{x},\mathrm{y})}{\partial \mathrm{x}}\right)^2 + \left(\frac{\partial \mathrm{D}(\mathrm{x},\mathrm{y})}{\partial \mathrm{y}}\right)^2} \quad (4)$$

$$\theta(x,y) = tan^{-1}\left(\frac{\frac{\partial \mathrm{D}(\mathrm{x},\mathrm{y})}{\partial \mathrm{y}}}{\frac{\partial \mathrm{D}(\mathrm{x},\mathrm{y})}{\partial \mathrm{x}}}\right)$$

Následne je z výsledných hodnôt θ(x,y) vytvorený histogram, ktorého váha je určená veľkosťami hodnôt m(x,y), čím sa získa primárny smer gradientov okolo kľúčových bodov[8].



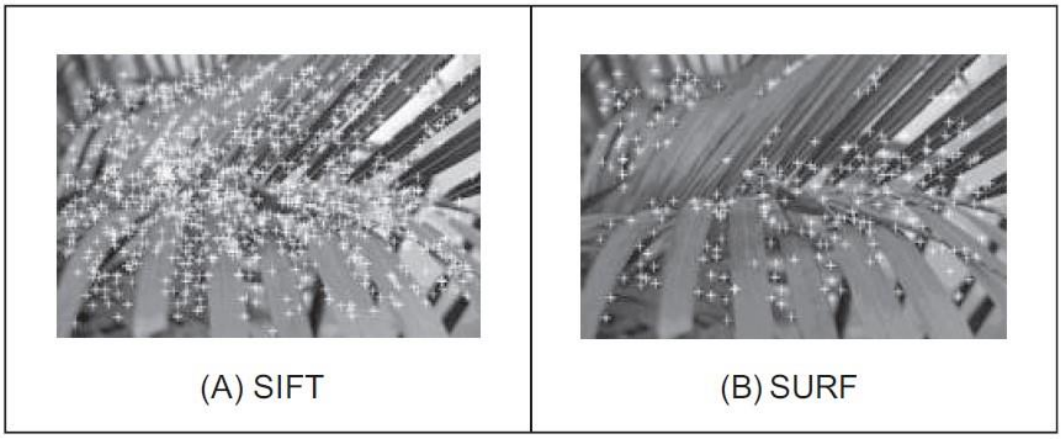

*Obrázok 6 Transformácia gradientov obrázku na deskriptory kľúčových bodov[8]*

- Deskripcia kľúčových bodov

Podľa obr. 6, každý región vytvorený zo 4x4 histogramov okolo každého kľúčového bodu má priradených 8 smerov, ktorých sčítaním je vytvorený 128-prvkový vektor výstupu. Každá takáto operácia musí byť vykonávaná osobitne[8].

### 2.4.3 SURF(Speeded up robust features)

SURF je algoritmus, ktorý je v porovnaní s algoritmom SIFT, od ktorého je algoritmus SURF odvodený, rýchlejší aj napriek zachovaniu určitej robustnosti voči zmenám uhlu pohľadu, či rotáciám obrázku. Rýchlejší výpočet algoritmu sa dosahuje budovaním na už existujúcich detektoroch a deskriptoroch a ich zjednodušovaním, čím sa vytvárajú nové spôsoby detekcie a deskripcie kľúčových bodov. To znamená, že SURF vie poskytnúť menej čŕt, ktoré pretrvávajú, čím je rýchlejší a SIFT viac, no niektoré môžu byť v budúcnosti odstránené, čím sa stáva pomalším. Použitie jednotlivých algoritmov záleží na ich aplikácii[9],[10].




*Obrázok 7 Porovnanie nájdených kľúčových bodov algoritmami SIFT a SURF[10]*

SURF využíva na detekciu kľúčových bodov Hesseho maticu(3), ktorá zaručuje vysokú presnosť a efektivitu. Hľadané kľúčové body sú určené na tých miestach, kde je výsledok determinantu maximum. SURF tak následne rozdelí obrázok na vrstvy, na ktoré sa použije Gaussov filter – rámec o veľkosti 9x9, čo reprezentuje najmenšiu mierku na výpočet máp. Jednotlivé vrstvy obrázku sú aplikované do Gaussovej krivky(2), ktorých výstup je vložený do Hessovej matice(3), z ktorej sa potom vypočíta determinant (5)[9].

$$\det(H_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2 \quad (5)$$

### 2.4.4 ORB(Oriented FAST and rotated BRIEF)

ORB je rýchly binárny deskriptor, ktorý využíva FAST detektor kľúčových bodov a BRIEF(binary robust independent elementary features) deskriptor. Kombináciou týchto dvoch techník sa tak dosiahne vysoký výkon a nízke náklady[11],[12].

Algoritmus na svojom začiatku použije detektor kľúčových bodov FAST-9(kruhový polomer 9 pixelov), ktorý ma celkovo výbornú efektivitu detekcie. FAST nedostatočne deteguje rohy v obraze, čo je následne vyriešené aplikáciou Harrisovho dektora rohov[12].

BRIEF deskriptor následne zabezpečí invariantnosť voči rotácii a odolnosť voči šumu výberom a porovnávaním intenzity párov náhodne vybraných bodov nachádzajúcich sa

v blízkom okolí vybraného kľúčového bodu. BRIEF potom priradí binárny opis danému kľúčovému bodu dosadením vybraných bodov do nasledovne:

$$\tau(p; x, y) := \begin{cases} 1 & : p(x) < p(y) \\ 0 & : p(x) \geq p(y) \end{cases}, \quad (6)$$

kde τ predstavuje kľúčový bod, p(x) a p(y) intenzitu vybraných bodov.

BRIEF deskriptor je tak charakterizovaný ako $n_d$-dimenzionálny bitový reťazec vytvorený zo vzťahu (6):

$$f_{n_d}(p) := \sum_{1 \leq i \leq n_d} 2^{i-1} \tau(p; x_i; y_i), \quad (7)$$

pričom $n_d$ sa môže rovnať hodnotám 128, 256 a 512. Následne sa tieto deskriptory porovnávajú výpočtom Hammingovej vzdialenosti, teda meraním minimálneho počtu zmien, ktoré sa vyžadujú na zmenu prvého deskriptoru na druhý. Na základe tejto vzdialenosti sa tak vyznačia najlepšie páry vytvorené z kľúčových bodov[12],[13].

## 2.5 Odhad RANSAC

RANSAC je algoritmus vyvinutý v rámci komunity počítačového videnia M.A. Fischlerom a R.C. Bollesom. Jeho hlavným princípom je vytváranie odhadu parametrov matematického modelu z poskytnutých dát, ktoré môžu obsahovať veľké množstvo nežiadúcich bodov. Algoritmus je široko používaný vzhľadom na jeho jednoduchosť a robustnosť, keďže dokáže tolerovať množstvo chybne zvolených kľúčových bodov (v niektorých prípadoch aj viac než 50% z celkového počtu zvolených kľúčových bodov)[14],[15].

Na začiatku sa vyberie čo najmenší počet náhodných bodov zo vstupnej množiny bodov, z ktorých sa určia parametre modelu. Následne sa určí, koľko bodov z množiny všetkých bodov spĺňa vopred určenú toleranciu ϵ (zvyčajne sa hodnota ϵ pohybuje v rozmedzí 1-3

pixely). Proces náhodného výberu sa následne opakuje N-krát, pričom množina s najväčším počtom „inliers" (bodmi spĺňajúcimi parametre modelu) je zvolená ako výsledná.

Počet iterácií N musí byť dostatočne veľký na to, aby bolo zaručené, že aspoň v jednej z iterácií sa bude nachádzať množina bez chybného bodu – „outlier". Tento vzťah môžeme zapísať do nasledujúcej rovnice (8):

$$1 - p = (1 - u^m)^N, \qquad (8)$$

pričom $p$ predstavuje pravdepodobnosť (väčšinou nastavená na 0.99), $u^m$ predstavuje pravdepodobnosť, že všetky náhodne zvolené body sú typu „inlier", (1 – $u$) udáva pravdepodobnosť detekcie bodu typu „outlier" a N minimálny počet iterácii.

Rovnicu tak možno upraviť nasledovne:

$$N = \frac{\log(1 - p)}{\log[1 - (1 - v)^m]}, \qquad (9)$$

kde $v$ = (1 - u) v rovnici (8)[1], [15].

*Obrázok 8 Spojenie bodov typu inlier (biela čiara) medzi 2 obrázkami[19]*

## 2.6 Zakrivenie a zarovnanie obrazu

Technika zakrivenia a zarovnania obrazu je dôležitým štádiom v modeli spájania obrazu. Používa sa prevažne na odstránenie optického skreslenia, ktoré sa vytvára odlišnou perspektívou pohľadu vytvorenou posunom fotoaparátu či kamery zachytávajúcej obraz.

Hlavným princípom pri transformácii obrazu zakrivením je, že všetky body (pixely) z prvého obrázku sa mapujú do druhého obrázku. Existuje množstvo rôznych spôsobov, ako dosiahnuť správne namapovanie bodov z prvého obrázku na druhý, no všeobecným prístupom je nájdenie kompromisu medzi zahladením optického skreslenia a nájdením správnych pozícii (x, y), ktoré sa použijú na namapovanie bodov.

Hladkosť optického skreslenia a správna lokalizácia pozícii na zarovnanie sa tak dosahuje rôznymi transformáciami[16]:

### 2.6.1 Afínna transformácia

Afínnou transformáciou sa zachovávajú tzv. „kolineárnosť“ (dodžanie vlastnosti, že všetky pôvodné body ležiace na priamke sa po transformácii budú nachádzať na rovnakej priamke) a pomery vzdialeností bodov (stred úsečky v rámci priamky ostane stredom aj po transformácii).

Všeobecne možno afínnu transformáciu charakterizovať ako kompozíciu translácii, rotácii, dilatácii, či skosení. Vlastnosť zachovávania vzájomných pomerov vzdialeností na priamke však nezaručuje zachovanie celkovej dĺžky alebo uhlov[17].

### 2.6.2 Perspektívna transformácia

Perspektívna transformácia alebo skrátene perspektíva je transformácia zachyteného 3D priestoru na 2D rovinu obrazu, čoho následkom je, že vzdialené objekty sa javia ako menšie než objekty, ktoré sú v zachytenom 3D priestore bližšie. Takúto vlastnosť je možné pozorovať aj u ľudského oka, či kamier.

Pravidlá perspektívneho zobrazenia môžeme rozčleniť nasledovne:

1. Horizont bude vyzerať ako priamka.

2. Priamky v priestore sa javia ak priamky v obrázku.

3. Množina všetkých paralelných priamok konverguje k tzv. „miznúcemu bodu“.

4. Množina paralelných priamok, ktoré sú rovnobežné s 2D rovinou obrazu, nemá „miznúci bod“[18].

### 2.6.3 Prokroustesova transformácia

Ak medzi porovnávanými obrázkami sa nachádza rozdiel spočívajúci v zväčšení alebo rotácii obrazu o $\theta$ stupňov, používa sa Prokroustesova transformácia(10) založená na nasledujúcom určení parametrov $u$ a $v$:

$$u = cx\cos\theta + cy\sin\theta + a_{00}$$
$$v = -cx\sin\theta + cy\cos\theta + b_{00} \quad (10)$$

kde konštanta $c$ upravuje zmenu veľkosti obrazu. Hodnota $c = 1$ nemení veľkosť, $c > 1$ vedie k zväčšeniu a $c < 1$ zmenšeniu obrazu. Prokroustesovú transformáciu tak môžeme použiť na akékoľvek 2 unikátne body z 2 obrázkov.

Všeobecne sa táto transformácia používa takým spôsobom, že všetky obrázky sa zarovnávajú na prvý obrázok. Touto transformáciou sa tak môže zmeniť veľkosť, či pozícia objektu, no tvar zostane nezmenený[16].

## 2.7 Spojenie obrazu

Finálnym dôležitým procesom, ktorým sa vytvára výsledný spojený obrázok, je spojenie alebo taktiež nazývané aj ako zahladzovanie obrazu. Počas procesu spájania obrazu sa môžu vytvárať viditeľné trhliny alebo šmuhy spôsobené rozdielmi v osvetlení obrázkov, rozdielnymi reakciami fotoaparátu, či kamery na okolité prostredie, odlišnom zarovnaní jednotlivých obrázkov, alebo zachytávaním pohyblivého objektu. Na minimalizovanie alebo úplne odstránenie týchto nežiadúcich efektov je potrebné zvoliť správne pixely, ktoré budú patriť do výsledného obrázku a správnu metódu zahladenia obrázku. Zahladením obrázku sa tak zakryjú nežiadúce trhliny a znížia sa farebné rozdiely medzi jednotlivými obrázkami.

*Obrázok 9 Porovnanie panorámy so zjavnými trhlinami (vľavo) s panorámou po aplikácii zahladenia obrazu (vpravo)[20]*

Spôsobov zahladenia obrazu existuje viacero[4],[19]:

### 2.7.1 Alfa zahladenie

Kanál alfa sa väčšinou využíva na určenie nepriehľadnosti pixelov a nadobúda hodnoty od 0% do 100%, pričom ak má pixel 0%, je neviditeľný a ak má 100%, je kompletne nepriehľadný. Alfa kanál môže nadobúdať aj hodnoty reálnych čísel medzi 0 a 1. Všeobecne sa kanál alfa môže chápať ako maska určujúca mieru, do akej by sa mali farby pixelov, ktoré sú na sebe, spojiť.

### 2.7.2 Feathering

Ďalším spôsobom zahladenia obrazu je metóda „Feathering“, ktorá ohodnocuje pixely v závislosti od ich vzdialenosti od stredu obrázku. Pixelom nachádzajúcim sa v strede alebo v blízkosti stredu je tak priradená vyššia „váha“ ako pixelom nachádzajúcim sa v okrajových oblastiach obrázku. Metóda Feathering dokáže s vysokou úspešnosťou odstrániť rozdiely v expozícii, no nedokáže odstrániť rozmazanie spôsobené pohybom objektov v rámci jednotlivých obrázkov[4].

### 2.7.3 Pyramída

Pyramída je hierarchická reprezentácia obrázku, ktorá vytvára súbor obrázkov s rôznymi rozlíšeniami, pričom najvyššia úroveň predstavuje najnižšie rozlíšenie a najnižšia úroveň najvyššie. Využíva sa nie len pri zahladzovaní obrazu, ale aj pri vylepšení alebo odstránení šumu z obrázku[19].

Algoritmus zahladzovania spočiatku rozloží vstupné obrázky do vrstiev s rozlíšeniami obrázkov od najvačšieho po najmenší, čím sa vytvorí Laplaceova pyramída. Následne sa z masiek vytvorených z každého zdrojového obrázku vytvorí dolnopriepustným filtrom Gaussova pyramída. Po tomto kroku sa vytvorí výsledný zahladený obrázok interpoláciou a vzájomným zlúčením všetkých úrovní Laplaceovej a Gaussovej pyramídy[4].

## 2.8 Stereoskopia v počítačovom videní

Prvý geometrický koncept stereo panorám bol vymyslený Ishigurom v roku 1992, no samotné vytvorenie a používanie stereo panorámy vo VR aplikáciách založených na obrázkoch sa začalo v roku 1998. V súčasnosti sa stereo panoráma sa používa pri navigácii robotov, dokumentácii okolitého prostredia, rekonštrukcii 3D scény, či pri vizualizácii stereo obrazu alebo rôznych typov systémov virtuálnej reality.

Princípom, na ktorom spočíva zobrazenie stereo obrazu, je zobraziť ľavú časť obrázku ľavému oku a právu časť pravému. Ľudský mozog tak z páru stereo obrázkov vyvodí informáciu o hĺbke prostredníctvom horizontálnej paralaxy, ktorá je definovaná vodorovným posunom sebe prislúchajúcich bodov v ľavom a pravom obrázku.

Hlavnou výzvou, ktorou je potrebné sa zaoberať počas tvorby stereo panorámy je zaručenie, aby osoba mohla vnímať hĺbku obrazu presne aj počas priblíženia v rôznych častiach obrázku.

Precíznosť vnímania hĺbky je limitovaná rozlíšením obrázku a médiom, ktoré obrázok zobrazuje, a teda je potrebné zvýšiť celkovú disparitu. Pri stereo pohľade je zároveň potrebné, aby hodnota disparity bola pod limitom maximálnej disparity u ľudí. Maximálna disparita je priamo úmerná pozorovacej vzdialenosti. Prekročením hornej hranice maximálnej disparity sa vytvára rozdvojený obraz(dipódia), čo spôsobuje u sledovateľa stereo obrazu nepohodlie a zvýšenú námahu očí, čo môže viesť aj k syndrómu počítačového videnia(CVS)[21].

# 3 Ciele, technológie, návrh a opis riešenia

V tejto kapitole si popíšeme ciele tejto práce, technológie, ktoré boli pri jej návrhu a tvorbe použité a celkový postup vytvorenia systému na základe nadobudnutých poznatkov.

## 3.1 Ciele práce

V súvislosti s touto prácou sme si vyčlenili 3 hlavné ciele:

Prvým a najdôležitejším cieľom bolo vytvoriť panoramatický obraz z dvoch alebo viacerých obrázkov, ktorý dokážeme vnímať stereoskopicky. Súčasťou tohto cieľa bolo dostatočne si naštudovať problematiku spájania obrazu, jednotlivé kroky spájania obrazu, využívané algoritmy a všetky možné spôsoby implementácie. Taktiež bolo potrebné zistiť, ako dosiahnuť 3D vnem, prípadne aké zariadenia použiť, aby sme ho nadobudli.

Ďalším nadväzujúcim cieľom bolo snaženie sa o optimalizáciu nami nájdeného spôsobu spájania obrazu. V rámci tohto cieľa bolo potrebné urýchliť celkový proces spájania obrazu, minimalizovať chybovosť, ktorá pri tomto procese môže v množstve prípadov nastať, zvoliť také algoritmy, ktoré najviac vyhovujú našim požiadavkám, ale taktiež aj dostatočne upraviť parametre, s ktorými pracovali algoritmy podieľajúce sa na procese spájania obrazu.

Posledným hlavným cieľom, ktorý sme si stanovili, bolo vytvorenie takého viacsmerového systému, ktorý dokáže prepojiť nami navrhnuté riešenie pre spájanie obrazu so zariadením, ktoré by nám 3D vnem z poskytnutého vstupu dokázalo vytvoriť. Taktiež by tento systém mal byť schopný prepínania zobrazovaného 3D obrazu na základe pohybov hlavy pri zobrazovaní daného 3D predmetu zaznamenávaného kamerovým systémom.

## 3.2 Použité technológie

Počas celej doby sme pracovali na operačnom systéme Windows 10 Home z dôvodu dlhoročnej skúsenosti, jednoduchosti používateľského rozhrania a použitia vývojového prostredia Visual Studio Community 2019, ktorých spoločným vývojárom je spoločnosť Microsoft, čím sme zaručili maximálnu kompatibilitu. Na implementáciu programu na spájanie obrazu bol použitý programovací jazyk C++, ktorý je v porovnaní s jazykom Python

rýchlejší a multiplatformová open-source knižnica OpenCV 4.1.1, ktorá je taktiež napísaná v jazyku C++. Knižnicu OpenCV sme si vybrali z dôvodu kompatibility, jednoduchej dostupnosti potrebných algoritmov na spracovanie obrazu a možnosti ich prípadnej úpravy.

Na vytvorenie virtuálnej scény na zachytávanie a vytváranie obrazu pre VR cardboard bol použitý program Unity 2019.4.20 s rozšírením o Google VR SDK. Zachytávané 3D modely boli spracované v programe Blender 2.92.

Pri testovaní navrhnutého riešenia bol použitý notebook Lenovo IdeaPad 700-15ISK so 16 GB RAM a procesorom Intel® Core™ i7-6700HQ CPU s frekvenciou 2.60 GHz, 4 jadrami a 8 vláknami.

## 3.3 Návrh a opis vytvoreného systému

V nasledujúcej časti si popíšeme jednotlivé štádia, ktorými nami navrhnutý systém prechádza, a to sú zachytenie obrazu z kamier, spojenie obrazu, načítanie a zobrazenie spojeného obrazu. Systém bude mať vytvorené 2 scény, v 1. scéne sa budú nachádzať kamery a 3D objekt, ktorý budú kamery zachytávať a v 2. scéne sa budú nachádať 2 kamery reprezentujúce ľavé a pravé oko, ktoré budú zobrazovať spojený stereo obraz na výstup.

### 3.3.1 Zachytenie obrazu z kamier

Počiatočným štádiom je proces výberu a zachytenia obrazu z kamier, ktoré sa budú nachádzať v nami vytvorenej scéne s 3D objektom prostredníctvom získaných údajov o polohe a smere pohľadu. V prvom rade je dôležité si správne rozmiestniť kamerový systém zachytávajúci obraz, ktorý je potrebné uložiť v priestore tak, aby paralaxa medzi zachytávaným objektom a kamerami nebola príliš veľká a ani príliš malá. Kamery budú rozmiestnené po dvojiciach okolo nami zachytávaného objektu do kružnice, čím dokážeme pozorovať objekt zo všetkých smerov. Zároveň je potrebné si určiť kamery, z ktorých sa bude spájať obraz pre pravé a ľavé oko. Po zachytení obrazu z vybraných kamier pre ľavé a pravé oko sa pošle obraz na ďalšie spracovanie a teda aj spájanie.

### 3.3.2 Spojenie obrazu

Po prijatí obrázkov na vstupe sa začne proces spájania panoramatického obrazu najprv pre ľavé a tak pre pravé oko. Samotná fáza spájania obrazu je zložená z viacerých krokov, ktoré sú nevyhnutné pre korektné a efektívne vytvorenie zošitej panorámy:

1. Načítanie obrázkov, ktoré sa budú spájať
2. Hľadanie kľúčových bodov medzi obrázkami (algoritmy SURF/SIFT/ORB a i.)
3. Deskripcia kľúčových bodov
4. Spojenie kľúčových bodov
5. Odhad RANSAC (výpočet matice homografie)
6. Zakrivenie a zarovnanie obrazu
7. Spojenie alebo zahladenie obrazu

Celý proces spájania obrazu je dôležité začať načítaním obrazu, pričom je potrebné vedieť presný počet obrázkov, ktoré sa budú spájať. Po overení dostatku obrázkov, ktorého minimom je počet 2 sa pristúpi k ďalšej fáze.

Počas fázy hľadania kľúčových bodov je kľúčové zvoliť algoritmus, ktorý dokáže detegovať kľúčové body v čo najväčšom množstve a čo najkratšom čase v obrázkoch, ktoré boli zachytené pod rôznymi uhlami a s rozdielnymi vzdialenosťami.

Ďalej bude po nájdení kľúčových bodov potrebné opísať a spojiť tieto body z dôvodu budúceho porovnávania týchto bodov.

Odhadom RANSAC sa vyrieši problém priradenia, ktoré časti z prvého obrázku možno priradiť druhému obrázku, a teda vypočítame fundamentálnu maticu s veľkosťou 3x3, ktorá bude udávať informáciu o vzťahu medzi 2 obrázkami v rovnakej scéne, kde sa môžu nami vybrané body zo scény zobraziť v oboch obrázkoch.

Následne sa obrázky zakrivia a zarovnajú, čím sa odstráni určité optické skreslenie vytvorené odlišnými uhlami pohľadu na scénu.

Po finalizácii všetkých predošlých fáz nastane proces splynutia, čím sa vytvorí finálny celistvý panoramatický obraz zošitých obrázkov zo vstupu a pripraví sa na ďalšie spracovanie.

### 3.3.3 Zobrazenie spojeného obrazu

Finálnou fázou tohto systému je zobrazenie spojeného obrazu do ľavého a pravého oka. Obraz sa začne postupne načítavať do predpripravenej scény pre pravé a ľavé oko, a tie sa zobrazia do zariadenia spôsobilého na premietanie 3D obrazu do očí. Týmto zobrazením sa završí celý cyklus navrhnutého systému a môže sa opakovať.

# 4 Implementácia

V tejto kapitole bude predstavená a detailnejšie popísaná praktická implementácia nami navrhnutého systému, ktorého návrh bol popísaný v kapitole 3.3, počínajúc prvotnou inštaláciou OpenCV a končiac finálnym systémom, ktorý vytvára viacsmerový 3D video prenos v Android smartfóne za použitia jednoduchého Google Cardboardu.

## 4.4 Konzolová aplikácia zošívajúca obraz

V tejto podkapitole si popíšeme našu najdôležitejšiu súčasť systému, ktorá je realizovaná ako C++ konzolová aplikácia spájajúca obraz.

Naša aplikácia funguje ako pozmenená forma API rozhrania šitia obrazu na vysokej úrovni, ktorú poskytuje knižnica OpenCV. Zvolili sme tak z dôvodu eliminácie zbytočného opakovania sa implementácie programu na spájanie obrazu, ktorý bol už mnohokrát v minulosti vytváraný a vylepšovaný. Naša aplikácia pracuje s OpenCV verziou 4.1.1 tejto knižnice, ktorá obsahuje veľké množstvo algoritmov (približne 2500) zameraných na počítačové videnie. Knižnica OpenCV však samotná neposkytuje rozšírenie o tzv. nonfree modul (modul pre patentované algoritmy, ktorých použitie je zakázané pre komerčné účely) obsahujúci napríklad algoritmus na detekciu kľúčových bodov ako SURF, ktorý je v našej implementácii použitý, preto bolo potrebné pri inštalácii OpenCV pridať voľne dostupné moduly OpenCV contrib.

### 4.4.1 Načítavanie obrázkov

Prvým krokom po spustení programu je nastavenie si cesty na načítanie vstupných, nespojených obrázkov s rozmerami 640x480 (pomer 4:3) a cesty, kde sa budú ukladať spojené obrázky. Celý program sa opakuje v nekonečnom cykle, pričom vždy sa na začiatku snaží volať funkciu parseFolderImages, ktorej prvý vstupný parameter je offset – číselná hodnota udávajúca index mena png súboru vo vektore zloženého z mien všetkých png súborov nachádzajúcich sa v adresári. Druhý vstupný parameter num_of_pictures udáva počet obrázkov, ktoré sa načítajú do kontajnera img_names, ktorý bude používaný neskôr. Táto hodnota je predvolená ako číslo 2, pretože v našom modeli spájame vždy 2 obrázky pre ľavé oko a 2 obrázky pre pravé oko. V prípade, že sa v adresári nenachádzajú žiadne png

súbory, ktoré by mohol náš program spájať, vráti sa naspäť do nekonečného cyklu a počká 1235 ms a znovu zopakuje volanie funkcie parseFolderImages, kým funkcia nenačíta potrebný počet obrázkov typu png.

```
static int parseFolderImages(int offset, int num_of_pictures)
{

    vector<String> fn;
    glob("C:/Users/kacma/Desktop/bc_panorama/screenshots/*.png", fn, false);
    size_t count = fn.size();
    if (count == 0) {
        return 1;
    }
    if (num_of_pictures + offset > fn.size()) {
        exit;
    }
    for (size_t i = offset; i < (num_of_pictures + offset); i++) {
        img_names.push_back(fn[i]);
        cout << "Proceeding to stitch: ";
        cout << fn[i];
        cout << "\n";
    }

    return 0;
}
```

V prípade úspešného načítania 2 obrázkov typu png nasleduje séria krokov spomenutých v podkapitole 3.3.2.

### 4.4.2 Hľadanie, deskripcia a spojenie kľúčových bodov

Najprv si náš program zvolí algoritmus SURF, ktorý sa použije na hľadanie a deskripciu kľúčových bodov. SURF sme si vybrali prevažne z dôvodu jeho invariantnosti voči rotácii a vzdialenosti kamery od zachytávaného objektu a taktiež aj kvôli jeho rýchlej deskripcii obrázku.

Následne dôjde k vypočítaniu pracovnej škály, pomocou ktorej dokážeme zmenšiť obrázky, s ktorými pracujeme, a tak docielime zvýšenie efektivity programu a znížime celkovú výpočtovú záťaž.

Vyžadovaný parameter, ktorý SURF potrebuje, je match_conf a táto hodnota udáva určitú istotu, či daný vybraný bod bude označený ako kľúčový, alebo nie. Čím vyššia táto hodnota je, tým menej kľúčových bodov algoritmus nájde, no je väčšia pravdepodobnosť, že budú označene správne.

Posledným krokom v tejto časti je zavolanie funkcie leaveBiggestComponent, ktorá spojí kľúčové body v prípade, že sa ich deskriptory rovnajú.

### 4.4.3 Získanie rotačnej matice a zakrivenie obrazu

V programe sa následne vytvorí približný odhad rotačnej matice zo spojených kľúčových bodov. Odhad rotácie je zrealizovaný 3x3 maticou homografie, ktorú získame prostredníctvom algoritmu RANSAC.

Pred samotnými krokmi zakrivenia a zarovnania obrazu sa ešte vykoná algoritmus BundleAdjusterRay, ktorý dokáže odhadnúť ohniskovú vzdialenosť a používa sa na spresnenie parametrov rotačnej matice, čím sa minimalizuje súčet vzdialeností medzi lúčmi prechádzajúcimi stredom kamery a kľúčovým bodom.

Finálnymi krokmi sú zakrivenie a zarovnanie obrazu, ktoré používajú rotačnú maticu s nami upravenými parametrami. Pri zakrivení obrazu sa vykonáva mapovanie našich bodov rotačnou maticou na prázdny obrázok s rovnakou veľkosťou, ako pôvodný obrázok. Do vektora corners zloženého z triedy typu Point, ktorá si ukladá informácie o pixeloch, sa taktiež ukladajú dáta popisujúce rohy vstupných obrázkov z dôvodu zmeny ich polohy pri mapovaní a ich potreby pri konečnom spájaní obrazu. Po zakrivení nám vznikne obraz, ktorý môžeme ďalej použiť pri zarovnaní.

### 4.4.4 Zarovnanie a exportovanie obrázkov

Na zarovnanie obrazu využívame triedu Blender typu MULTI_BAND, teda na zahladenie obrázkov použije multi-band algoritmus, ktorý najprv zakrivené obrázky a z nich vytvorené masky (čiernobiela verzia obrázkov) použije na vypočítanie Gaussovho rozdielu. Vstupom tohto algoritmu je zakrivený obraz, jeho maska a vektor, v ktorom sú uložené rohy pôvodných obrázkov. Následne sa masky 2 obrázkov orežú, spoja a na predošlé prislúchajúce miesta masiek sa vložia upravené ohnuté obrázky.

Posledným krokom je vloženie výsledného spojeného obrázku na stanovené miesto. V našom prípade posielame spojené obrázky do zložky Resources, ktoré využíva v Unity pri ďalšom spracovaní.

## 4.5 Unity projekt

V tejto časti si opíšeme druhú časť nášho systému realizovanú v Unity. Projekt je zložený z 2 scén. Prvá scéna sa používa na simuláciu snímania priestoru 2 stereoskopickými kamerami a druhá na zobrazenie už spojeného obrazu v Android smartfóne prostredníctvom Google Cardboardu.

### 4.5.1 SceneVR

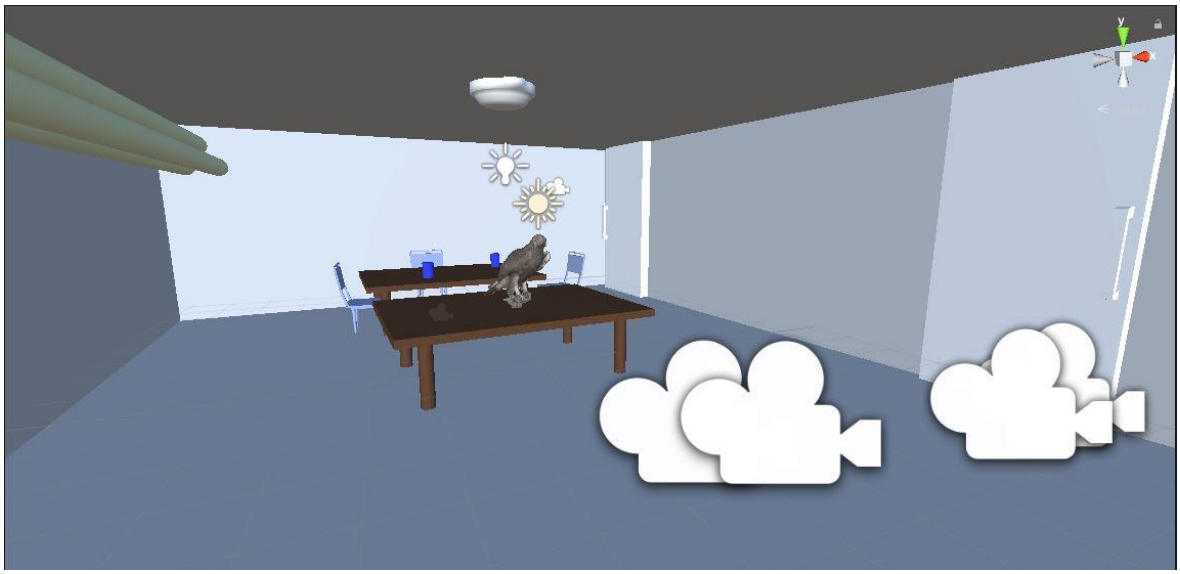
Scéna sceneVR je tvorená z miestnosti Submarine Room 1, ktorá je voľne dostupná na stránke sketchfab a jej autorom je tomkranis[22]. Tento 3D model miestnosti je zároveň chránený licenciou Creative Commons Attribution (CC BY 4.0)[23].

SceneVR je zložená z ďalších objektov, ktoré slúžia na vytvorenie osvetlenia a modelu orla, ktorý je taktiež voľne dostupný na stránke Sketchfab. Orol tvorí dynamický prvok scény, pretože má k sebe priradenú animáciu rotácie o 360°, čo nám umožní zachytiť ho zo všetkých strán.

V poslednom rade je scéna tvorená zo systému 4 kamier – 2 pre ľavé a 2 pre pravé oko, pomocou ktorých budeme vytvárať stereo obraz. Celý systém je priradený objektu MainCamera, na ktorý je aplikovaný C# skript MainCamera.cs.

V skripte MainCamera.cs máme 4 hlavné parametre. ResWidth a resHeight, prostredníctvom ktorých môžeme meniť rozlíšenie zachytávaného obrazu, name a counter, ktoré neskôr slúžia na vytvorenie unikátneho názvu vytvoreného súboru z dôvodu eliminácie nežiadúceho prepisovania už vytvorených obrázkov. Po spustení scény sa zavolá funkcia onCouroutine, ktorá v cykle volá funkciu takePics a na konci cyklu yield, ktorý slúži na korutinu, teda umožňuje Unity prevziať kontrolu, pokračovať s animáciou orla na nám stanovený čas a opäť sa vrátiť k vykonávaniu príkazov v cykle. Ak by sme túto korutinu v skripte nemali, najprv by zbehol cyklus, kde by boli všetky obrázky zachytené len v stacionárnej pozícii orla a po zbehnutí cyklu by orla začal rotovať. Pri funkcii onCoroutine je dôležité uviesť, že cyklus while je obmedzený na určitý počet obrázkov, ktoré sa zachytia, a to z dôvodu testovania správnej funkcionality a zbytočného zaťaženia výpočtovej techniky. Funkcia takePics, ktorá sa volá v cykle slúži na zachytenie obrázkov z kamier, ktoré sa zakódujú do typu PNG, zavolá sa funkcia ScreenShotName, kde sa na základe už skôr spomenutých parametrov, ktoré sa iterujú a upravujú, vytvorilo unikátne meno súboru

na uloženie spolu s cestou k nami určenému priečinku, kde máme uloženú aplikáciu na spájanie obrazu.



*Obrázok 10 Scéna sceneVR vytvorená v Unity*

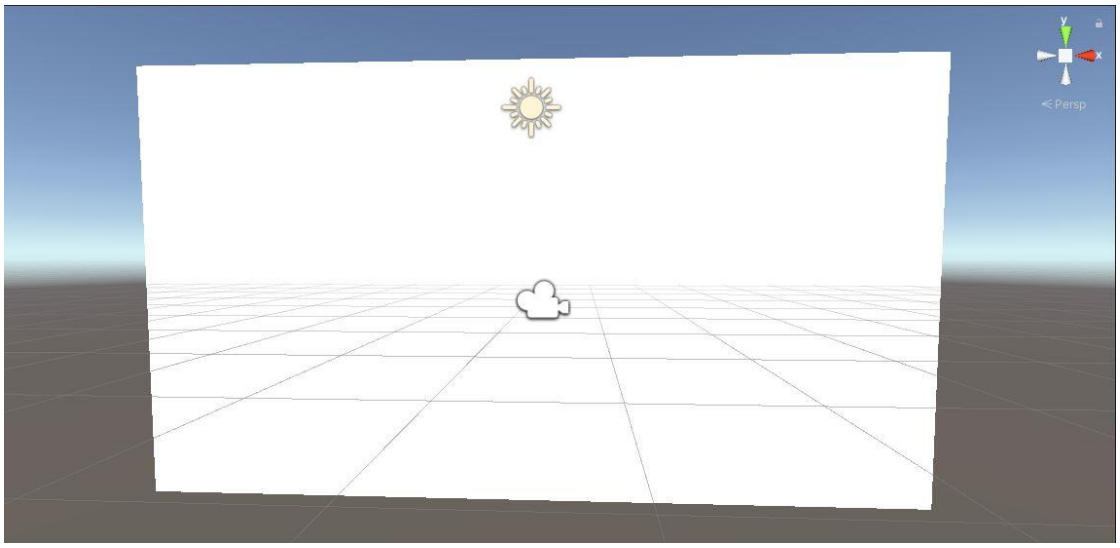
### 4.5.2 VR

Druhú nami vytvorenú scénu predstavuje scéna VR, ktorá je jednoduchšia a predstavuje poslednú fázu zobrazovania spojených obrázkov. Scéna sa skladá z rovín a kamier, ktoré ich sledujú.

Kamery určené pre pravé a ľavé oko sa nachádzajú v súradnicovom systéme na rovnakých pozíciách, pričom majú zmenšené zorné pole, aby sme sa mohli pohľadom voľne pohybovať v rámci roviny. Kamery používajú skripty CameraL.cs a CameraR.cs, ktoré sú mierne pozmenené v zmene v premennej counter, prostredníctvom ktorej sa načítava pozícia obrázku v zložke Resources. Pri týchto skriptoch taktiež potrebné pripomenúť, že z dôvodu testovania správnej funkcionality je cyklus while ohraničený polovičnou hodnotou celkového počtu vytvorených obrázkov v sceneVR, čím sme dosiahli nekonečné prehrávanie spojených obrázkov v cykle.

Roviny plainL a plainP sú taktiež umiestnené na rovnakých pozíciách a používajú skript Acceleration.cs s parametrom speed udávajúcim rýchlosť pohybu, akou sa môže rovina hýbať. Skript na základe získania parametrov pohybu z akcelerometra nachádzajúceho sa v smartfóne upravuje súradnicové umiestnenie roviny pri každom pohybe zariadenia, pričom pohyb v rámci osí x a y je limitovaný veľkosťou roviny.

Po zostavení, spustení a vložení Android smartfónu do Google Cardboardu scéna VR začne v cykle o veľkosti počtu obrázkov nachádzajúcich sa v zložke Resources načítavať ako textúry do rovín plainL a plainP zošité obrázky pre ľavé a pravé oko, ktoré môžeme stereoskopicky vnímať prostredníctvom kamier CameraL a CameraR.




*Obrázok 11 Scéna VR zobrazujúca na rovinu obraz pre pravé a ľavé oko*

# 5 Testovanie a výsledky

V tejto časti otestujeme našu aplikáciu na spájanie obrazu, ale aj celkovo celý systém. Testované boli rôzne konfigurácie a parametre, s ktorými náš systém pracuje. Výsledky zrealizovaných testov si popíšeme a porovnáme jednotlivo.

Systém bol testovaný na notebooku so špecifikáciou spomenutou v kapitole 3.2.

## 5.6 Test rýchlosti hľadania kľúčových bodov

Jeden z najdôležitejších krokov pri spájaní obrazu je hľadanie kľúčových bodov. Preto bolo dôležité zvoliť si taký algoritmus, ktorý najviac spĺňa naše požiadavky, a to, aby bol čo najrýchlejší a aby bol robustný voči zmenám uhlu a rotácii pohľadu. Na porovnanie sme použili všetky algoritmy, ktoré sme mali k dispozícií, a to SIFT, SURF, ORB, AKAZE. Hodnoty v tabuľke predstavujú priemernú hodnotu po 5 opakovaniach.

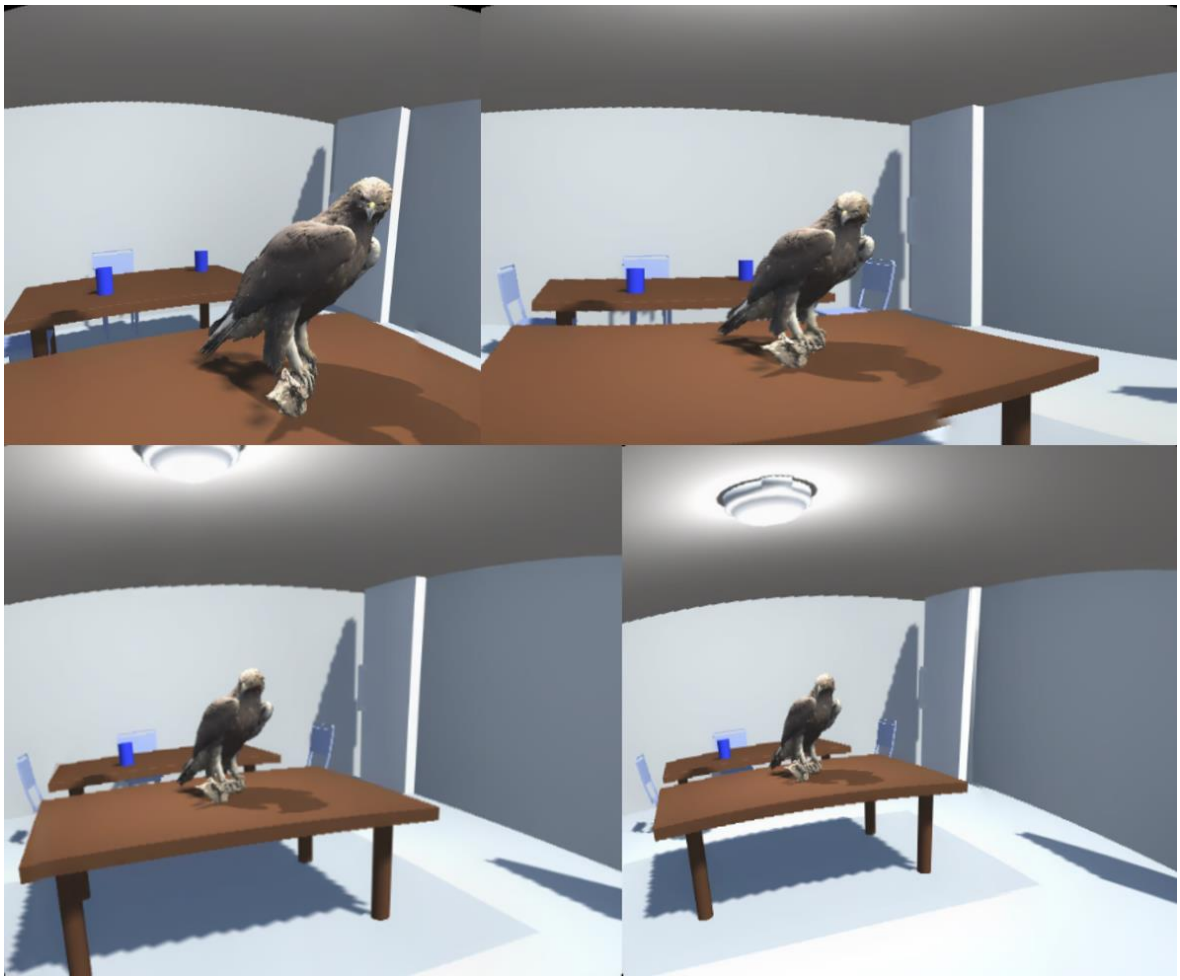
*Tabuľka 1 Porovnanie algoritmov*

| | Obrázok s rozmermi 500x400 | | Obrázok s rozmermi 4000x3000 | |
|---|---|---|---|---|
| Meno | Čas (ms) | Počet bodov | Čas (ms) | Počet bodov |
| SIFT | 1712 | 280 | 2219 | 505 |
| SURF | 1353 | 374 | 2133 | 997 |
| ORB | 1292 | 480 | 1764 | 490 |
| AKAZE | 1660 | 200 | 2185 | 247 |

Čas nájdenia kľúčových bodov bol spomedzi všetkých 4 algoritmov najrýchlejší pri použití algoritmu ORB, no napriek tomu sme si zvolili SURF, a to z dôvodu invariantnosti voči vzdialenosti, ktorú ORB nemá.

## 5.7 Test spájania obrazu z odlišných vzdialeností

Pri spájaní obrazu bolo dôležité určiť optimálnu vzdialenosť, pri ktorej naša aplikácia dokáže spojiť obraz z 2 alebo viacerých kamier bez viditeľnej chybovosti. Veľkým problémom, ktorý sa pri znižovaní vzdialenosti pozorovaného objektu a viacerých objektívov objavuje, je tzv. paralaxa, teda ak pozorujeme vzdialený predmet, jeho posun pri pozorovaní ľavým a pravým okom je minimálny, no ak je predmet blízko, jeho posun je v porovnaní s pohľadom z pravého a ľavého oka väčší. Porovnali sme výsledky spojenia obrázkov z rôznych vzdialeností a vyhodnotili ich.



*Obrázok 12 Porovnanie spojených obrázkov z malej a väčšej vzdialenosti*

Rozdiel medzi spojenými obrázkami z malej a väčšej vzdialenosti je minimálny z veľkej miery z dôvodu rovnakého uhlu pri vytváraní obrázkov. Chybovosť spojenia obrázkov je pri menšej vzdialenosti od pozorovaného objektu väčšia, čoho dôkazom je v obrázku 10 obrázok vpravo hore, kde bol chybne spojený stôl, ktorý sa nachádza k objektívu bližšie než orol.

Na zníženie chybovosti pri spájaní obrázkov z 2 a viacerých kamier pri pozorovaní objektu je potrebné, aby bol kamerový systém vzdialený od objektu v súradnicovom systéme aspoň o číselnú hodnotu 5 na osi Z.

# 6 Záver

V tejto práci sa nám podarilo úspešne zrealizovať scénár, kde sme sa pokúšali v zjednodušenej verzii vytvoriť snímanie priestoru stereoskopickou kamerou, ktoré sme simulovali 2 dvojicami stereokamier. Snímali sme dynamickú scénu, a tú sme spájali pre ľavé a pravé oko zvlášť. Následne je táto scéna zobrazovaná cez Android smartfón do okuliarov Google Cardboard, v ktorých si ju dokážeme prezerať stereoskopicky.

Problémom, ktorý sa vyskytol pri práci, bol v hardvéri, ktorý nebol dostatočne prispôsobený na online prenos a teda hardware nepodporoval taký typ komunikácie.

V budúcnosti by sme tento systém vedeli vylepšiť o real-time prenos zachyteného a spojeného obrazu do zobrazovacieho hardvéru s minimálnym oneskorením. Taktiež by tento systém mohol byť obohatený o zachytávanie polohy a smeru pohľadu, kam sa pozeráme. V aplikácii na spájanie obrazu by sme mohli pridať rozšírenie o CUDA, ktorou by sa časť výpočtov presunula na GPU a tak by dokázala mnohonásobne urýchliť proces spájania obrazu.

Počítačové spájanie obrazu je nesmierne výpočtovo náročný proces, ktorý stále aj po mnohých rokoch predstavuje ohromnú výzvu v oblasti počítačového videnia. Tento proces možno neustále upravovať a zdokonaľovať novými technológiami, algoritmami alebo lepším hardvérom, čím sa nám otvárajú dvere k novým poznatkom a spôsobom ich aplikácie v skutočnom, ale aj virtuálnom svete.

# Zoznam použitej literatúry


[1] SZELINSKI, Richard. Image Alignment and Stitching: A Tutorial. In *Foundations and Trends in Computer Graphics and Vision* [online]. 2006, 2(1), 18-29 [cit. 2021-01-08]. Dostupné z: doi:10.1561/0600000009

[2] KALE, Pranoti a K. R. SINGH. A Technical Analysis of Image Stitching Algorithm. In *International Journal of Computer Science and Information Technologies* [online]. 2015, 6(1), 284-288 [cit. 2021-01-08]. ISSN 0975-9646. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.666.3271&rep=rep1&type=pdf

[3] ADEL, Ebtsam, Mohammed ELMOGY a Hazem ELBAKRY. Image Stitching based on Feature Extraction Techniques: A survey. In *International Journal of Computer Applications* [online]. 2014, 99(6), 1-8 [cit. 2021-01-10]. ISSN 0975-8887. Dostupné z: DOI:10.5120/17374-781

[4] SAKHARKAR, Ms. Vrushali S. a Prof. Dr. S. R. GUPTA. Image Stitching Techniques-An overview. In *International Journal Of Computer Science And Applications* [online]. 2013, 6(2), 324-330 [cit. 2021-01-17]. ISSN 0974-1011. Dostupné z: http://researchpublications.org/IJCSA/NCAICN-13/230.pdf

[5] BIADGIE, Yenewondim a Kyunh-Ah SOHN. Feature Detector Using Adaptive Accelerated Segment Test. In *International Conference on Information Science & Applications (ICISA)* [online]. 2014, 1-4 [cit. 2021-01-18]. ISSN 2162-9048. Dostupné z: doi:10.1109/ICISA.2014.6847403

[6] ZHENG, Jin. et al. An Accurate Multi-Row Panorama GenerationUsing Multi-Point Joint Stitching. In *IEEE Access* [online]. IEEE, 2018(6), 27827-27839 [cit. 2021-01-21]. Dostupné z: doi:10.1109/ACCESS.2018.2829082

[7] LOWE, David G. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision* [online]. IEEE, 1999(2), 1150-1157 [cit. 2021-01-23]. Dostupné z: doi:10.1109/ICCV.1999.790410

[8] HEYMANN, S. et al. SIFT Implementation and Optimization for General-Purpose GPU. In *WSCG '2007: Full Papers Proceedings: The 15th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2007 in co-operation*

*with EUROGRAPHICS: University of West Bohemia Plzen Czech Republic* [online]. 2007,317-322[cit. 2021-01-23]. Dostupné z: https://dspace5.zcu.cz/handle/11025/11026

[9] BAY, Herbert. et al. Speeded-Up Robust Features (SURF*).* In *Computer Vision and Image Understanding* [online]. Elsevier, 2008, 110(3), 346-359 [cit. 2021-01-24]. ISSN 1077-3142. Dostupné z: doi:10.1016/j.cviu.2007.09.014

[10] AGUADO, Alberto S. a Mark S. NIXON. Feature Extraction and Image Processing for Computer Vision [online]. 4. London: Elsevier, 2019 [cit. 2021-01-24]. ISBN 978-0-12-814976-8. Dostupné z: https://doi.org/10.1016/B978-0-12-814976-8.00001-4

[11] AGLAVE, Prashant a Vijaykumar S. KOLKURE. IMPLEMENTATION OF HIGH PERFORMANCE FEATURE EXTRACTION METHOD USING ORIENTED FAST AND ROTATED BRIEF ALGORITHM. *IJRET: International Journal of Research in Engineering and Technology* [online]. 2015, 4(2), 394-397 [cit. 2021-03-24]. ISSN 2319-1163. Dostupné z: http://www.ijret.org

[12] RUBLEE, Ethan. et. al. ORB: an efficient alternative to SIFT or SURF. *2011 IEEE International Conference on Computer Vision* [online]. 2011, 2564-2571 [cit. 2021-03-24]. Dostupné z: doi:10.1109/ICCV.2011.6126544.

[13] CALONDER, Michael. et. al. BRIEF: Binary Robust Independent Elementary Features. *Eur. Conf. Comput. Vis.* [online]. 2010, 6314, 778-792 [cit. 2021-03-24]. Dostupné z: doi:10.1007/978-3-642-15561-1_56

[14] 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*. ACM New York, NY, USA, 1981, 24(6), 381-395.

[15] DERPANIS, Konstantinos G. Overview of the RANSAC algorithm. *Image Rochester NY.* 2010, 4(1), 2-3.

[16] GLASBEY, C. A. a K. V. MARDIA. A review of image-warping methods. *Journal of Applied Statistics* [online]. 1998, 25(2), 155-171 [cit. 2021-04-07]. Dostupné z: doi:10.1080/02664769823151

[17] WEISSTEIN, Eric W. Affine transformation. *Https://mathworld.wolfram.com* [online]. 2004 [cit. 2021-04-07]. Dostupné z: https://mathworld.wolfram.com/AffineTransformation.html

[18] WEISSTEIN, Eric W. Perspective. *Https://mathworld.wolfram.com* [online]. [cit. 2021-04-07]. Dostupné z: https://mathworld.wolfram.com/Perspective.html

[19] ADEL, Ebtsam, Mohammed ELMOGY a Hazem ELBAKRY. Image Stitching System Based on ORB Feature-Based Technique and Compensation Blending. *International Journal of Advanced Computer Science and Applications* [online]. Citeseer, 2015, 6(9), 55-62 [cit. 2021-04-07]. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.695.5230&rep=rep1&type=pdf

[20] JUAN, Luo a Gwun OUBONG. SURF applied in panorama image stitching. *2010 2nd International Conference on Image Processing Theory, Tools and Applications* [online]. IEEE, 2010, 2010, 495-499 [cit. 2021-4-29]. ISBN 978-1-4244-7247-5. Dostupné z: doi:10.1109/IPTA.2010.5586723

[21] HUANG, Fay a Reinhard KLETTE. Stereo panorama acquisition and automatic image disparity adjustment for stereoscopic visualization. Multimedia Tools and Applications [online]. 2010, 47(3), 353-377 [cit. 2021-5-3]. ISSN 1380-7501. Dostupné z: doi:10.1007/s11042-009-0328-2

[22] TOMKRANIS. Submarine Room 1. *Sketchfab* [online]. [cit. 2021-6-10]. Dostupné z: https://sketchfab.com/3d-models/submarine-room-1-b0ec2563bca6436e989f161470a9c8ff

[23] Creative Commons [online]. [cit. 2021-6-2]. Dostupné z: https://creativecommons.org/licenses/by/4.0/deed.cs

# Prílohy

Príloha A: Štruktúra elektronického nosiča

Príloha B: Návod na inštaláciu programov

Príloha C: Používateľská príručka

# Príloha A: Štruktúra elektronického nosiča

V prílohe, ktorá sa nachádza v Akademickom informačnom systéme (AIS) sú priložené:

- Bakalárska práca v elektronickej forme ako súbor PDF
- Zip súbor, v ktorom sa nachádza Unity projekt s C# skriptami a Visual studio projekt s kódom aplikácie Image stitcher a samotný spustiteľný súbor aplikácie na spájanie obrazu ImageStitcher

# Príloha B: Návod na inštaláciu programov

Pre vlastnú kompiláciu a prípadnú úpravu aplikácie ImageStitcher je potrebné mať nainštalovaný program Visual Studio, ktorý je voľne dostupný na web stránke spoločnosti Microsoft. Ďalej je potrebné stiahnuť OpenCV verziu 4.1.1, OpenCV modul contrib, nainštalovať program CMake, prostredníctvom ktorého ich spojíme dokopy, avšak je dôležité počas konfigurácie urobiť ENABLE_NONFREE, čím si povolíme prístup k algoritmom ako SIFT alebo SURF. V Properties projektu ImageStitcher je potrebné nasledovne v časti C/C++ nastaviť cestu k zložke include, ktorá by sa mala nachádzať v zložke, ktorú vytvoril CMake. Poslednou úpravou v Properties je kliknutie na časť Linker a v časti Additional Library Directories zmena cesty na cestu k zložke lib, ktorá sa nachádza v rovnakej zložke vytvorenej programom CMake.

Pre prácu s Unity je potrebné si stiahnuť Unity verziu 2019.4.20, ktorá má Android Build Support a Android SDK & NDK Tools.

Pre vytvorenie aplikácie v smarfóne je potrebné otvoriť File, Build Settings a zmeniť platformu na Android. Pre overenie je potrebné kliknúť na tlačidlo Edit vľavo hore, Preferences a uistiť sa, že všetky zaškrtávacie políčka v sekcii Android sú zaškrtnuté. Poslednou podmienkou pre úspešné vytvorenie aplikácie v smartfóne je spojenie smartfónu a počítača.

# Príloha C: Používateľská príručka

Pred samotným spustením aplikácie na spájanie obrazu a scén v Unity projekte je potrebné urobiť inštaláciu a konfiguráciu programov, ktorá je opísaná v prílohe B. Po vykonaní potrebnej inštalácie a konfigurácie urobiť 2 hlavné nastavenia ciest.

V kóde pre program na spájanie obrazu je potrebné upraviť na riadku 85 pôvodnú cestu k zložke Resource Unity projektu na cestu na miesto, kde ste si uložili Unity projekt. Dôležité je odkázať sa na zložku Resources, ktorá sa používa pri zobrazovaní zošitých obrázkov. V Unity projekte je potrebné na riadku 30 taktiež zmeniť pôvodnú cestu na cestu, kde máte uloženú aplikáciu ImageStitcher.exe.

Po nastavení ciest môžete spustiť aplikáciu ImageStitcher.exe, ktorá bude čakať na vstupné obrázky poslané z Unity. Následne otvorte Unity projekt panorama, otvorte scénu sceneVR a spustite. Do zložky projektu Resources by sa mali uložiť zošité obrázky.

Po zastavení scény je potrebné zmeniť scénu na VR, zapojiť Váš Android smartfón do počítača, kliknúť na File vľavo hore a následne kliknúť na Build and Run, čím sa vám vytvorí v smartfóne aplikácia, ktorá po spustení Vám bude zobrazovať nahranú scénu.