VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

HYBRIDNÍ APLIKACE PRO KALIBRACI KAMEROVÝCH ČOČEK

HYBRID APPLICATION FOR CALIBRATION OF CAMERA LENSES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **DMYTRO DOVHALENKO**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VÁCLAV ŠIMEK**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

148969

Ústav:	Ústav počítačových systémů (UPSY)
Student:	**Dovhalenko Dmytro**
Program:	Informační technologie
Specializace:	Informační technologie
Název:	**Hybridní aplikace pro kalibraci kamerových čoček**
Kategorie:	Zpracování obrazu
Akademický rok:	2022/23

Zadání:

1. Seznamte se s problematikou a s nástroji na potlačení zkreslení obrazu vlivem zakřivení kamerových čoček.
2. Zabývejte se oblastí tzv. hybridních webových aplikací. Proveďte rešerši možností jejich realizace.
3. S ohledem na bod 2 analyzujte přístup a technologie použité u NXP nástrojů jako je například Audio Tuning Tool či Voice Recognition Tool.
4. Na základě poznatků z bodu 1 zvolte nástroj pro korekci zkreslení, pro nějž definujte uživatelské prostředí zaměřené na možnosti konfigurace a procesu kalibrace.
5. S ohledem na poznatky z bodů 1 až 3 popište možné scénáře užití aplikace a navrhněte pro ně vhodnou architekturu. V návrhu popište výhody a nevýhody každého scénáře.
6. Na základě bodu 5 implementujte nejvýhodnější scénář užití navržené aplikace.
7. Vhodným způsobem ověřte vlastnosti realizovaného řešení na sadě konkrétních úloh z praxe.
8. Zhodnoťte dosažené výsledky a diskutujte možnosti pokračování projektu.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:
Splnění bodů 1 až 4 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Šimek Václav, Ing.**
Vedoucí ústavu:	Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání:	1.11.2022
Termín pro odevzdání:	10.5.2023
Datum schválení:	31.10.2022

Abstrakt

Cílem teto bakalářské práce je navrhnout a realizovat nástroj pro korekci zkreslení obrazu způsobeného fyzickým zakřivením kamerové čočky typu rybího oka. V teoretické častí se věnuje analýze dostupných řešení a snaží se najít způsob, jak realizovat nástroj pro korekci zkreslení obrazu za využití moderních technologií. Použitelné technologie se analyzuji a popisuji a hledá se alternativní pohled který umožní změnit přístup v dodávce softwaru a dodat nástroj ve formě webové aplikace. Práce je specifická nutností dosažení cílů pro nasazení a budoucí vývoj v korporátní prostředí, a proto tyto potřeby analyzuje a přizpůsobuje se jim. Zaměřuje se na problematiku generování nastavení pro hardwarově akcelerovanou korekci obrázků, protože se v praxi využívá častěji než softwarová korekce. Praktická část je zaměřená na analýzu scénářů užiti, návrh vhodné architektury a uživatelského prostředí. K vyřešení problému využívá hybridní webové technologií v kombinaci s dodávkou knihovny OpenCV ve formě webassembly, a další moderní nástroje jako SSO autentizace a vestavěné úložiště prohlížeče. Práce je přínosná pro reálné zákazníky, kteří potřebují moderní řešení nástroje pro kalibraci zkreslení obrazu. Dokazuje, že využitím hybridních webových aplikací je možné dodat ekvivalentní náhradu za nativních aplikace a současně s tím navýšit uživatelský komfort a rovněž poskytuje důležité informace pro možnost transformace nástrojů z nativních aplikací do hybridních webových.

Abstract

The goal of this bachelor's thesis is to design and implement a tool for correction of image distortion caused by the physical curvature of a fisheye camera lens. The theoretical part is concerned with available solutions analysis in order to find an image distortion correction tool that uses modern technologies. The technologies are analyzed and described to find alternative points of view resulting in a way to change software delivery process and deliver a tool in the form of a web application. The thesis takes into account the necessity to deliver a tool that can be developed and deployed in a context of corporate environment. It focuses on the issue of generating settings for hardware-accelerated image correction since it is used more frequently in the real world than its software counterpart. The practical part analyzes various usage scenarios, suitable architecture design as well as user environment design. To achieve this, hybrid web technologies in combination with OpenCV library in the form of a web assembly, as well as other modern tools such as SSO authentication and browser's built-in storage are used. This thesis will benefit mostly customers in need of modern image distortion calibration solutions. It proves that it is possible to use hybrid web applications to deliver native application alternative that simultaneously increases user comfort. The thesis also provides data important for transformation of native applications into hybrid web applications.

Klíčová slova

hybridní aplikace, kamerové čočky typu rybího oka, kalibrace kamery, korekce snímků, zkreslení obrazů, OpenCV, webassembly

Keywords

hybrid applications, fisheye camera lenses, camera calibration, image correction, image distortion, hybrid applications, OpenCV, web assembly

Citace

DOVHALENKO, Dmytro. *Hybridní aplikace pro kalibraci kamerových čoček*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Václav Šimek

Hybridní aplikace pro kalibraci kamerových čoček

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Václava Šimka. Další informace mi poskytl vedoucí týmu firmy NXP Jakub Cieslar. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Dmytro Dovhalenko
8. května 2023

Poděkování

Na tomto místě rád bych poděkoval vedoucího svého týmu ve firmě NXP Jakuba Cieslara, za odborné konzultace, rady a pomoc během celého cyklu vývoje teto práce.

Rovněž bych chtěl poděkovat Ing. Václava Šimka za cenné rady při vytvoření technické zprávy.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Při snímání obrazu lze použít spoustu různých čoček, a to bud' čočky s přiblížením, širokoúhlé objektivy, což pomůže zachytit větší část scény, kterou snímáte, nebo čočky typu rybího oka, které nabízí obrovské zorné pole. Ovšem tyto čočky můžou také zkreslovat, zejména kolem okrajů obrazu. Tento projekt bude zaměřený na práci s objektivem typu rybího oka. Snímání obrazu z takové čočky trpí na zkreslení způsobené fyzickým zakřivením čočky. Výsledkem je, že kamera deformuje obraz roztažením kolem zaoblené čočky, což zvětšuje velikost objektu ve středu fotografie, rovné čáry vypadají jako zakřivené nebo ohnuté a rohy obrazu můžou být ztmavené. Použití obrázku s takovým efektem v dalších aplikacích, které využívají tato data, je nevhodné bez provedení odpovídající korekce snímků, jež odstraní účinky geometrického zkreslení způsobeného objektivem fotoaparátu, proces odstranění zkreslení dále je pojmenovaný jako dewarping. Pro dewarping se používají různé metody matematické korekce zkreslených obrázků. Výsledkem je obraz s narovnanými liniemi a objekty, které vypadají přirozeně.

Existují dva hlavní typy dewarpingu obrázku z kamery typu rybího oka: hardwarový a softwarový dewarping.Při použití hardwarového dewarpingu korekce probíhá pomocí specializovaného hardwaru zabudovaného ve fotoaparátu nebo samostatného obrazového procesoru. Výhodou provedení hardwarového dewarpingu je odstranění zkreslení v reálném čase, což znamená, že opravený obraz je k dispozici okamžitě bez nutnosti následného zpracování. Kvůli rychlosti zpracování může být tento typ dewarpingu užitečný zejména pro aplikace, které vyžadují rychlé zpracování velkého množství dat. Ovšem před samotným zpracováním je nutné vyřešit nastavení kalibrace, získat vnější a vnitřní parametry kamery, zjistit hodnoty zakřivení způsobeného kamerovou čočkou pomocí softwarových nástrojů a vytvořit odpovídající způsob korekcí těchto efektů pomocí zabudovaného hardwaru kamery. Odstranění zkreslení hardwaru může být drahé, protože vyžaduje zabudování specializovaného hardwaru do kamery a může vyžadovat značný výpočetní výkon, který může omezit snímkovou frekvenci nebo rozlišení výsledného obrazu. Použití hardwarového dewarpingu se ale stejně vyplatí, jelikož výsledný algoritmus potřebuje jenom zpracovávat vstupní obrázky podle zjištěných nastavení a nemusí pokaždé vypočítávat tato nastavení. Samotné zpracování obrázku bude zabírat minimum času, tedy mnohem méně než při použití nějakého softwaru, což je hlavní výhodou takového přístupu.

Cílem projektu je poskytnout nutné informace pro provedení hardwarového dewarpingu s použitím softwarového dewarpingu. Tento typ odstranění zakřivení obrázků je založený na kompenzaci zkreslení způsobem aplikace matematických transformací pro odstranění zakřivení v určité oblasti. Zadáním diplomové práci je vytvořit aplikaci, která by mohla odstranit vady obrázků, vypočítat parametry zkreslení čočky kamery a poskytnout tyto informace pro

použití v dalších nástrojích, které umožní provést dewarping pomocí vestavěných součástek hardwarově. Jednou z běžných technik pro softwarový dewarping je použití vzoru šachovnice. Tato technika zahrnuje umístění šachovnicového vzoru do zorného pole kamery a jeho použití k odhadu vnitřních a vnějších parametrů kamery stejně jako koeficientů zkreslení. Ke kalibraci bude uživatel potřebovat pouze zachytit sérii snímků šachovnicového vzoru z různých úhlů a vzdáleností a zadat informace o velikosti šachovnice, které jsou snadno dostupné. Algoritmus dewarpingu analyzuje vzor šachovnice a vypočítá parametry zkreslení kamery, jež popisují, jak čočka fotoaparátu deformuje obraz v závislosti na poloze kalibračního vzoru v obrazu. Pomocí těchto parametrů je možné korigovat zkreslení objektivu, aby byl zpracovaný obraz optimalizován pro zamýšlenou aplikaci. Podrobněji o postupu provedení softwarového dewarpingu v kapitole 3.1.

Diplomová práce popisuje nástroje a přístupy pro vytvoření hybridní webové aplikace, která umožní zpracování obrázků a dewarping s provedením většiny operací pomocí vestavěné funkcionality moderních prohlížečů. Popis technologií v kapitolách 6.2, 6.5.

V kapitole 7.2. popsaná další možná rozšíření zaměřená na připojení a nastavení výpočetních součástek jako čipy kamery přímo pomocí této aplikace pro zvýšení rychlosti a ulehčení práce s obrázky.

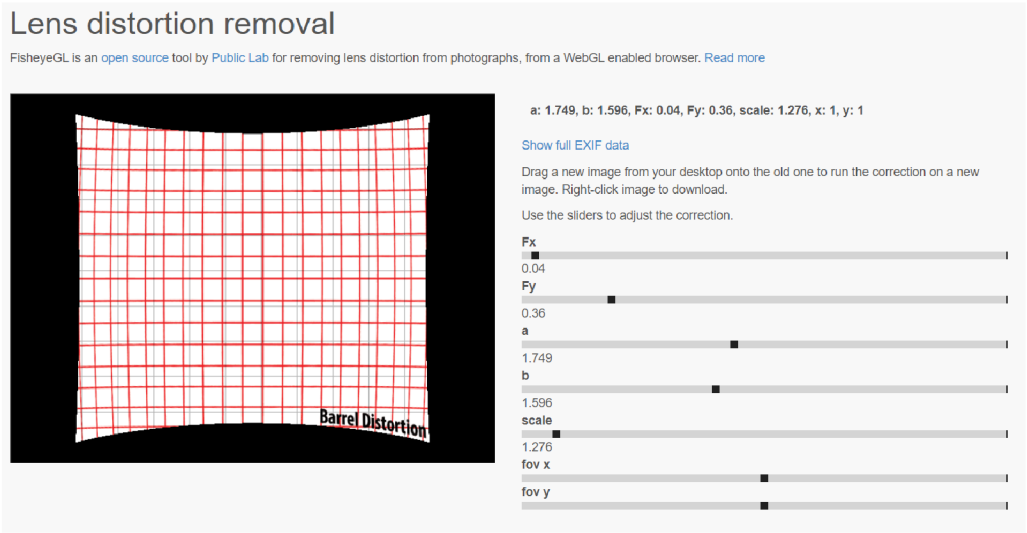
Kapitola 2

Existující nástroje pro odstranění zkreslení kamerových čoček

Momentálně existují aplikace, které můžou pomoct odstranit zkreslení obrázků vytvořených pomocí kamery typu rybího oka. Většina z nich ale nepoužívá žádné algoritmy, ale jenom poskytuje uživateli grafické rozhraní, ve kterém je možné ručně odstraňovat zakřivení. Pro efektivní řešení problému a provedení automatické opravy hned po vytvoření snímku to nestačí, proto je nutné využít jeden z automatizovaných algoritmů pro detekce zkreslení a korekce obrázků. Dále jsou uvedené některé z existujících nástrojů pro korekce obrázků a postupy práce s nimi.

2.1 FisheyeGL

FisheyeGL (obrazek 2.1) je nástroj s otevřeným zdrojovým kódem od Public Lab pro odstranění zkreslení čočky z fotografií.



Obrázek 2.1: Nástroj FisheyeGL

Pomocí posuvníků může tato stránka odstranit efekt "rybího oka", i když pro jeho dosažení také ořízne rohy obrázků. Hlavní nevýhoda tohoto nástroje spočívá v tom, že uživatel

musí ručně měnit nastavení, což negarantuje dosažení vhodného výsledku pro použití v dalších aplikacích.

Aplikace je v podobě webové stránky a napsaná pomocí JavaScript s použitím WebGL.

WebGL (Web Graphics Library) je JavaScript rozhraní pro programování aplikací (API) a umožňuje vytvářet aplikace, které využívají funkcionality jiných aplikací nebo služeb, a to bez potřeby znát detaily jejich implementace. Vyžívá se pro vykreslování interaktivní 3D a 2D grafiky v libovolném webovém prohlížeči. Vnitřní implementace tohoto nástroje spočívá v tom, že pomocí posuvníků mění nastavení nahraného obrázku a zobrazuje výsledek. K tomu nepoužívá žádné metody pro odhalení a odstranění zakřivení zkreslení.

2.2 PhotoWorks

PhotoWorks je desktopová aplikace, která má funkcionalitu pro opravu zdeformovaných snímků (obrázek 2.2). Aplikace nabízí možnost práce s několika typy zakřivení:

Obrázek 2.2: Náhled na aplikace PhotoWorks

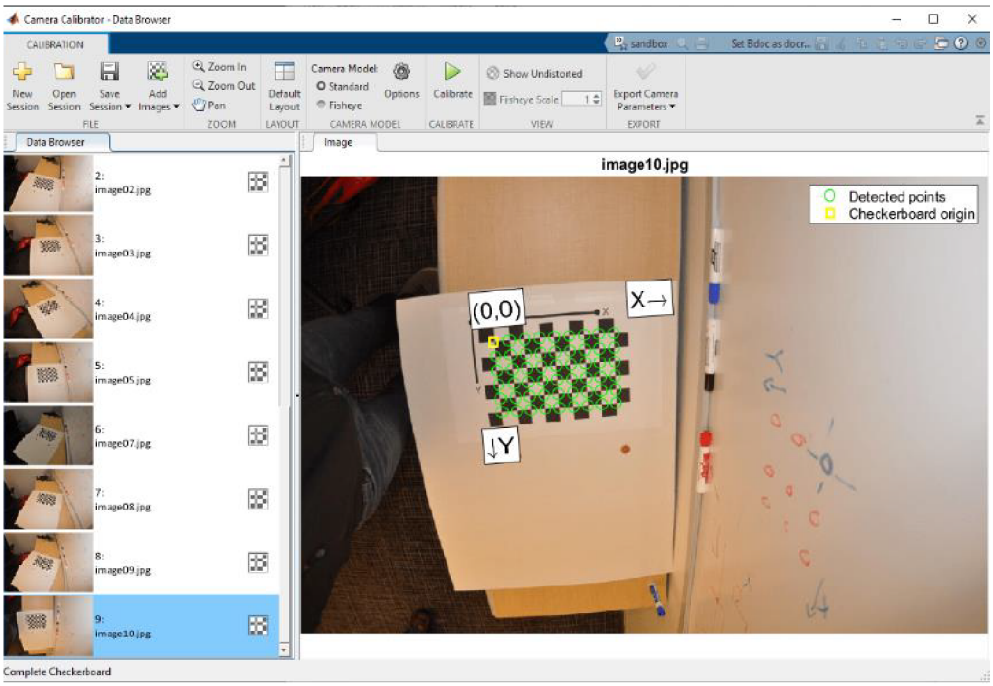
- Negativní zkreslení (Barrel distortion) - čáry jsou zakřivené dovnitř. Ve středu fotografie linie můžou vypadat zcela přirozeně, ale postupně se ohýbají směrem k okrajům rámu.
- Pozitivní zkreslení (Pincushion Distortion). V tomto případě se rovné čáry zakřivené směrem ven - od středu fotografie k její rohům. Čím dále je objekt od objektivu fotoaparátu, tím viditelnější je tento efekt.
- Zkreslení perspektivy – nepřiměřené objekty. Pokud je fotografie pořízená příliš blízko objektu, který je focený, může tento objekt vypadat nepřiměřeně (příliš velký nebo zkreslený) ve srovnání s jinými objekty v pozadí.

PhotoWorks poskytuje tuto funkcionalitu v podobě rozhraní, které dovoluje zapnout mřížku a ovládat zakřivení obrázku ručně pomocí posuvníků. Aplikace nepoužívá žádné metody pro odhalení a odstranění zkreslení, před použitím potřebuje stažení a instalaci a je placená.

2.3 Camera Calibrator

Camera Calibrator (obrázek 2.3) umožňuje odhadnout vnitřní a vnější vlastnosti a parametry zkreslení objektivu. Tyto parametry kamery lze použít pro různé aplikace počítačového vidění.

Aplikace je součástí Matlabu. Matlab je interaktivní výpočetní prostředí a programovací jazyk používaný především v oblasti numerických výpočtů, vizualizace dat a vědeckého programování. Matlab kombinuje výpočetní funkce, vizualizace a programování do jednoho prostředí. Programování v Matlabu se provádí pomocí skriptů a funkcí psaných v jazyce Matlab, který má syntaxi podobnou jazyku C.



Obrázek 2.3: Náhled na uživatelské rozhraní nástroje Camera Calibrator

Tato aplikace pro zjištění parametrů zkreslení se spouští z příkazového řádku Matlabu nebo jeho grafického rozhraní se zadáním příslušných parametrů pro odstranění zakřivení obrázků a odhadu parametrů zkreslení objektivu, například obrázky, které je nutné změnit, nebo fotografie šachovnice a reálná velikost šachovnicových čtverců obsažených na obrázku. Princip pro odstranění zkreslení čočky z obrázku použitý touto aplikací (stejně i pro čočky typu rybího oka) spočívá v tom, že potřebujete detekovat šachovnicový kalibrační vzor a zkalibrovat kameru podle zjištěných bodů šachovnice a předem zadaných reálných rozměrů. Aplikace zjišťuje vzorové body z obrázků šachovnice, vypočítá vlastnosti zkreslení a odstraňuje zakřivení, podrobněji v kapitole 3.1, ale před použitím musí být stažený a nainstalovaný Matlab Toolstrip a ovládání je dost složité a neintuitivní.

2.4 Knihovny a nástroje pomáhající usnadnit práce s obrázky

Jako výsledek rešerše existujících řešení pro kalibrace obrázků je, že většina aplikací nepoužívá žádné algoritmy pro automatizované odstranění vad obrázků a nalezení koeficientů zkreslení. Z představených řešení jenom poslední nástroj (2.3) využívá tyto možnosti. Ovšem má problémy z pohledu návrhu uživatelského rozhraní, práce s aplikací je dost složitá a před použitím je potřeba stáhnout a zprovoznit aplikace na vlastním stroji. Cílem bakalářské práci je využít technologii k tomu, aby se zlepšil postup ovládání podobným nástrojem a odstranila se nutnost stažení a instalace bez ztrát z pohledu výkonu aplikace. Využité postupy a technologie v kapitole 6 .

Záměrem je automaticky zpracovávat obrázky a dostávat na výstupu nastavení, která lze použít k automatické opravě zkreslení na všech snímcích pořízených fotoaparátem. Pro tyto účely je možné použití knihoven a nástroje, které poskytují možnost detekovat a opravovat zkreslení obrazu.

- OpenCV - softwarová knihovna pro počítačové vidění a strojové učení s otevřeným zdrojovým kódem. Softwarová knihovna je sbírka předem napsaných a přeložených funkcí. Použití knihovny může výrazně usnadnit a urychlit proces vývoje softwaru tím, že poskytuje připravený kód pro opakované úkoly a umožňuje se zaměřit na vývoj vlastního kódu. Knihovna OpenCV poskytuje algoritmy zaměřené na práci s obrázky a s kamerou typu rybího oka, které můžou pomoct s vyřešením daného problému.

- Computer Vision Toolbox - komerční sada nástrojů, která poskytuje algoritmy, funkce a aplikace pro navrhování a testování systémů počítačového vidění a zpracování videa. Umožňuje také automatizovat pracovní postupy kalibrace pro kamery s jedním objektivem, stereo kamery a kamery s rybím okem.

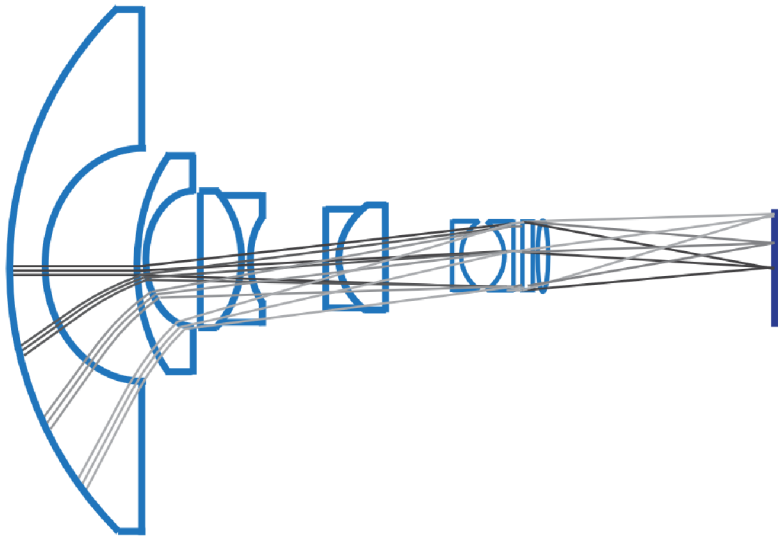
- ShiftN – software pro automatickou korekci zkreslení objektivu. Nejprve ShiftN hledá rovné čáry a hrany v obraze a ty, které jsou dostatečně vertikální, budou použity jako vzor. Poté software spustí proces optimalizace, který se pokusí určit perspektivu a opraví obraz tak, aby byly čáry rovnoběžné.

Za lepší řešení pro daný problém uvažuji použití knihovny OpenCV. Důvodem je hlavně to, že je to nástroj s otevřeným zdrojovým kódem, který nabízí různé metody opravy zkreslení obrázku. Jelikož to není koncový software, neomezuje uživatele v tom, co může dostat na výstupu po zpracování obrázku. Podrobněji o využití knihovny v kapitole 3.1.

Kapitola 3

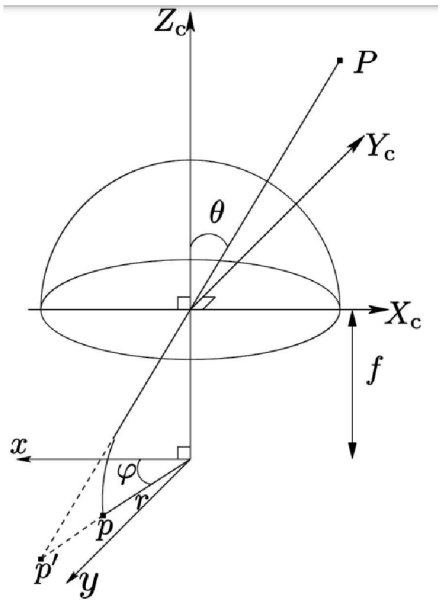
Detekce a odstranění zkreslení obrazů

Aby kamera mohla zachytit mnohem větší oblast scény, využívají objektivy čočky typu rybího oka a jsou složené ze sady různých čoček pro zvětšení zorného pole kamery, jak ukazuje obrázek 3.1. Čočky však dosahují tohoto extrémně širokoúhlého pohledu zkreslením linií perspektivy na snímcích.



Obrázek 3.1: Schemat objektivu typu rybího oka.
Převzato z [5]

Vzhledem k tomu, že model perspektivní projekce není vhodný pro rybí oko čočky, existuje flexibilnější, radiálně symetrický projekční model (obrázek 3.2).



Obrázek 3.2: Rozdil modelu perspektivní projekce od modelu kamery typu rybího oka
Převzato z [1]

Kde projekce bodu P je v bodu p pro kameru typu rybího oka namísto bodu p' v modelu perspektivní projekce.

Pro provedení kalibrace podle [1] je potřeba vytvořit model kamery, který bude zahrnovat zkreslení v radiálním směru:

$$\Delta_r(\theta,\varphi) = (l_1\theta + l_2\theta^3 + l_3\theta^5)(i_1 cos\varphi + i_2 sin\varphi + i_3 cos2\varphi + i_4 sin2\varphi) \tag{3.1}$$

a zkreslení v tangenciálním směru:

$$\Delta_t(\theta,\varphi) = (m_1\theta + m_2\theta^3 + m_3\theta^5)(j_1 cos\varphi + j_2 sin\varphi + j_3 cos2\varphi + j_4 sin2\varphi) \tag{3.2}$$

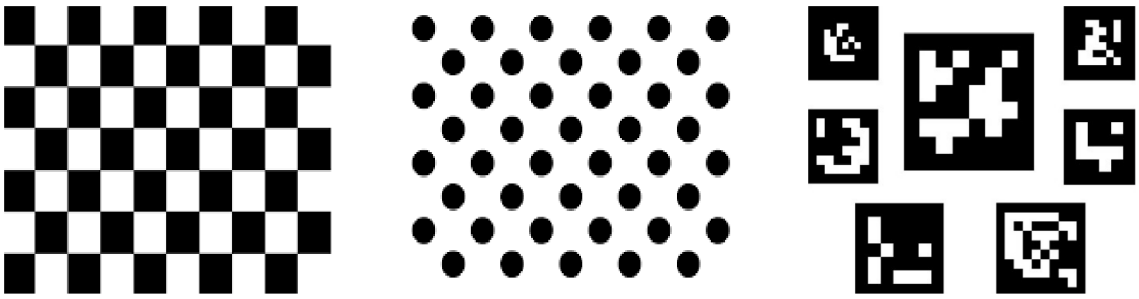
Dále podle zjištěných parametrů zkreslení zjistit pozice zkreslených bodů a vypočítat vnější a vnitřní parametry kamery (viz. sekce 3.1). Vnitřní parametry kamery se vztahují k vlastnostem, které jsou specifické pro kameru, jako je její ohnisková vzdálenost a optický střed. Tyto parametry jsou obvykle reprezentovány ve formě matice nazývané matice kamery. Vnější parametry představují umístění kamery v 3D scéně a skládají se z rotace a translace.

3.1 Knihovna OpenCV pro provedení kalibrace

Po analýze existujících možností zpracování obrázků, jako lepší řešení daného problému zvažuji použití knihovny OpenCV, jež poskytuje nástroje pro vyřešení problému zkreslení obrázků vytvořených čočkou typu rybího oka a vypočítání vyžadované matice kamery. Důvodem je hlavně to, že je to nástroj s otevřeným zdrojovým kódem, který nabízí různé metody opravy zkreslení obrázku. Jelikož to není koncový software, neomezuje uživatele v tom, co může dostat na výstupu po zpracování obrázku.

Sekce vytvořena pomocí zdrojů [13], [12] a dokumentace OpenCV.

K určení vnitřních a vnějších parametrů kamery se provádí proces zvaný kalibrace, jenž zahrnuje pořizování snímků kalibračního objektu a zadání jeho vlastností. OpenCV standardně používá metodu kalibrace pomocí kalibračního vzoru v podobě šachovnice kruhového vzoru, nebo volitelně pomocí ArUco značek složených ze širokého černého okraje a vnitřní binární matice, která určuje jeho identifikátor (obrázek 3.3).



Sachovnice | Kruhový vzor | ArUco značka

Obrázek 3.3: Typy kalibračních vzorů

Model kamery určuje algoritmus pro výpočet matice kamery a koeficientů zkreslení, které se mají použít pro odstranění zakřivení. Postup metody je v detekci souřadnic rohů jednotlivých čtverců šachovnic, nebo v detekci rohů značek, při jejich použití, porovnáním se zadanými rozměry a následným výpočtem matice kamery, jejích vnějších a vnitřních parametrů a koeficientů zkreslení, podle kterých jde odstraňovat zakřivení z obrázků. Vnější parametry představují umístění kamery ve 3D scéně, skládají se z rotace a translace. Počátek souřadnicového systému kamery je v jejím optickém středu a její osa x a y definují rovinu obrazu. Vnitřní parametry představují optický střed a ohniskovou vzdálenost kamery.

Matice kamery nezohledňuje zkreslení objektivu. Pro přesnou reprezentaci skutečné kamery zahrnuje model kamery radiální a tangenciální zkreslení čočky. K radiálnímu zkreslení dochází, když se světelné paprsky ohýbají více v blízkosti okrajů čočky než v jejím optickém středu. Tangenciální zkreslení nastane, když čočka a rovina obrazu nejsou rovnoběžné. Pro výpočet zkreslení OpenCV bere v úvahu radiální a tangenciální faktory. Pro radiální faktor se používá následující vzorec 3.3:

$$\begin{aligned} Xdistorted &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ Ydistorted &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{aligned} \tag{3.3}$$

k1, k2 a k3 — Koeficienty radiálního zkreslení čočky.
r2 = x2 + y2

Pro nezkreslený bod pixelu na souřadnicích (x, y) bude jeho poloha na zdeformovaném obrázku (Xdistorted, Ydistorted). Přítomnost radiálního zkreslení se projevuje ve formě efektu „rybího oka“.

Tangenciální zkreslení reprezentované pomocí vzorců 3.4:

$$\begin{aligned} Xdistorted &= x + [2p_1 xy + p_2(r^2 + 2x^2)] \\ Ydistorted &= y + [p_1(r^2 + 2y^2) + 2p_2 xy] \end{aligned} \tag{3.4}$$

p1 a p2 — Koeficienty tangenciálního zkreslení čočky.

Pro každý pořízený snímek OpenCV najde kalibrační objekt a výsledná data se použijí k vytvoření rovnic, jejichž řešením lze určit matici kamery. Počet požadovaných snímků závisí na typu použitého kalibračního objektu, přičemž šachovnicový vzor obvykle vyžaduje více snímků kvůli vyšší úrovni šumu přítomného v pořízených snímcích.

Jakmile je určena matice kamery, lze také vypočítat koeficienty zkreslení, které představují nedokonalosti objektivu fotoaparátu a jež je nutné vzít v úvahu pro přesné výsledky. Koeficienty zkreslení zůstávají stejné bez ohledu na použité rozlišení fotoaparátu, ale měly by být odpovídajícím způsobem upraveny na základě aktuálního rozlišení. Stručně řečeno, kalibrace kamery je zásadním krokem v počítačovém vidění, jenž zahrnuje zachycení snímků kalibračních objektů a použití výsledných dat k určení vnitřních parametrů kamery, jako je její ohnisková vzdálenost a optický střed, reprezentovaný maticí kamery. Tento proces umožňuje přesnější výsledky v různých aplikacích počítačového vidění. Pomocí knihovny OpenCV je možné provést kalibraci obrázků a dostat na výstupu opravené obrázky a všechna potřebná data ohledně míry zkreslení obrázků a hodnoty posunu pixelů, které je nutné provést pro odstranění zakřivení. Příklad výstupu aplikace v příloze C.

Kapitola 4

Problematika hybridních aplikací

Jedním z prvních rozhodnutí, které je potřeba udělat při vývoji softwaru, je typ budoucí aplikace. Mezi základní možnosti patří webová nebo nativní aplikace, ale existuje i další možnost jako využití hybridního typu aplikací. V této kapitole jsou popsané různé typy aplikací, jejich využití s větším důrazem na hybridní aplikace a nástroje, které využívají tento přístup.

4.1 Webové a nativní aplikace

Webová aplikace je kolekcí statických a dynamických webových stránek, které nevyžadují instalaci na zařízení. Místo toho jsou přístupné prostřednictvím prohlížečů a díky tomu podporují zařízení různých platforem adaptovaná pod různá rozlišení obrazovky. Webové aplikace jsou dost levné a také méně náročné na stavbu a údržbu. Jsou vytvářeny prostřednictvím stejných technologií jako web stránky – HTML, CSS, JavaScript, PHP apod., avšak doplněné o pokročilejší funkcionalitu.

Na rozdíl od webových aplikací jsou nativní aplikace vyvíjeny speciálně pro jednu platformu. Proto jsou rychlé a poskytují vynikající výkon. Je potřeba je stáhnout před použitím a nejsou přístupné prostřednictvím prohlížečů. Znovupoužitelnost kódu je proto v případě nativních aplikací minimální. Hlavní výhodou tohoto typu aplikace bude výkon a možnost lepšího navržení uživatelského rozhraní díky tomu, že je aplikace vyvinuta jenom pro konkrétní platformu. Ovšem jestli je záměrem vyvinout aplikace pro různé platformy, nastává problém, že pro každou platformu se to musí dělat zvlášť, což zvyšuje náročnost vývoje a cenu projektu. Tyto aplikace také vyžadují vysokou údržbu.

4.2 Hybridní aplikace

Existuje další možnost – zvolit pro vývoj hybridní typ aplikace. K vývoji hybridních aplikací jsou přijaty techniky pro webové i nativní aplikace. Takové aplikace lze stáhnout do zařízení a lze k nim také přistupovat pomocí prohlížečů. Nejsou však tak výkonné jako nativní aplikace, ale fungují lépe než webové a stejně mají možnost podpořit adaptace pod různá rozlišení obrazovky. Hybridní aplikace jsou velmi přitažlivé, protože vývojáři mohou vytvořit jednotnou kódovou základnu, kterou lze použít pro aplikaci na všech platformách. S výjimkou některých aspektů aplikace (hlavní je uživatelské rozhraní, u kterého uživatelé očekávají, že bude přizpůsobeno jejich zvolené platformě) stačí vývojářům napsat kód pouze jednou a poté jej lze znovu použít při vývoji aplikace pro jakoukoli jinou cílovou platformu.

Vývoj hybridních aplikací není tak drahý jako nativních. Údržba je mnohem jednodušší, protože vyžaduje pouze jednu kódovou základnu k vytvoření více verzí stejné aplikace.

Metodologie vývoje hybridních aplikací spočívá v tom, že vytváříme aplikace pomocí frameworku, který může cílit na všechny potřebné platformy: Android, IOS, Windows atd. Framework je sada nástrojů, knihoven, konvencí a pravidel pro vývoj softwaru, která umožňuje programátorům rychleji a efektivněji vytvářet aplikace. Framework poskytuje základní strukturu a architekturu pro aplikace, čímž usnadňuje programátorům vytvářet a spravovat různé funkce aplikace. Výhodou použití frameworku v tom, že není potřeba psát všechny funkce a algoritmy od začátku. To znamená, že vývoj aplikace může být rychlejší a efektivnější. Tvorba hybridních aplikací je snazší než nativních, použití frameworku šetří čas při vývoji ve srovnání s vývojáři pracujícími na projektech navržených speciálně pro každou platformu. Frameworky se pravidelně aktualizují o nové funkce, což umožňuje vývojářům využívat specifické funkce určitých platforem a zároveň zachovat známé vývojové prostředí. To má ale vedlejší účinek, kdy se zvětšuje rozměr frameworku a výsledné aplikace, protože musí zohledňovat všechna různá zařízení a verze OS, na které se aplikace zaměřuje. Pokud ale je potřeba vyvíjet pro několik platforem najednou, je efektivnější použít přístup hybridních aplikací než vyvíjet nativní aplikace pro každou platformu zvlášť. Mezi výhody použití modelu hybridní aplikace patří:

- Největší výhoda hybridní aplikace spočívá v tom, že je kód napsán pouze jednou a následně konvertován pro všechny platformy. K vývoji tedy stačí pouze jeden vývojář (tým) a framework umožňující konverzi.[8]
- Díky tomu, že aplikace vypadá a funguje na všech zařízeních stejně, je usnadněn přechod mezi tabletem, počítačem nebo smartphonem. Uživatel si tak nemusí zvykat na odchylky v prostředí. Zároveň je tím odstraněna problematika volby, pro který operační systém aplikace vyvíjet.[8]
- Možnost použití různých technologií k vytvoření aplikace, což poskytuje větší flexibilitu.
- Není potřeba znát jazyk vyžadovaný každou platformou.
- Použití stejných komponent a prvků návrhu na různých platformách.

4.3 Frameworky pro realizace hybridních aplikací

Nejlepší hybridní aplikační frameworky, které lze použít pro snadnější vývoj aplikací hybridního typu:

- React Native. Když je React knihovna na vytváření uživatelského rozhraní typicky pro webové aplikace, React Native je další samostatná javascript knižnice, která obsahuje soubor speciálních React komponent. Tyto komponenty je snadno možné přeložit do nativních komponent a využívají se pro Android nebo IOS. Podobně jako React pro Web jsou aplikace React Native psány pomocíJavaScriptu a speciálního značkovacího jazyka navrženého pro tyto frameworky – JSX a používá podobnou architekturu jako React s komponentami a virtuálním DOM, ale místo vykreslování do prohlížeče používá nativní komponenty uživatelského rozhraní pro každou platformu. Proto pro IOS React Native vyvolá nativní rozhraní API pro vykreslování v Objective-C a pro Android využije Javu. Výsledná aplikace se tedy bude vykreslovat pomocí skutečných

komponent mobilního uživatelského rozhraní, nikoli webových zobrazení, a bude vypadat a působit jako jakákoli jiná mobilní aplikace. Nevýhodou je to, že framework zaměřený především na vývoj mobilních aplikací. Vytvořeno pomocí [2].

- Flutter – je open-source framework vyvinutý společností Google, který používá vlastní programovací jazyk zvaný Dart. Může dodávat aplikace na mobilní, webové, nebo desktopové platformy, a dokonce i na vestavěná zařízení. Při psaní a ladění aplikace Flutter běží ve virtuálním stroji Dart, který obsahuje spouštěcí modul just-in-time. To umožňuje rychlou kompilaci a také „hot reload“, s jehož pomocí lze do běžící aplikace vložit úpravy zdrojových souborů. Flutter to dále rozšiřuje o podporu pro znovunačtení se zachováním stavu, kdy se ve většině případů změny zdrojového kódu projeví okamžitě v běžící aplikaci, aniž by bylo nutné ji restartovat a bez ztráty stavu. Pro lepší výkon také používají na všech platformách předběžnou kompilaci. Popis je vytvořen pomocí [2] a [6].

- Ionic – poskytuje sadu nástrojů pro vytváření nativních aplikací pro IOS a Android a progresivních webových aplikací připravených pro mobilní zařízení. Používá Ionic Capacitor – multiplatformní nativní most, který umožňuje přeměnit jakýkoli webový projekt na nativní mobilní aplikaci pro IOS nebo Android. Capacitor umožňuje snadný přístup k běžným funkcím zařízení pomocí základního JavaScriptu s úplným přístupem k nativnímu operačnímu systému, když jej potřebujete. Ionic Framework rozšiřuje Capacitor tím, že poskytuje bohatou knihovnu komponent uživatelského rozhraní optimalizovaných pro mobily – plus mobilní směrování, navigace, gesta a animace. Ionic Framework pracuje se specifickými frameworky JavaScript, včetně React, Angular a Vue. Vytvořeno pomocí [7].

- Electron je open-source framework vyvinutý společností GitHub, který umožňuje vývojářům vytvářet desktopové aplikace pomocí webových technologií, jako jsou HTML, CSS a JavaScript. Electron používá Chromium a Node.js k vytvoření runtime prostředí pro desktopové aplikace. To umožňuje vývojářům vytvářet desktopové aplikace s webovými technologiemi a také poskytuje přístup k rozhraním API na úrovni systému pro přístup k systému souborů, sítím a dalším funkcím. Jednou z hlavních výhod Electronu je, že umožňuje vývojářům vytvářet desktopové aplikace s webovými technologiemi, které mohou být známé a snadno použitelné pro mnoho vývojářů. Umožňuje také použití stávajících webových nástrojů a knihoven k vytváření desktopových aplikací. Popis vytvořen pomocí [4].

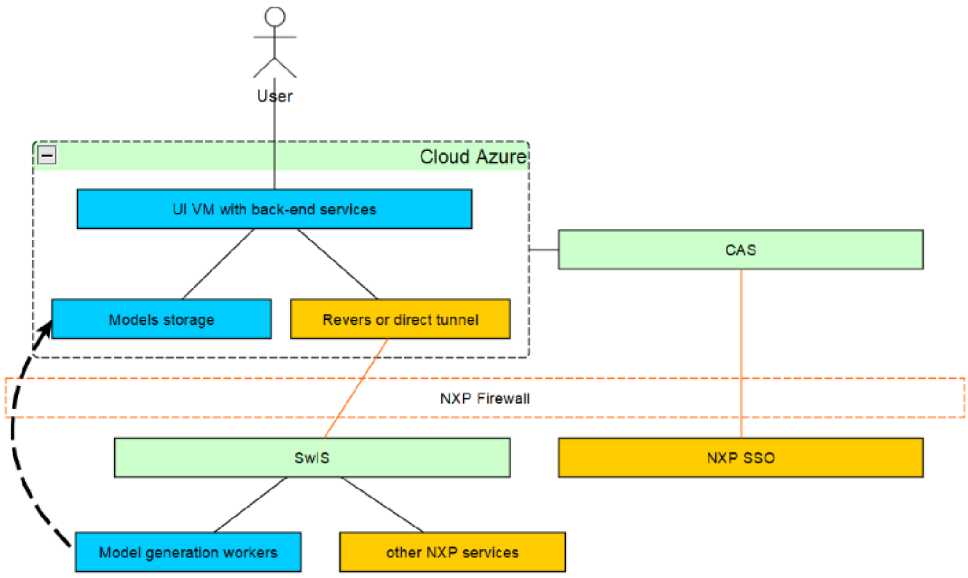
4.4 Hybridní aplikace vyvinuté firmou NXP

Diplomová práce je vytvořena v spolupráci s firmou NXP, zabývající návrhem a vytvořením čipů a softwaru spojeného s tím. Firma NXP má vytvořeno několik aplikací zvoleného typu a podobného záměru – hybridní aplikaci Audio Tuning Tool a Voice Recognition Tool. Analýza použitých přístupů a jejich rozšíření při vývojů tohoto projektu pomůže vyhnout se problémům a zmenšit časovou a cenovou náročnost aplikace.

4.4.1 Voice Recognition Tool

Voice Recognition Tool (VIT) je online nástroj zaměřený na vytvoření modelů pro rozpoznání hlasových příkazů s použitím jednoho ze zařízení NXP. Využívá knihovnu hlasové

inteligentní technologie a poskytuje podporu hlasového uživatelského rozhraní s neustále zapnutou detekcí slov probuzení a místními příkazy. Je založen na nejmodernějších technologiích hlubokého učení a rozpoznávání řeči a poskytuje kompletní řešení zvukového rozhraní / slov na probuzení / hlasových příkazů. Nasazení VIT je provedeno v podobě hybridní aplikace, kde je uživatelské rozhraní (dále UI) a úložiště modelů zprovozněno na veřejném cloudu (Azure). Architektura aplikace je zobrazena na obrázku 4.1. Veřejný cloud umožňuje uživatelům přistupovat k různým výpočetním zdrojům, jako jsou servery, úložiště dat a aplikace, a to bez nutnosti vlastnit a spravovat vlastní hardwarovou infrastrukturu. Tyto zdroje jsou sdíleny mezi mnoha uživateli a jsou obvykle fakturovány na základě užívání. Poskytovatelé veřejného cloudu, jako jsou Amazon Web Services, Microsoft Azure a Google Cloud Platform, nabízejí různé služby, jako jsou úložiště dat, virtuální servery, nástroje pro vývoj aplikací a mnoho dalších.



Obrázek 4.1: Architektura aplikace Voise Recognition Tool

Těžká práce po zpracování hlasových modelů probíhá v prostředí NXP SwIS, což je interní cloudový servis NXP podobný Azure. SwIS a Cloud jsou propojeny přes reverzní tunely bez přímého vystavení prostředí SwIS.

Reverzní tunel bez přímého vystavení je technika využívaná v síťových komunikacích k zabezpečení spojení mezi dvěma uzly (např. mezi klientem a serverem), kdy spojení je iniciováno zvnitřku uzlu, který se nachází za firewallem nebo NAT (Network Address Translation). Při reverzním tunelování bez přímého vystavení se klient připojuje na vzdálený server, který umožňuje vzdálený přístup. Vzdálený server vytvoří "tunel"a klient s jeho pomocí komunikuje s vnitřním uzlem, který se nachází za firewallem nebo NAT. Tímto způsobem se podaří obejít omezení, která jsou v místě, kde se nachází klient, a umožní se vzdálené připojení k internímu síťovému zařízení. Reverzní tunelování bez přímého vystavení se často používá v případech, kdy se chce zabezpečit spojení a zároveň umožnit přístup do sítě zvenčí, aniž by byla nutná úprava konfigurace firewalu nebo NAT. Popis vytvořen ze zdrojů [11]

Modul odpovídající za generace modelů má několik agentů, které provádějí veškeré operace. Výpočetní agenti navržené tak, aby vykonávali určité úlohy v prostředí počítačové sítě. Výpočetní agenti se používají v distribuovaných výpočtech, kde více počítačových zdrojů (uzlů) spolupracuje na řešení jedné úlohy, v tomto případě když vice uživatelů chtějí vygenerovat hlasový model. Když uživatel spouští generování modelu, je kontaktován jeden z dostupných agentů. Agenti mohou v případě potřeby potenciálně běžet také na straně uživatele. Agenty umožňuji snazší způsob podpořit hybriditu aplikace a pomoct zvýšit bezpečnost aplikace, protože mají pouze omezený počet oprávnění a nelze je použít k penetračním útokům. Využiti agentů bylo by jedním z řešení vhodných i pro aplikace na kalibraci čoček, kde veškeré operace budou prováděné přes agenty.

4.4.2 Audio Tuning Tool

Audio Tuning tool - nástroj který umožňuje graficky ovládat zpracovaní a laděni zvukové nahrávky pomoci externí knihovny pro prací s audio daty - Essential Audio Processing. Knihovna poskytuje základní zvukový algoritmus pro přehrávaní zvuku a dodává tónový generátor, mixér, ekvalizér, vylepšení basů, vylepšení výšek, maximalizátor hlasitosti, ovládání hlasitosti, kontrolu rovnováhy, automatickou regulace hladiny, analýzu výkonového spektra, 3D efekt (zvuk koncertu). Konfigurace zvukové simulace je možné nastavit přes grafický program Audio Tuning Tool, který je dostupný v podobě desktopové aplikaci. Toto řešeni umožňuje komunikaci taky s "hardwarem"(s NXP čipem) – nahrát na něho konfigurace pro zpracovaní zvuku, nebo stáhnout nastaveni čipu pro předběžné nastaveni konfigurace a provedeni simulace zvuku v aplikaci. Projekty jsou implementované pomocí interního frameworku firmy NXP. Tento framework byl vyvíjen jako univerzální software pro vývoj multiplatformních aplikací, je zaměřený na potřeby interně vyvíjených aplikací a používá se pro jejich implementaci. Hlavní jeho zaměření je:

- poskytovat nástroje pro rychlý vývoj multiplatformních aplikací (Windows, Linux, Mac OS X, Android, iOS, Windows Phone, Blackberry, webové servery)
- využití webových technologií s offline podporou, konektivitou operačního systému a vestavěnou podporou CORS - mechanismus, který umožňuje serveru nastavit jakoukoli doménu, schéma nebo port, jiný než jeho vlastní, ze kterého by měl prohlížeč povolit načítání zdrojů.
- Kompatibilita mezi prohlížeči (IE7+, Edge, Firefox, Chrome, Opera, Safari)
- flexibilita a škálovatelnost
- plně přizpůsobitelné uživatelské ovládací prvky
- externí konfigurace a snadná podpora lokalizace
- vývoj zdrojového kódu nezávislý na běhu — vývojář se nemusí starat o cílovou platformu

Framework nabízí sadu vestavěných komponent pro UI prvky jako panely, tlačítka, vstupní pole a funkci nad nimi pro validaci vstupu uživatele nebo zpracování událostí, které pomáhají urychlit a ulehčit implementace frontendové součásti projektu. Framework má doporučené techniky a postupy zpracování určitých typů operace jako způsob komunikace frontendu i backendu, uspořádání dat z databáze, komunikace s ní a další operace.

Vidím jako výhodu použít tento framework pro implementaci aplikace pro kalibraci čoček, jelikož má podobný záměr projektu z pohledu architektury a typu aplikace. Výhodou použití interního frameworku bude rychlost implementace, možnost snížit rizika díky zkušenostem z minulých projektů stavěných na tomto frameworku a předem připravené komponenty.

Kapitola 5

Návrh kalibračního nástroje a uživatelského rozhraní

5.1 Organizace a plánování pomocí přístupu DevOps

Během vývoje aplikací byl dodržován přístupu známého jako DevOps. Je to kombinace postupů a nástrojů, které zvyšují schopnost organizace dodávat aplikace a služby vysokou rychlostí. DevOps umožňuje efektivnější spolupráci mezi vývojáři a správci infrastruktury, kteří společně pracují na vývoji, testování, nasazení a správě aplikací i služeb v průběhu celého jejich životního cyklu. Klíčovými prvky DevOps jsou automatizace, kontinuální integrace a nasazení, standardizace procesů a nástrojů, spolupráce a komunikace mezi vývojáři a správci infrastruktury. DevOps je způsob myšlení a pracovní postup, který zahrnuje celý životní cyklus aplikací. Jednotlivé fáze nejsou určené pouze pro konkrétní role a vyžadují spolupráci a zapojení všech členů týmu, kteří jsou schopni přinést hodnotu do celého procesu. V kultuře DevOps je klíčové, aby vývojáři a správci infrastruktury spolupracovali a komunikovali průběžně po celou dobu životního cyklu aplikace. To umožňuje včasnou detekci a řešení potenciálních problémů a zajišťuje, že celý tým je informován o změnách a vývoji aplikace.

Fáze plánování v rámci DevOps představuje klíčovou část procesu vývoje softwaru. V této fázi se zaměřuje na vymýšlení, definování a popisování funkcí a možností aplikací a systémů, které budou vytvářeny. V rámci plánování se zaměřuje na řízení a sledování průběhu úloh na různých úrovních od jednotlivých úloh pro konkrétní produkty až po úlohy související s celým portfoliem produktů. Pro plánování a řízení průběhu se využívají různé nástroje a metody, například vytváření backlogů, sledování chyb, správa agilního vývoje softwaru pomocí Scrumu, využití panelů Kanban a vizualizace průběhu pomocí řídicích panelů.

Během vývoje aplikace byla použita metodologie Scrum. Je to agilní metodologie vývoje softwaru, která umožňuje efektivně a rychle vytvářet softwarové produkty. Tato metodologie se zaměřuje na iterativní a inkrementální přístup k vývoji softwaru, což znamená, že produkt bude vytvářen postupně v krátkých cyklech nazývaných sprinty. Během sprintu je vytvořeno určité množství funkcí nebo vlastností produktu, které jsou definovány v backlogu produktu. Backlog produktu obsahuje seznam funkcí nebo vlastností, které musí být vytvořeny, a je neustále aktualizován a přizpůsobován během celého vývojového cyklu. [3]

V každém sprintu je vytvořen hotový kus softwaru, který je následně testován. Pokud jsou potřeba nějaké změny, jsou začleněny do backlogu produktu a implementovány v dalším

sprintu. Na tomto projektu trval jeden sprint 1 nebo 2 týdny a na konci každého sprintu se prováděla kontrola dosažených výsledků, splněných a nesplněných úkolů a plánování úkolů na další sprint. Scrum je velmi flexibilní metodologie, která umožňuje přizpůsobit proces vývoje potřebám a rychle reagovat na změny v požadavcích nebo problémech během vývoje.

Vývojová fáze DevOps zahrnuje všechny aspekty kódování (psaní, testování, kontrolu i integraci kódu). Pro dosažení těchto cílů se používají nástroje pro automatizaci procesů a iteruje se malými kroky. Důraz je kladený na automatizaci testování a kontinuální integraci, což umožňuje rychlé testování a integraci kódu do celkového systému. Při integraci využívají nástroje pro správu konfigurace k řízení změn v systému a minimalizaci rizika chyb při úpravách konfigurace. Správa konfigurace zahrnuje sledování stavu různých prostředků v systému, včetně serverů, virtuálních strojů a databází. Pro lepší kontrolu změn se využívá správa verzí, což je postup správy kódu ve verzích, to znamená sledování historie změn a úprav pro zajištění snadné kontroly kódu a jeho obnovení. Díky tomuto přístupu lze udržovat systém vždy v konzistentním stavu a snadněji odhalit chyby. [3]

DevOps kultura zahrnuje spolupráci mezi týmy založenou na transparentnosti a důvěře. Týmy, jako jsou vývoj a provoz IT, jsou vzájemně propojené a pravidelně sdílejí své problémy, priority a procesy. Tyto týmy společně plánují práci, definují cíle a měřítka úspěchu a společně pracují na dosažení těchto cílů. Důraz kladou na otevřenou a průhlednou komunikaci, což umožňuje efektivní spolupráci a koordinaci mezi týmy. [3]

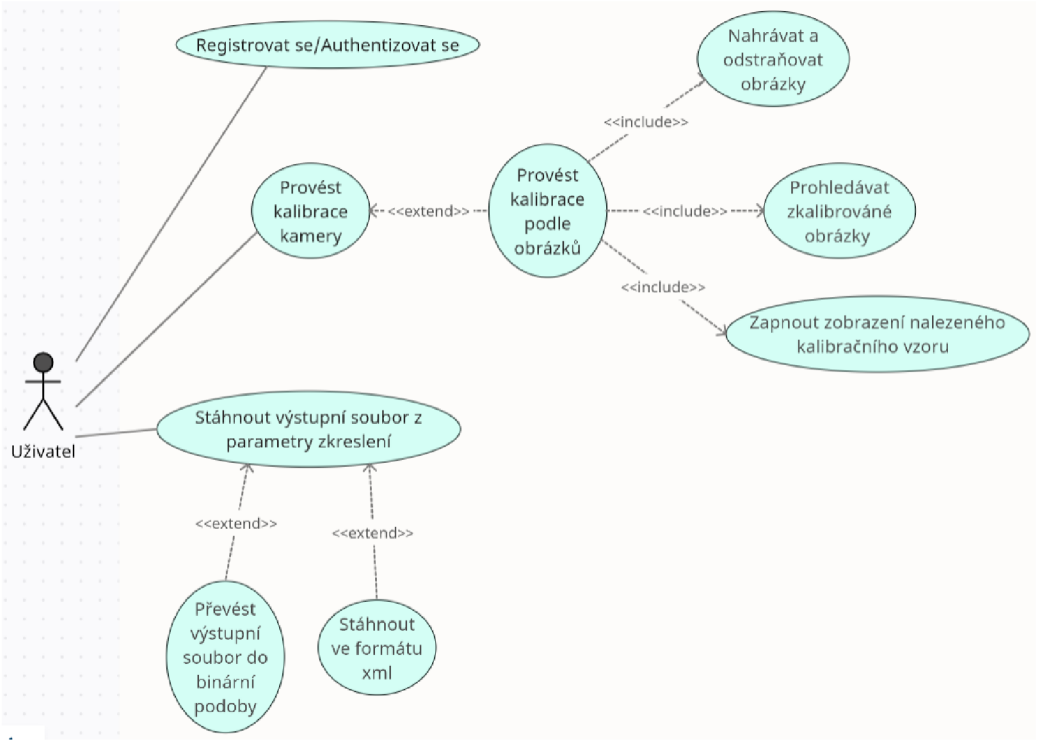
Při této práci byly využity kratší cykly přidání a testování nové funkcionality, což usnadnilo plánování a správu rizik. To také omezuje případné dopady na stabilitu systému. Zkrácení cyklu vydávání také umožnilo přizpůsobovat se měnícím se potřebám a problémům spojeným s vývojem a použitými technologiemi.

Hlavní výhodou mojí práce v týmu, který využívá DevOps, vidím to, že tým se orientuje na růst. Všichni členové týmu se rychle adaptují prostřednictvím zpětné vazby a zjištěné poznatky začleňují do svých procesů, neustále se zlepšují, zvyšují kvalitu vyvíjených produktů a zrychlují inovace a přizpůsobení aktuálnímu trhu.

5.2 Návrh uživatelského rozhraní

První fáze návrhu uživatelského prostředí je analýza uživatelských požadavků na aplikaci a definici scénáře užití aplikace (obrazek 5.1). K dosažení tohoto cíle pomůže návrh diagramu případů užití. Diagram případů užití je grafický nástroj, který se používá k modelování chování systému z pohledu uživatele. Tento diagram popisuje funkce a interakce mezi uživateli a systémem v různých scénářích.

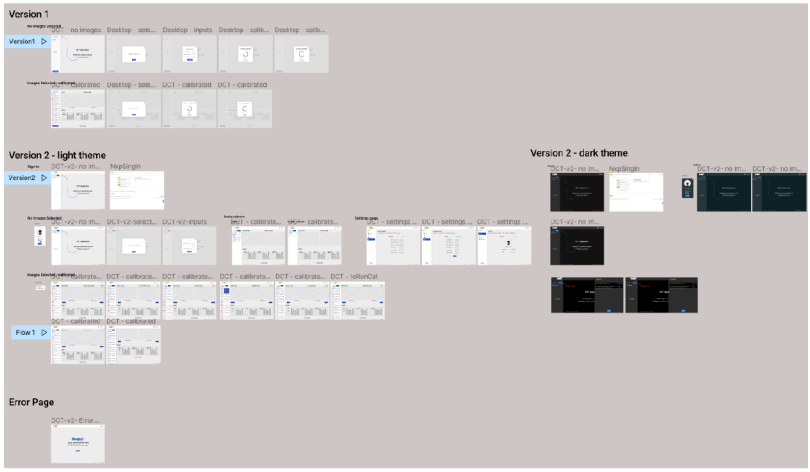
Hlavním cílem použití aplikace je vypočítat nastavení kamery pro použití v dalších aplikacích, a to způsobem nahrávání obrázků z kamery, kterou uživatel chce zkalibrovat, zadání vstupních parametrů a na výstupu dostat vhodným způsobem parametry zkreslení. Protože kalibraci je potřeba provést pouze jednou na kameru, má smysl výsledky po úspěšné kalibraci uložit. OpenCV umožňuje exportovat zjištěné parametry zkreslení do xml souboru, který by bylo možné uchovávat do databáze. Tím pádem uživatel bude mít možnost stáhnout soubor z nastaveními provedené kalibrace. Vzhledem k tomu, že uživatel může kalibrovat různé kamery na stejném kalibračním vzoru, bylo by vhodné přidat nastavení pro kalibraci a nastavení pro celou aplikaci a také je uchovávat v databázi. Pro tento účel je nutné zavést registrace i autentizace uživatele. Nahrané obrázky slouží jenom pro kalibraci a z důvodu velkého obsahu a soukromí uživatele je vhodné neuchovávat obrázky a využít je jenom pro výpočet nastavení a představení výsledků. Pro návrh uživatelského pohledu



Obrázek 5.1: Diagram případů úžiti

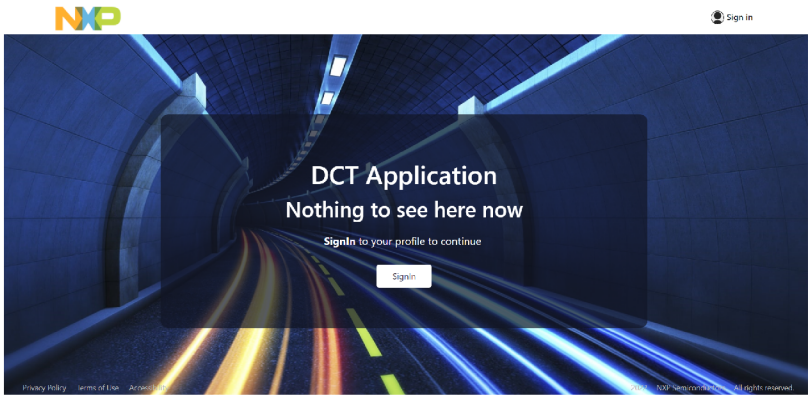
v krátkém čase s minimálními riziky a cenou byl zvolen nástroj Figma (Návrh designu na obrázku 5.2).

Figma je profesionální nástroj pro design, který umožňuje vytvářet, sdílet a spolupracovat na návrzích uživatelského rozhraní (UI) a uživatelského zážitku (UX) prostřednictvím webového rozhraní. Figma je často používána pro návrh webových stránek, mobilních aplikací, grafických prvků a dalších digitálních produktů. Figma je webová aplikace, takže je přístupná z jakéhokoli zařízení s internetovým připojením, což umožňuje pracovat odkudkoli. Umožňuje pracovat na stejném projektu současně a vidět změny ostatních v reálném čase. To znamená, že lze rychleji reagovat na problémy a přizpůsobit se změnám v průběhu projektu. Umožňuje vytvářet interaktivní prototypy, které umožňují testovat funkčnost a použitelnost produktu v různých fázích vývoje.



Obrázek 5.2: Návrh uživatelského rozhraní v nástroji Figma

Podle navrženého diagramu případů užití vznikl design stránek pro popsané scénáře: Úvodní stránka, kterou uvidí uživatel po otevření aplikace v prohlížeči. Veškerá funkcionalita programu vyžaduje autentizace uživatele, proto úvodní stránka nabízí jenom možnost přihlášení (obrázek 5.3). Podrobněji o autentizaci v sekci 6.3

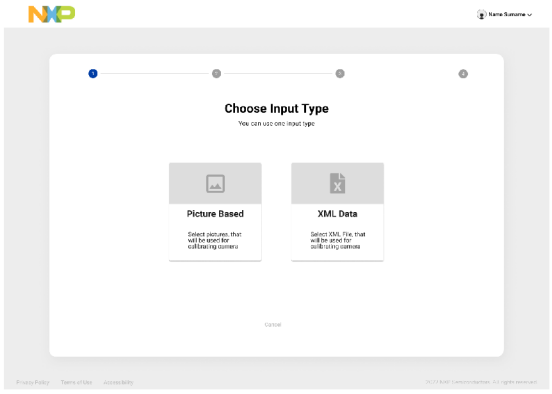


Obrázek 5.3: Úvodní stránka aplikace

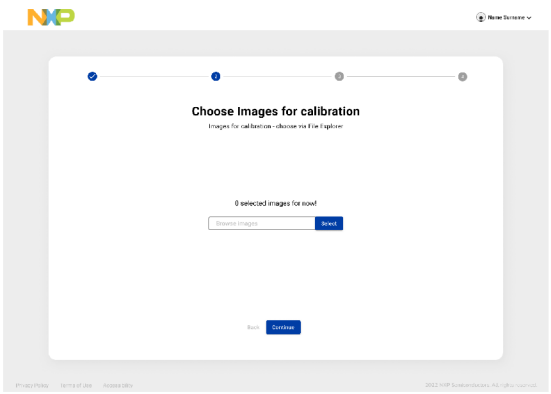
Přihlášený uživatel dostane přístup ke stránce, ze které může nakonfigurovat a odstartovat novou kalibraci kamery (obrázek 5.4).

Obrázek 5.4: Hlavní stránka po přihlášení

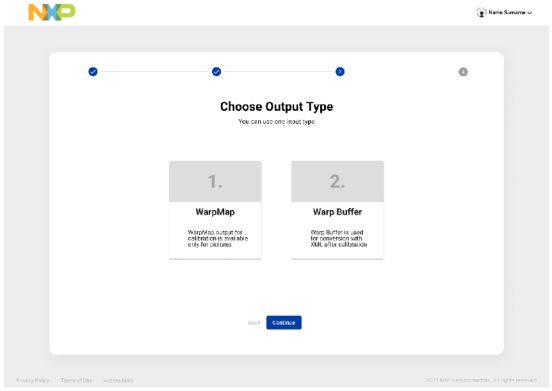
Pro konfiguraci kalibrace se uživateli otevře dialog, ve kterém je možné krok za krokem možné zvolit nastavení kalibrace. Uživatel má možnost výběru typu vstupu kalibrace (obrázek 5.5), jestli to chce kalibrovat podle obrázků obsahujících kalibrační vzor, nebo podle xml souboru s parametry kamery. Při výběru kalibrace podle obrázků bude vyzván k výběru a načtení obrázků z lokálního souborového systému počítače (obrázek 5.6). Pak jde zvolit typ výstupu: xml soubor obsahující parametry zkreslení obrázků nebo binární soubor (obrázek 5.7). Poslední sekce je pro zadání parametru obrázků, jako počet úhlů šachovnice vertikálně a horizontálně, rozměr jednotlivé buňky šachovnice nebo její celkový rozměr (obrázek 5.8).



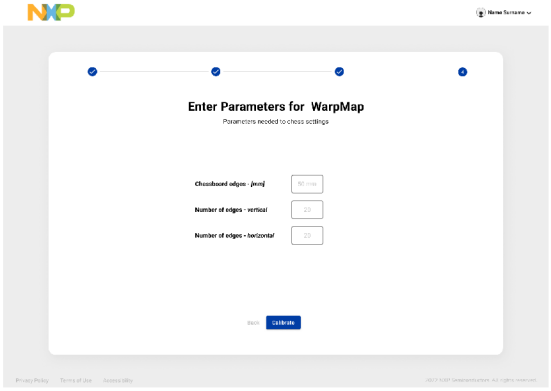
Obrázek 5.5: Výběr vstupního typu kalibrace



Obrázek 5.6: Nahrávání obrázků

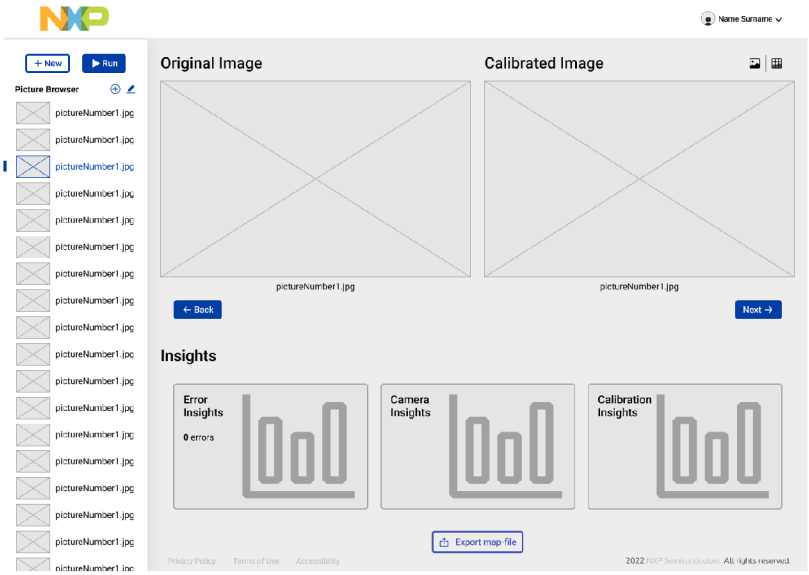


Obrázek 5.7: Výběr typu výstupního souboru



Obrázek 5.8: Nastavení parametrů kalibrace

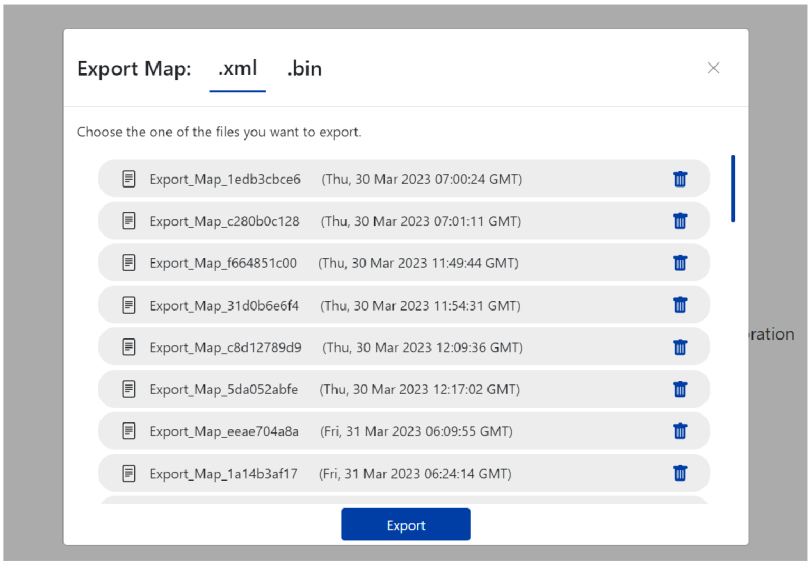
Po proběhnutí kalibrace může uživatel projít jednotlivé obrázky, kde uvidí, jak bude vypadat zkalibrovaný obrázek, může zapnout zobrazení nalezeného kalibračního vzoru (obrázek 5.9) a může stáhnout xml soubor, který obsahuje vstupní parametry kalibrace, vypočítanou matici kamery, vnější a vnitřní parametry čočky.



Obrázek 5.9: Hlavní okno po provedení kalibraci

Pomocí dialogu na exportování souborů může uživatel vybrat a stáhnout xml výstup libovolné úspěšné kalibrace, binární soubor, jestli při provedení kalibrace byl zvolen odpovídající typ výstupu, nebo smazat soubory, které nepotřebuje (obrázek 5.10).

Příklady výstupu aplikace jsou v příloze C



Obrázek 5.10: Dialogové okno pro exportování výstupu

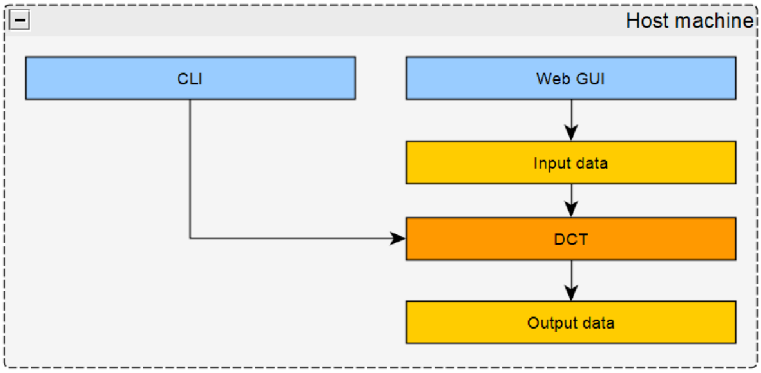
Kapitola 6

Implementace kalibračního nástroje na bázi hybridní architektury

6.1 Návrh možných architektur

Architektura aplikace navrhuje možnost ovládání běhu pomocí CLI (rozhraní příkazového řádku), což je textové uživatelské rozhraní používané ke spouštění programů, správě počítačových souborů a interakci s počítačem. Z příkazové řádky bude možné nahrát obrázky, podle kterých bude provedena kalibrace do určité složky, spustit modul odpovídající za kalibraci obrázků a který vygeneruje výstup do jiné složky. Jiný způsob ovládání bude pomocí webového uživatelského rozhraní. Při návrhu architektury vyniklo několik možných variant:

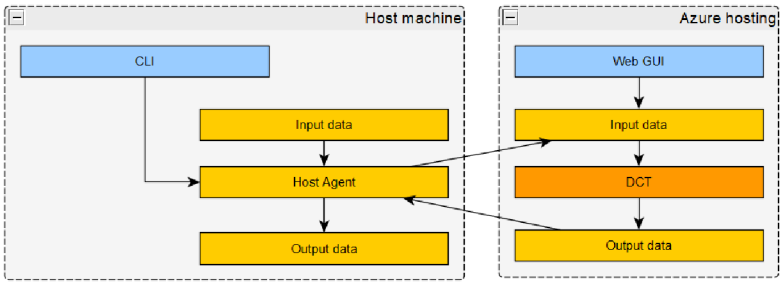
- Zprovozněno na straně uživatele (obrázek 6.1). DCT je dodáváno společně se serverem pro hostování webového GUI na stroji uživatele, CLI umožňuje uživateli ovládat DCT bez použití GUI. Výstupní data jsou následně zpracována uživatelem ručně. Aplikace bude dodávána společně se serverem pro hostování webového GUI na stroji uživatele. CLI bude umožňovat uživateli ovládat běh aplikace bez použití GUI. Výstupní data budou následně zpracována uživatelem ručně.



Obrázek 6.1: Aplikace zprovozněná na straně uživatele

Výhodami architektury jsou rychlejší zpracování, protože není vyžadována synchronizace dat, přímý přístup k místnímu prostředí stroje a možná snadnější integrace do jiných nástrojů. Nevýhodami přístupu jsou nepředvídatelné problémy s nedostatkem zdrojů (RAM, CPU, místo na disku), možné problémy související s uživatelskými oprávněními, omezeními brány firewallu, dostupností portů nebo jinými problémy se zabezpečením nebo sítí. Při této architektuře je těžké vynutit aktualizaci na nejnovější verzi nebo spustit více verzí na stejném počítači. Těžko laditelné problémy a shromažďování požadovaných dat pro replikaci problému.

- Zprovozněno na cloudu (obrázek 6.2). Aplikace spolu s webovým GUI bude hostována na cloudovém serveru, což je vzdálený počítačový server, který dává k dispozici výpočetní a úložné kapacity poskytované přes internet jako Azure nebo Google Cloud Platform. Bude poskytnuta možnost připojit počítač uživatele ke cloudovému serveru pomocí agentů a ovládat aplikace přes CLI. Hostitel bude zodpovědný za synchronizaci vstupních a výstupních dat mezi hostitelem a cloudem.

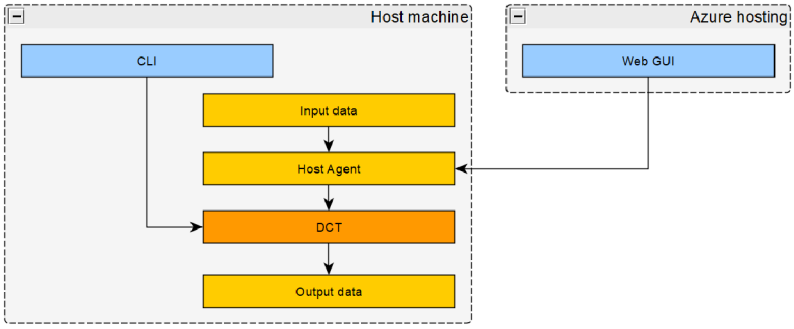


Obrázek 6.2: Aplikace zprovozněná na cloudu

Výhodou je lepší stabilita díky běhu ve vyhrazeném prostředí (žádné problémy s firewallem nebo výkonem), stále aktuální verze, data lze nahrávat/stahovat přímo přes prohlížeč a následně ručně zpracovávat, schopnost používat uživatelské profily a funkce ukládání do mezipaměti. Nevýhody – potřeba platit za hosting, kde může virtuální počítač stát příliš mnoho kvůli požadavkům na zdroje instance aplikace, pomalejší pracovní postup kvůli potřebě synchronizace dat.

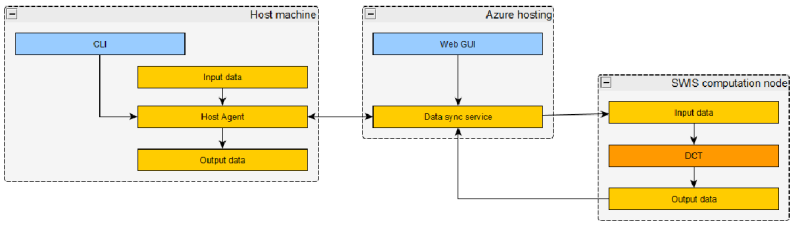
- Výpočet běží na uživatelském stroji, GUI dostupné na cloudu (obrázek 6.3). Na serveru poskytovatele cloudu je hostováno pouze webové GUI. Výpočetní modul běží na počítači uživatele a lze jej ovládat přímo přes CLI. Hostitelský agent je zodpovědný za možnost připojit aplikace ke cloudu a ovládat jej vzdáleně přes webové GUI.

  Výhody – rychlejší zpracování, protože není vyžadována synchronizace dat, vzdálený přístup k místnímu prostředí stroje a možná snadnější integrace do jiných nástrojů, lepší stabilita GUI díky běhu ve vyhrazeném prostředí (žádný firewall nebo problémy s výkonem), GUI je stále v aktuální verzi a verzi instance aplikace na stroji uživatele lze snadno ověřit, schopnost používat uživatelské profily. Nevýhody – systém nemůže fungovat hned po vybalení, protože na straně klienta musí běžet stažená instance aplikace, je třeba platit za hosting.



Obrázek 6.3: GUI běží na cloudu

- DCT běžící na výpočetním uzlu (obrázek 6.4). Webové grafické uživatelské rozhraní je hostováno na serveru poskytovatele cloudu. Hostitelský počítač může být připojen k serveru hostitelským agentem. Výpočetní modul je hostován na výpočetním uzlu běžícím ve SWIS, což je interní cloudový servis NXP podobný Azure a připojený k serveru. Uživatel je schopen ovládat aplikace přes CLI nebo webové GUI pomocí služby synchronizace dat.



Obrázek 6.4: Aplikace běží na výpočetním uzlu

Nejvíce škálovatelný systém, pro výpočetní uzel jde využít interní server firmy NXP, což je výhodné ze strany ceny provozu, pokud běží ve SWIS, má lepší stabilitu díky běhu ve vyhrazeném prostředí (žádné problémy s firewallem nebo výkonem), stále aktuální verze, jde použít systém hned, není potřeba nic instalovat, schopnost používat uživatelské profily, funkce ukládání do mezipaměti a sdílení, řešení je levnější než v případě, že vše běží v cloudu. Je nejsložitější variantou, má pomalejší pracovní postup kvůli potřebě synchronizace dat a je třeba platit za hosting, ale tato varianta je nejpřínosnější z pohledu provozování a údržby a uživatelsky nejpřívětivější, což ji vyhodnocuje jako lepší návrh architektury.

6.2 Nástroje a technologie použité při vývoji

Hybridní webová aplikace kombinuje prvky webových aplikací s prvky nativních aplikací. To znamená, že využívá webové technologie pro vytváření uživatelského rozhraní, ale zároveň může využívat nativní funkce zařízení, v tomto projektu využití zvláštního modulu, který by prováděl veškeré výpočty a byl zaměřený na rychlost zpracování dat, nebo rozšíření aplikace o možnost připojení a nastavení hardwaru kamery, což by umožňovalo mít přístup k jejím parametrům nebo nahrávat na kameru nastavení pro kalibraci.

Architektura hybridní webové aplikace se skládá z frontendu, backendu a výpočetního modulu. Frontend je část aplikace, která běží v prohlížeči a zajišťuje uživatelské rozhraní.

Využívá webové technologie, jako jsou HTML, CSS a skriptovacího jazyka pro zpracování logiky programu, kde byl zvolen TypeScript. HTML (HyperText Markup Language) je značkovací jazyk používaný pro tvorbu webových stránek a aplikací. HTML umožňuje popsat strukturu a obsah webových stránek pomocí různých tagů a atributů. Tyto tagy a atributy jsou poté interpretovány webovým prohlížečem a použity pro zobrazení obsahu na stránce. Spolu s HTML se využívá CSS (Cascading Style Sheets), což je jazyk používaný pro popis vzhledu webových stránek a aplikací napsaných v jazyce HTML. CSS umožňuje oddělit obsah webových stránek od jejich prezentace, což umožňuje snadnější úpravy a údržbu. Umožňuje definovat různé styly a vlastnosti pro jednotlivé prvky HTML, jako jsou barva, velikost, pozadí, zarovnání, vnitřní okraje, ohraničení a mnoho dalších. CSS také zvyšuje flexibilitu a rychlost webových stránek tím, že odděluje vzhled od obsahu, což umožňuje načítání stránek rychleji a poskytuje uživatelům lepší zážitek z prohlížení webových stránek a umožňuje vhodným způsobem zobrazovat elementy aplikace v závislosti na rozlišení plochy - vytvořit responzivní aplikace.

Pro snadnější práci s CSS byl použit samotný skriptovací jazyk SASS. Tento jazyk rozšiřuje CSS tím, že poskytuje několik mechanismů dostupných v tradičnějších programovacích jazycích, zejména objektově orientovaných, které však nejsou dostupné pro CSS samotné. SASS umožňuje používat proměnné, zanořovat bloky pravidel, podporuje importy, dědičnost, vestavěné funkce a další věci. Prohlížeč nemůže pracovat přímo s kódem SASS. Je potřeba provést převod souborů se styly s rozšířením .scss nebo .sass do .css souboru pomocí pre-procesoru SASS.

Uživatelské rozhraní pro webovou aplikaci je navržené v podobě SPA. Single-page aplikace (SPA) je typ webové aplikace, která funguje jako jedna stránka. To znamená, že všechny potřebné komponenty se načítají pouze jednou a dále se uživateli prezentují dynamicky bez nutnosti načítání další stránky. Mezi hlavní výhody SPA patří: rychlost, snadnější použití, lepší výkon. SPA aplikace načítají obsah rychleji, protože uživatel musí načíst pouze jednu stránku, což vede k lepšímu uživatelskému zážitku.

Rychlejší a snadnější vývoj uživatelského rozhraní podle navrženého designu umožňuje využití Bootstrap a softwaru Bootstrap studio. Bootstrap je front-endový framework pro vývoj webových stránek a aplikací, který usnadňuje tvorbu responzivního designu. Jedná se o soubor předem napsaných CSS stylů, javascriptových knihoven a HTML kódů, které lze použít k vytvoření různých komponent, jako jsou například navigační menu, formuláře, tlačítka, modální okna a další. Bootstrap usnadňuje vývoj a zjednodušuje proces návrhu, protože není potřeba vytvářet všechny prvky od základu, ale je možné použít hotové komponenty a přizpůsobit je svým potřebám. Bootstrap také usnadňuje vytvoření responzivního designu, protože poskytuje třídy pro nastavení velikostí a pozic prvků v závislosti na šířce obrazovky. To umožňuje vytvářet webové stránky, které se automaticky přizpůsobí velikosti obrazovky a používanému zařízení, což zlepšuje uživatelskou zkušenost a usnadňuje používání stránek na mobilních zařízeních.

Bootstrap Studio je specializovaný vizuální editor webových stránek a aplikací, který je určený pro vývojáře využívající Bootstrap framework. Editor umožňuje rychle a snadno vytvářet webové stránky bez potřeby psát kód ručně. Tento nástroj poskytuje přístup ke kompletní knihovně komponent, které jsou předem navržené a přizpůsobené pro použití s Bootstrap frameworkem. Má uživatelsky přívětivé rozhraní a funkce drag-and-drop, což umožňuje snadno přidávat prvky, jako jsou tlačítka, formuláře, modální okna, navigační menu a další. Nástroj poskytuje možnost prací s CSS styly zvláštních komponent pomocí vestavěného rozhraní nebo vytvoření vlastních prací s CSS styly.

Backend zajišťuje komunikaci s databází a poskytuje API pro frontend a spojení s výpočetním modulem. Ke zpracování událostí a veškeré logiky aplikace byl zvolen programovací jazyk Typescript, a to jak pro frontendovou, tak i pro backendovou část.

TypeScript je programovací jazyk, který vychází z jazyka JavaScript a jehož cílem je zlepšit vývoj webových aplikací a snížit počet chyb v kódu. JavaScript je programovací jazyk používaný pro vývoj webových stránek a aplikací. Je to skriptovací jazyk, což znamená, že kód JavaScriptu je interpretován a vykonáván v prohlížeči, místo aby byl přeložen na spustitelný soubor. To umožňuje vývojářům vytvářet dynamické webové stránky, které mohou reagovat na uživatelské vstupy a provádět složité operace bez nutnosti načítání nových stránek. TypeScript je rozšířením tohoto jazyku, takže má stejnou funkcionalitu a navíc přidává statickou typovou kontrolu do JavaScriptu, což znamená, že vývojáři mohou definovat typy proměnných, funkcí a objektů, což umožňuje odhalit mnoho chyb již při psaní kódu. Syntax je víc strukturovaná, díky tomu je kód čitelnější a snadněji se udržuje. TypeScript má mnohem bohatší sadu nástrojů a podporu IDE než JavaScript, jako jsou nápovědy a syntaktické a sémantické kontroly.

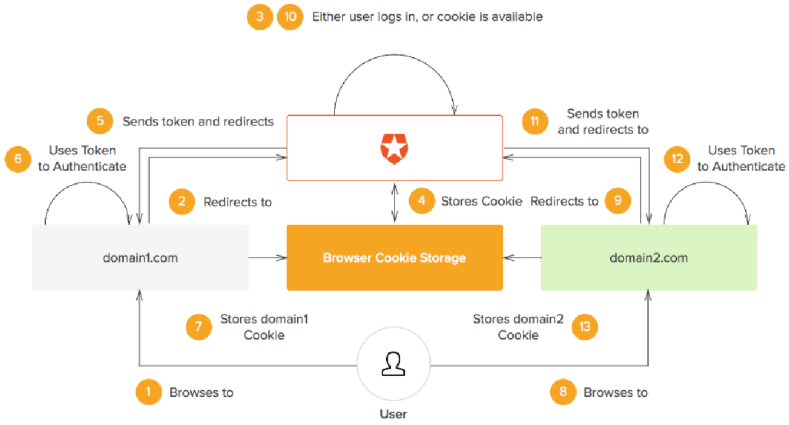
K těmto účelům implementace aplikace probíhala v IDE IntelliJIdea. Je to integrované vývojové prostředí (IDE) vyvíjené firmou JetBrains, které poskytuje širokou škálu funkcí a nástrojů pro vývojáře. Mezi funkce IntelliJ IDEA patří:

- Editor kódu s podporou syntaxe, zvýraznění kódu, automatickým doplňováním a refaktoringem kódu.
- Podpora správy verzí pomocí integrace s Gitem nebo dalšími verzovacími nástroji.
- Podpora správy verzí pomocí integrace s Gitem nebo dalšími verzovacími nástroji.
- Nástroje pro testování a ladění aplikací.
- Integrace s nástroji pro sestavování projektů.
- Nástroje pro analýzu kódu a detekci chyb včetně integrace s nástroji pro statickou analýzu kódu.

K verzování projektu byl použit verzovací systém Git. Jedná se o software s otevřeným kódem, který umožňuje vývojářům pracovat společně na projektech, sledovat změny v kódu a spravovat různé verze aplikace. Git pracuje tak, že ukládá všechny změny v kódu v tzv. "commitech"a udržuje historii verzí kódu. Každý commit obsahuje informace o tom, kdo provedl změny a kdy byly provedeny. Git také umožňuje vývojářům pracovat s větvemi, což je funkce, která umožňuje oddělit vývoj nové funkcionality od stabilní výrobní verze aplikace, což je dobrým způsobem k udržování aplikace vždy v konzistentním stavu.

6.3 Autentizace uživatele pomocí SSO technologie

Pro účely autentizace uživatele bylo rozhodnuto použít metodu ověřování SSO (obrázek 6.5). Single Sign-On (SSO) je technologie, která umožňuje uživatelům přihlašovat se ke všem svým aplikacím a službám pomocí jediného uživatelského jména a hesla. Tento přístup eliminuje potřebu opakovaného zadávání uživatelských údajů pro každou aplikaci nebo službu a umožňuje uživatelům snadný a pohodlný přístup ke všem svým digitálním zdrojům. Jednotné přihlášení funguje na základě vztahu důvěry vytvořeného mezi aplikací a poskytovatelem identity, v tomto případu – NXP.



Obrázek 6.5: Diagram scénáře SSO
Převzato z[10]

Tento vztah důvěry je často založen na certifikátu, který si vyměňují poskytovatel identity a poskytovatel služeb. Certifikát se používá pro podepsání informací o účtu poskytovatelem identity, podle kterého služba vyžadující autentizaci ověří důvěryhodnost zdroje. V SSO mají tato identifikační data formu tokenů, které obsahují informace o uživateli, jako je e-mailová adresa uživatele nebo uživatelské jméno. Postup přihlášení obvykle vypadá takto: uživatel přejde na stránku služby, jak je uvedeno na obrázku výše. Z této stránky se odešle token, obsahující informace o uživateli, poskytovatel identity uživatele zkontroluje, jestli uživatel už byl ověřený a v takovém případě rovnou udělí přístup ke službě bez opakované výzvy k přihlášení. Pokud je uživatel nepřihlášený, bude k tomu vyzván poskytovatelem identity k zadání svých přihlašovacích údajů.

Jakmile poskytovatel identity ověří poskytnuté přihlašovací údaje, odešle službě zpět token potvrzující úspěšné ověření. Tento token je předán přes prohlížeč uživatele a je ověřen podle vztahu důvěry, který byl nastaven mezi službou a poskytovatelem identity během počáteční konfigurace. Uživatel dostane přístup ke službě, a když se pokusí získat přístup k jinému webu, který podporuje SSO metodu, dostal by přístup podle ověřeného tokenu pomocí vztahu důvěryhodnosti nakonfigurovaného s řešením SSO. Sekce je vytvořená s využitím zdrojů [10], [14].

6.4 Zachování dat mezi spuštěními aplikace

Termín "persistence"se používá k popisu schopnosti prohlížeče ukládat data a informace, aby byla zachována i po opuštění stránky nebo uzavření prohlížeče. Existují různé způsoby, jak prohlížeče mohou zachovat data. Například můžou ukládat cookies, což jsou malé soubory, uložené na počítači uživatele nebo formou ukládání dat do lokálního úložiště, které umožňuje webovým stránkám ukládat větší množství dat (například obrázky nebo dokumenty).

V tomto projektu je využitá persistence dat v podobě cookies pro uložení dat, které prohlížeč dostal od poskytovatele identity v rámci SSO komunikace a které se využívají k autorizaci dotazů provedených uživatelskou aplikací nebo k autentizaci uživatele při sledujícím spouštění aplikace nebo při zpětném otevření prohlížeče. Jelikož technologie SSO umožňuje přihlášení pod stejným účtem do různých aplikací, tento způsob ukládání in-

formací pro autentizaci dovoluje používat aplikace podporující SSO technologie pomocí uchovaných cookies bez opakovaného přihlášení.

6.5 Technologie webassembly

WebAssembly je binární instrukční formát a virtuální stroj pro webové prohlížeče, který umožňuje spouštění vysoce výkonných aplikací napsaných v jiných jazycích než JavaScript, například C++, Rust nebo Go přímo v prohlížeči.

WebAssembly byl vytvořen s cílem zlepšit výkon webových aplikací zejména v oblastech, jako jsou vykreslování grafiky a vědecké výpočty. Je to nízkoúrovňový jazyk podobný assembleru s kompaktním binárním formátem, který běží s téměř nativním výkonem a poskytuje jazyky s nízkoúrovňovými paměťovými modely, jako je C++ a Rust, s cílem kompilace, aby mohly běžet na webu. Webová platforma se skládá ze dvou částí:

- Virtuální stroj (VM), který spouští kód webové aplikace.
- Sada rozhraní, která může webová aplikace volat k ovládání funkcí prohlížeče.

Po vzniku WebAssembly pracuje nyní virtuální stroj se dvěma typy kódů – JavaScript a WebAssembly. Různé typy kódů se mohou navzájem volat podle potřeby. Exportovaný kód WebAssembly se zabalí do funkcí JavaScriptu, které lze volat normálním způsobem [15].

WebAssembly nabízí několik výhod oproti tradičnímu vývoji webových aplikací:

- Díky optimalizaci virtuálního stroje pro rychlost a výkon umožňuje spouštět vysoce výkonné aplikace.
- Podporuje mnoho jazyků, jako jsou C++, Rust nebo Go, což umožňuje psát webové aplikace v preferovaném jazyce.
- WebAssembly může být spouštěn na různých platformách a operačních systémech.
- Běží v bezpečném prostředí v prohlížeči a nepřistupuje k citlivým datům na zařízení uživatele, což zvyšuje bezpečnost a ochranu dat.
- WebAssembly může být stažen a dekódován rychleji než kód napsaný v JavaScriptu, což umožňuje rychlejší načítání webových stránek a aplikací.
- Umožňuje rozdělit aplikaci na menší moduly, které mohou být spouštěny na vyžádání, což zvyšuje efektivitu.

Tento přístup byl využit pro oddělení výpočetně náročných operací do zvláštního modulu. Tento modul zajišťuje samotný výpočet matice kamery, parametrů zkreslení čočky a provedení dewarpingu pomocí knihovny OpenCV a je napsaný v jazyce C++. Ve fázi sestavování aplikace se modul napsaný v C++ kompiluje do modulu s rozšířením .wasm. Následně je možné spustit funkce tohoto modulu pomocí volání funkce s TypeScript kódu.

Jelikož pro provedení kalibrace je potřeba předat výpočetnímu modulu obrázky a parametry kalibrace, je využit sdílený souborový systém. K tomu se používá FS (File System) modul, což je modul vestavěný v Node.js, který umožňuje interakci s lokálním souborovým systémem. FS modul poskytuje synchronní a asynchronní metody pro práci se soubory a složkami.

6.6 Využití databáze

Pro účely uchování dat spojených s uživatelem jako jeho nastavení aplikace a sada souborů, vygenerovaných po provedení kalibrace se používá databáze MongoDB. Hlavní výhodou je, že MongoDB používá dokumentový model, který umožňuje ukládat a zpracovávat různé typy dat bez předem definovaného schématu. Jelikož se pro implementace používá Typescript, v aplikaci jsou popsané modely jednotlivých celků jako informace o exportovaných souborech nebo informace o nastaveních uživatele. Je to vhodné pro snadnější práci s těmito informacemi jako s objekty, které mají přesně definovanou strukturu, i když MongoDB byla navržena bez nutnosti definování předem pevné struktury schématu. Následně vytvoření a komunikace s databází probíhá pomocí objektů odpovídajících těmto modelům.

Pravděpodobně dost často bude uživatel provádět kalibraci podle stejných obrázků, což znamená, že by bylo vhodné pro následující spouštění aplikace nebo při znovunačtení stránky zachovat tyto obrázky. Kvůli soukromí uživatele a dalším omezením jako například časová náročnost přenosu velkého obsahu dat po síti se využití vzdálené databáze nevyplatí. Proto je pro tyto účely využito IndexedDB, což je databázová technologie používaná ve webových prohlížečích, která umožňuje ukládat a manipulovat s velkými množstvími dat v rámci webového prohlížeče, aniž by bylo nutné ukládat data na server.

IndexedDB funguje na principu objektově orientovaného úložiště, kde jsou data uložena v objektech a každý objekt má svůj klíč pro rychlé vyhledávání a manipulaci. IndexedDB umožňuje webovým aplikacím ukládat data, jako jsou obrázky, zvukové soubory a další binární data. Využívá k uchování dat diskový prostor počítače, data jsou ukládána přímo v prohlížeči. To umožňuje rychlý přístup k datům a snižuje zpoždění při zpracování dat, jelikož uživatel obvykle nahraje několik obrázků, z nichž každý má rozměr několika megabajtů a je asynchronní, což znamená, že během načítání dat stále probíhá vykreslování uživatelského rozhraní. Data jsou chráněna proti ztrátě nebo poškození, což zlepšuje jejich bezpečnost, neodesílají se na žádný server, což je výhodou při práci se soukromými obrázky uživatele [16].

Po kalibraci se v databázi IndexedDB uchovávají obrázky nahrané uživatelem a další 2 obrázky vygenerované OpenCV knihovnou, z nichž jeden je s odstraněným zakřivením a druhý má označené detekované body kalibračního vzoru, podle něhož se provádí kalibrace. Tento přístup dovoluje aplikaci po dalším spuštění vrátit ji do stavu, ve kterém skončil uživatel.

Kapitola 7

Testování aplikace a návrh rozšíření

7.1 Metody testování

Testování je nejlepší způsob odhalit většinu chyb a nesrovnalosti v chování aplikace. Během vývoje projektu byly využité přístupy manuálního testování a integrační testování.

Manuální testování se v softwarovém vývoji používá k ověřování funkčnosti a kvality aplikace. Při tomto přístupu se procházejí různé testovací scénáře jako například zadání různých vstupů, kliknutí na různá tlačítka, otestování různých funkcí a ověřuje se, jestli se aplikace chová podle očekávání. Je možné testovat aplikaci v různých prostředích, jako například na různých zařízeních nebo v různých prohlížečích. Manuální testování může být prováděno buď ručně, nebo s využitím automatizovaných nástrojů.

Jelikož se při vývoji aplikace používal přístup DevOps a nástroje pro plánování jednotlivých úkolů, po přidání funkcionálního celku programu bylo prováděno většinou manuálně testování přidané funkcionality.

Pro ověření správnosti provedení kalibrace byla vytvořena sada obrázků s kalibračním vzorem šachovnice pomocí kamery s čočkou typu rybího oka, které mají viditelné zakřivení. Testování probíhalo způsobem nahrávání těchto obrázků do aplikace a zhodnoceni výstupu aplikace. Ověřovaly se obrázky po odstranění zakřivení, výstupní xml soubor nebo správnost chybové hlásky. Testování se provádělo po realizaci nové funkcionality, testovala se veškerá funkcionalita spojená s procesem kalibrace a přidané funkce. Při přidání funkcionality, která by mohla ovlivnit chování jiných funkcí, se tyto funkce opakovaně testovaly.

Jelikož hybridní aplikace má dost podobné vykreslování uživatelského rozhraní jako webové aplikace, bylo vhodné aplikovat přístupy testování pro webové aplikace. Při vývoji vzhledu aplikace velkou část času zabírá vytvoření stylování stránky pomocí HTML CSS. Většinou je po úpravě stylů potřeba znovu sestavit projekt a ověřit správnost provedených změn v prohlížeči. Aby tento proces zabíral méně času, výrazně pomohl nástroj Bootstrap Studio, kde bylo možné se zaměřit jenom na vzhled jednotlivých součástek stránky a vidět i testovat změny hned po jejich provedení. Po odladění stylování exportoval se výsledný kód do aplikace. Záměrem testování této části bylo hlavně přizpůsobit aplikace pro zobrazení v různých prohlížečích a na různých zařízeních, která mají různé rozlišení obrazovky.

Další typ testování, které bylo využito – integrační testování. Jelikož během vývoje projektu vždy byla zajištěna funkční verze aplikace a vývoj nové funkcionality probíhal zvlášť, při integraci změn do produkčního prostředí probíhá automatizované integrační tes-

tování. Jeho cílem je ověřit, zda jednotlivé komponenty správně spolupracují a integrují se navzájem a zda výsledný software funguje správně jako celek. Jelikož má aplikace oddělený modul odpovídající za kalibrace obrázku, zvlášť se provádí testování modulu, jestli má správně nakonfigurované a zpřístupněné závislosti a zvlášť ověření závislostí hlavního modulu aplikace.

Cílem integračního testování je minimalizovat riziko chyb v softwaru, které mohou být způsobeny špatnou integrací komponent, a zajistit, že software bude fungovat správně jako celek.

Vylepšení procesu testování

Pro zlepšení efektivity testování aplikace bude vhodné využití možnosti automatizovaného testování. K tomu je možné využít nástroje a frameworky pro napsání automatických testů. Pro testování uživatelského rozhraní je možné využít nástroj Selenium.

Selenium je sada programů s otevřeným zdrojovým kódem, které se používají k testování webových aplikací a správě stránek. Programy Selenium umožňují automatizovat akce prohlížeče.

Mezi jeho hlavní součástky patří Selenium IDE a Selenium Web driver. Selenium IDE je modul prohlížeče pro záznam akcí uživatele a jejich přehrávání pro testování. Jedná se o knihovnu Selenium s grafickým rozhraním a možnostmi pro práci s testovacími skripty webových stránek. IDE generuje kód pro Selenium WebDriver, který přehraje zaznamenané akce uživatele.

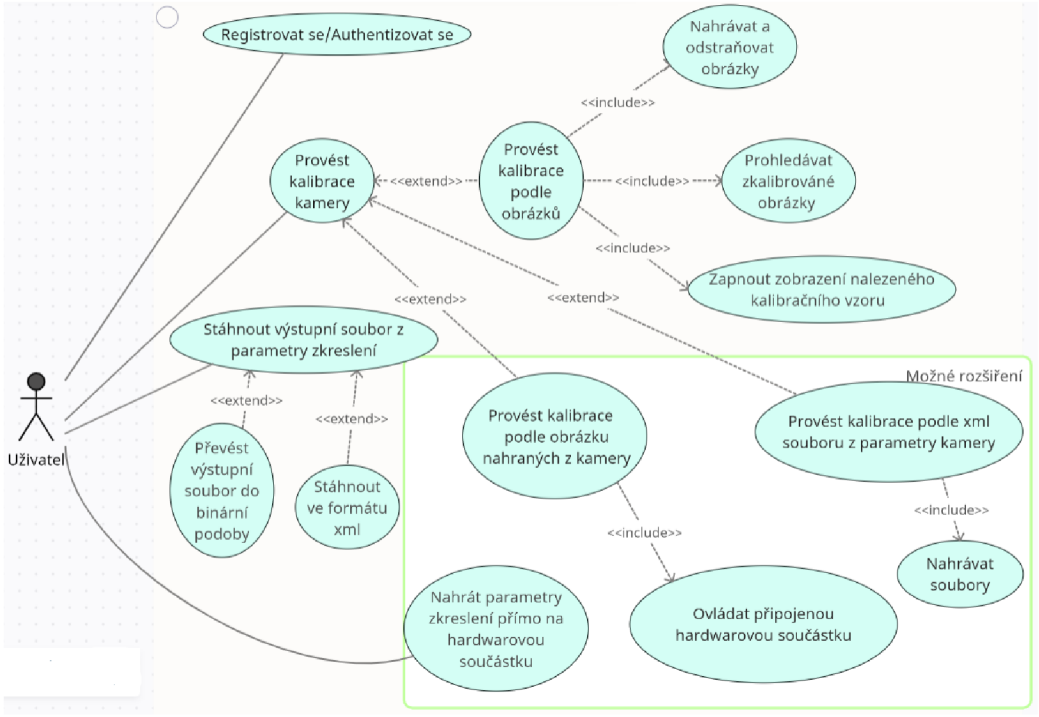
Selenium WebDriver je sada ovladačů pro různé prohlížeče a sada klientských knihoven v různých programovacích jazycích pro práci s ovladači. Selenium WebDriver je softwarová knihovna, která nemá uživatelské rozhraní, ale umožňuje různým dalším programům komunikovat s prohlížečem, ovládat jeho chování, přijímat některá data z prohlížeče a provádět v prohlížeči nějaké příkazy.

Spolu se Selenium WebDriver jde použít jeden z frameworku pro napsání testů jako například Citrus Framework. Pomocí tohoto frameworku je možné odesílat a přijímat dotazy pomocí různých protokolů a následně validovat odpovědi, porovnávat možné hlásky. Nástroj dovoluje pracovat s JSON soubory, ověřovat schémata XML souborů, porovnavat xml stromy. Pracovat s frameworkem je možné v IDE IntelliJ IDEA, která byla využita v tomto projektu.

7.2 Možné rozšíření aplikace

Hlavním cílem vytvoření bakalářské práce je návrh hybridní aplikace pro kalibrace kamery a realizace uživatelského rozhraní s možností provedení kalibrace, odstranění zakřivení z obrázků a vygenerování výstupního souboru s parametry zakřivení. Ovšem hlavním záměrem aplikace a jejího dalšího rozšíření je zamýšleno ovládání hardwarovou součástkou připojenou k uživatelskému zařízení přímo pomocí této aplikace přes prohlížeč. Toto rozšíření by mělo přidat možnost načíst parametry připojené hardwarové součástky, mít možnost s jeho pomocí vytvořit obrázky a následně zapsat vypočtené koeficienty zkreslení čočky pro aplikování korekce obrázků pomocí hardwarového dewarpingu (obrázek 7.1).

Momentálně modul odpovídající za samotné provedení kalibrace využívá přistup webassembly a provádí veškeré výpočty pomoci prohlížeče. Lepší realizaci bylo by přesunout výpočty na server, a umožnit provedení kalibraci pomoci agentů, bežících na serveru. Při dotazu na provedení kalibrace, jeden z agentů převezme ten požadavek, spustí funkcionalitu odpovídající za kalibrace a vrátí výstupní soubory uživatelovi, který na to dotazoval (podobna funkcionalita existuje na jiném projektu firmy NXP – Voice recognition tool 4.4.1).



Obrázek 7.1: Diagram případů užití pro možné rozšířeni

Kapitola 8

Závěr

Cílem této bakalářské práce je navrhnout aplikace hybridního typu, která by pomohla vyřešit problém zkreslení obrázků způsobeného fyzickým zakřivením čočky kamery. Z tohoto důvodu byly srovnány různé typy aplikací jako webová, nativní a hybridní a popsány jejich výhody, provedené rešerše mezi nástroje, které můžou pomoct s vyřešením problému zkreslení obrázku. Jako nejvíc vhodný nástroj pro provedení kalibrace, bylo zvoleno využiti knihovny OpenCV.

Byl popsán proces provedení kalibrace kamery cestou výpočtu vnějších a vnitřních parametrů kamery, výpočet matice kamery a s tím spojené využití knihovny OpenCV. Provedení kalibrace vyžaduje od uživatele nahrát sadu obrázků obsahujících jeden z kalibračních vzorů a zadat parametry kalibrace. Podle těchto požadavků byl popsán návrh uživatelského rozhraní pro aplikace pomocí Figmy a následně jeho realizace. Před popisem implementace bylo navrženo a porovnáno několik možných architektur aplikace. Podle zvolené architektury byly vybrány a popsány nástroje a technologie, které byly využité při vývoji uživatelského rozhraní, implementace modulu odpovídajícího za samotnou kalibraci, komunikace různých částí aplikace a použité úložiště dat.

Poslední část popisuje testování a další rozšíření aplikace jako přímé připojení a komunikace s hardwarovými součástkami, které potřebují kalibraci.

Literatura

[1] *A Generic Camera Model and Calibration Method for Conventional, Wide-Angle, and Fish-Eye Lenses* [online]. [cit. 2023-04-24]. Dostupné z: https://ieeexplore.ieee.org/document/1642666.

[2] Budziński, M. *What Is React Native?* [online]. 2021 [cit. 2023-04-06]. Dostupné z: https://www.netguru.com/glossary/react-native#what-is-react-native.

[3] *Co je DevOps?* [online]. [cit. 2023-04-11]. Dostupné z: https://azure.microsoft.com/cs-cz/resources/cloud-computing-dictionary/what-is-devops/.

[4] *Electron framework* [online]. [cit. 2023-04-06]. Dostupné z: https://www.electronjs.org/.

[5] *Fisheye Calibration Basics* [online]. [cit. 2023-04-11]. Dostupné z: https://www.mathworks.com/help/vision/ug/fisheye-calibration-basics.html#mw_f299f68d-403b-45e0-9de5-55203a09460d.

[6] *Flutter* [online]. [cit. 2023-04-06]. Dostupné z: https://flutter.dev/.

[7] *Ionic framework* [online]. [cit. 2023-04-06]. Dostupné z: https://ionicframework.com/.

[8] Koďousková, B. *VÝVOJ HYBRIDNÍ APLIKACE PRO PODNIKÁNÍ, SROVNÁME PRO A PROTI* [online]. [cit. 2023-04-06]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-hybridni-aplikace.

[9] *MongoDB Installation* [online]. [cit. 2023-05-01]. Dostupné z: https://www.mongodb.com/docs/manual/installation/.

[10] Peyrott, S. *What Is Single Sign-On Authentication (SSO) And How Does It Work?* [online]. [cit. 2023-04-11]. Dostupné z: https://auth0.com/blog/what-is-and-how-does-single-sign-on-work/.

[11] SHAH, S. *SSH Tunneling Explained* [online]. 2021 [cit. 2023-04-06]. Dostupné z: https://goteleport.com/blog/ssh-tunneling-explained/.

[12] *Camera calibration With OpenCV* [online]. [cit. 2023-04-11]. Dostupné z: https://docs.opencv.org/4.x/d4/d94/tutorial_camera_calibration.html.

[13] *What is camera calibration* [online]. [cit. 2023-04-11]. Dostupné z: https://de.mathworks.com/help/vision/ug/camera-calibration.html.

[14] *How Does Single Sign-On Work?* [online]. [cit. 2023-04-11]. Dostupné z: https://www.onelogin.com/learn/how-single-sign-on-works.

[15] *WebAssembly Concepts* [online]. [cit. 2023-04-11]. Dostupné z: https://developer.mozilla.org/en-US/docs/WebAssembly/Concepts.

[16] YAKUBU, V. *How to Use IndexedDB* [online]. [cit. 2023-04-13]. Dostupné z: https://www.freecodecamp.org/news/how-indexeddb-works-for-beginners/.

Příloha A

Obsah CD

Obsahem přiloženého CD je funkční aplikace včetně souboru využitého framevorku. Soubory které byly vytvořené v rámci bakalářské prací jsou jenom ve složkách uvedených dále:

- dependencies/io-dct-gui/source/typescript/Io/Dct/Gui - obshuje skripty odpovidajíci za praci s frontendem:
  - /Components, /Dialogs, /Pages, /Panels, /UserControls - odpovídají za vytvoření jednotlivých komponent html stránky.
  - /Controllers - skripty přidávající ovládaní elementy stranky.
  - /Bindings, /Utils - pomocne skripty pro validace vstupu, nebo autentizace uživatele.
- source/typescript/Io/Dct/Application- obsahuje skripty pro backend a připojeni frontendu:
  - /DAO - složka obsahující součástky pro vytvoření schematu databáze.
  - /Utils - skripty ovládající práce s databázi (/ExportsManager, /SettingsManager) a další pomocné skripty.
  - /Conectors - skripty odpovídající za propojeni frontendu a backendu.
- Společné využiti pro složky:
  - /HttpProcessor - přesměrovaní dotazů mezi odpovídajícími kontroléry.
  - /Interfaces - pro vytvoření šablony datových objektů použitých v aplikaci.
- Dalši složky
  - dependencies/io-dct-gui/resource/sass - stýlování aplikace.
  - /dependencies/io-dct-gui/resource/graphics - použite obrázky.
  - /input - testovací sada obrázku pro provedení kalibrace.
  - /output - zkalibrované obrázky, obrázky s nalezeným kalibračním vzorem, xml soubor s parametry zkreslení.

Další složky obsahuji závislosti projektu a konfigurační soubory pro spuštění aplikace na interních serverech firmy a nejsou součásti moji práce na tomto projektu.

Příloha B

Spuštění aplikace

Spuštěni vyžaduje určitou minimální přípravu na straně uživatele a znalost Linuxu.

Obsah /build složky obsahuje soubory pro spuštění aplikace, ale je přizpůsobeny pro běh v docker kontejneru. Kvůli tomu složka obsahuje skripty ve složce source/docker, které starají o spuštění aplikace přes docker. Jelikož image jako takový nelze dodat z licenčních důvodu, ve složce jsou konfigurační soubory pro docker a dodává se kompletní software včetně zdrojových souboru.

Danou aplikaci je možné spustit i lokálně na Linuxu pomoci skriptu build/target/cmd/DctApplication.sh s argumentem "start"pokud na dánem počítací běží MongoDB na výchozím portu 27017. MongoDB lze nainstalovat dle tohoto návodu:[9].

Pro spuštění aplikace je nutne:

- nainstalovat MongoDB.
- spustit MongoDB na portu 27017
- spustit skript build/target/cmd/DctApplication.sh s argumentem "start"

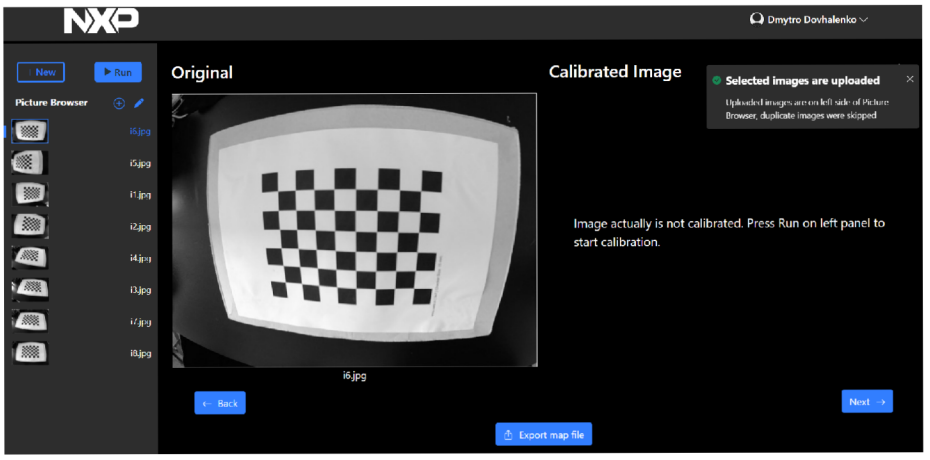
Příloha C

Výstup aplikace

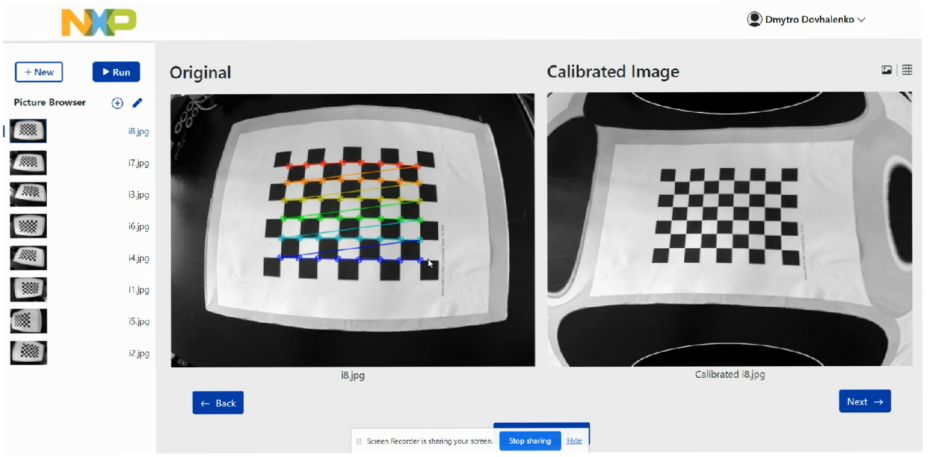
Příklad obsahu výstupního xml souboru po provedení kalibrace:

```
<?xml version="1.0"?>
<opencv_storage>
<calibration_time>"Wed May 03 15:52:42 2023"</calibration_time>
<nr_of_frames>8</nr_of_frames>
<image_width>640</image_width>
<image_height>480</image_height>
<board_width>8</board_width>
<board_height>6</board_height>
<square_size>50.</square_size>
<fix_aspect_ratio>1.</fix_aspect_ratio>
<!-- flags: +fix_aspectRatio +fix_principal_point
+zero_tangent_dist +fix_k4 +fix_k5 -->
<flags>6158</flags>
<fisheye_model>0</fisheye_model>
<camera_matrix type_id="opencv-matrix">
  <rows>3</rows><cols>3</cols>
  <dt>d</dt>
  <data>
    4.4599939027123446e+02 0. 3.1950000000000000e+02 0.
    4.4599939027123446e+02 2.3950000000000000e+02 0. 0. 1.</data></camera_matrix>
<distortion_coefficients type_id="opencv-matrix">
  <rows>5</rows><cols>1</cols>
  <dt>d</dt>
  <data>
    -6.2878186677362491e-01 5.6318492556325817e-01 0. 0.
    -3.0811905900254333e-01</data></distortion_coefficients>
<avg_reprojection_error>1.0337018647189380e+00</avg_reprojection_error>
<per_view_reprojection_errors type_id="opencv-matrix">
  <rows>8</rows><cols>1</cols>
  <dt>f</dt>
  <data>
    9.41045523e-01 1.25983763e+00 1.37248254e+00 7.05023170e-01
    1.23254740e+00 5.79994857e-01 1.16822159e+00 6.88827872e-01
```

```
  </data></per_view_reprojection_errors>
<!-- a set of 6-tuples (rotation vector + translation vector) for each view -->
<extrinsic_parameters type_id="opencv-matrix">
  <rows>8</rows><cols>6</cols>
  <dt>d</dt>
  <data>
    -1.6221910899898667e-01 7.8609930141056134e-02
    2.4685984023379601e-02 -1.7623082278022028e+02
    -1.7974046803049598e+02 ...
    5.1641987496241450e+02</data></extrinsic_parameters>
```



Obrázek C.1: Stav aplikace před kalibraci



Obrázek C.2: Nalezený vzor a zkalibrovaný obrázek