Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Diplomová práce

Prezentace produktů pomocí rozšířené reality

Bc. Jiří Doležal

Vedoucí práce: Ing. David Sedláček Ph.D.

26. května 2017

Poděkování

Na tomto místě bych rád poděkoval svému vedoucímu Ing. Davidu Sedláčkovi Ph.D. za ochotu a pomoc při vypracovávání této práce, své přítelkyni Kateřině Břicháčkové za trpělivost a pomoc při tvorbě a testování a své rodině za podporu v tomto projektu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 26. května 2017 .....

České vysoké učení technické v Praze
Fakulta elektrotechnická
© 2017 Jiří Doležal. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě elektrotechnické. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí, je nezbytný souhlas autora.

Odkaz na tuto práci

Doležal, Jiří. *Prezentace produktů pomocí rozšířené reality*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta elektrotechnická, 2017.

Abstract

With cellphone performance rising rapidly, applications using virtual or augmented reality became widespread. However, in most cases the functionality is a quick technology showcase and nothing more. This project presents a desing and implementation of a mobile app for virtual wristwatch presentation, using 3D model tracking with augmented reality. Application is supported by an editorial webserver implemented in PHP framework Symfony. Mobile application is developed for the Android operating system, but can be deployed for other platforms as well. These two subjects then communicate with each other using the REST protocol, where the webserver is used as a storage for all wristwatch data and the mobile application then downloads those data and presents them to the user.

Keywords Android, application, augmented reality, webserver, náramkové hodinky

Abstrakt

Se stále se zvyšujícím výkonem mobilních zařízení se popularita aplikací využívajících virtulní/rozšířené reality na mobilních zařízeních prudce šíří, avšak ve většině případů se jedná o jednoduchou ukázku technologie a nic více. Tato práce představuje návrh a implementaci mobilní aplikace pro prezentaci virtuálních náramkových hodinek okolo celé ruky uživatele pomocí sledování 3D objektů rozšířené reality. Aplikace je podpořena webovým serverem implementovaným v PHP frameworku Symfony. Mobilní aplikace je odzkoušena primárně pro operační systém Android, avšak vyvinuta multiplatformně a nasaditelná tedy i na ostatní platformy. Dvojice aplikace-server poté využivá komunikace pomocí REST protokolu, kdy na webový server jsou nahrána veškerá data a mobilní aplikace tyto data získává a uživateli zobrazuje virtuální náramkové hodinky.

Klíčová slova Android, aplikace, rozšířená realita, webový server, náramkové hodinky

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Motivace

V dnešní době mobilní aplikace zdaleka nenabízejí pouze jednoduché funkce, jako je lepší hudební přehrávač či editor dokumentů. Každým rokem se výpočetní výkon mobilních zařízení zvyšuje, a tím se otevírají nové možnosti pro tvorbu aplikací. V několika posledních letech zažila obrovský boom virtuální realita. Uživatel nasadí na hlavu speciální headset s displejem, na který je promítán virtuální, počítačově generovaný svět. Zpočátku bylo tohoto dosahováno pomocí speciálních náhlavních souprav a programy pro virtuální realitu byly tak náročné, že bylo zapotřebí výkonných počítačů. Výkon mobilních telefonů ale vzrostl natolik, že bylo brzy možné tyto aplikace spustit i na nich (ačkoliv stále v nižším rozlišení). Programy pro virtuální realitu mají u veřejnosti úspěch, avšak uživatel stále musí zakoupit speciální headset a aplikace lze používat pouze v určitých prostředích, jelikož uživatel nevidí (a často ani neslyší) nic z reálného světa.

Tato nepraktičnost dala příležitost novému typu aplikací - takovým, které spojují technologickou senzaci virtuálního obsahu s reálným světem. V současné době existuje již mnoho mobilních aplikací, které rozšířenou realitu využívají k velkému množství různých účelů - od vzdělávacích přes architektonické až po medicínské a archeologické. Aplikace byly samozřejmě vyvinuty i v rámci prezentací různých produktů, avšak většina dosavadních řešení používá jednoduché sledování 2D obrazu a vytvořený zážitek není dostatečně přesvědčivý.

V rámci této práce je prozkoumána možnost vizualizace 3D virtuálních hodinek v plném rozsahu 360°kolem uživatelovy ruky a zároveň možnost využití tohoto způsobu pro nasazení ve velkém měřítku a znovupoužití již

existujících obrazových dat těchto hodinek.

Kapitola 1

Augmentovaná realita

Na úvod je nejdříve nutno vysvětlit, co se pod pojmem augmentovaná realita skrývá. Jedná se o typ „smíšené reality“ (také známé jako hybridní realita), což je spojení reálných a virtuálních světů pro tvorbu nových prostředí a vizualizací, kde fyzické a digitální objekty existují a interagují spolu v reálném čase. Graficky se jedná o osu zobrazující přechod smíšenou realitou od reálného prostředí skrz augmentovanou realitu a augmentovanou virtualitu až po virtuální prostředí. Augmentovaná realita je tedy přímý či nepřímý pohled na fyzický, reálný svět, jehož součásti jsou „augmentovány“, neboli podpořeny počítačově generovaným obsahem, jako například zvukem, obrazovými daty, GPS informaci či jiným obsahem. Opakem k augmentované realitě je poté virtuální realita, kdy je reálný svět zaměněn za počítačově generovaný, simulovaný.

1.1 Hardwarové řešení

Systém pro augmentovanou realitu se skládá ze 4 základních součástí:

Senzory Pro zachycování vstupních dat pro rozšířenou realitu jsou potřeba rozličné senzory. Základním prvkem je kamera, která snímá scénu v reálném čase. K ní je poté možno přidat další senzory, jako je akcelerometr, kompas a GPS senzor.

Vstupní zařízení Slouží pro interakci uživatele se systémem a například se jedná o dotykovou obrazovku mobilního telefonu, pomocí které uživatel aplikaci ovládá, či o mikrofon spojený se systémem rozpoznávání hlasu.

Alternativně se jedná například i o externí zařízení spojené se systémem rozpoznávání gest a pohybů těla.

Display Výstupní zařízení systému, kde se uživateli zobrazí výsledný obraz doplněný o augmentovaná data.

Procesor Řídící jednotka celého systému, probíhá zde zpracování vstupních dat ze senzorů a vstupních zařízení a zobrazení výsledku na výstupní display. V případě přítomnosti kamery toto zahrnuje analýzu obrazu a doplnění augmentovaných prvků do již existující scény.

Ačkoliv existuje řada různých hardwarových řešení pro použití augmentované reality, v této práci se budeme zabývat hlavně řešením použitým v mobilních zařízeních. Tato zařízení jsou z velké části vybavena kamerou a dodatečnými senzory, které zařízení předurčují k snadnému vývoji aplikací. Ty poté mohou být nasazeny pro využití velkou skupinou uživatelů. Jiný příklad využití je například použití ve speciálních brýlích, u kterých je reálný svět snímán kamerou a elementy rozšířené reality jsou poté zobrazeny na sklíčka těchto brýlí. Podobným řešením je tzv. Head-Up-Display, což je průhledná obrazovka zobrazující data přímo do zorného pole uživatele. Tato technologie technicky vzato předcházela augmentované realitě, jelikož na začátku se zobrazovaná data přímo netýkala analyzované scény. V průběhu času se však tento způsob implementoval i do HUD.

V současné době se také experimentuje s využitím speciálních kontaktních čoček, jež budou obraz zobrazovat přímo na svůj povrch, či s tzv. virtuálním sítnicovým displayem. Tato technologie umožňuje „naskenování“ virtuálního displaye přímo na sítnici uživatelova oka – ten potom vidí obraz vznášející se přímo v prostoru před ním.

1.2 Softwarové řešení

Klíčovým aspektem augmentované reality je schopnost realisticky integrovat augmentovaný obsah do reálné scény. Tento proces má za úkol odvodit souřadný systém reálného světa, který je nezávislý na kameře, ze vstupních obrazů. K tomu je použito množství metod a algoritmů z odvětví počítačového vidění. Celý proces má dvě části – nejdříve jsou analyzovány významné body, statické referenční body, či optický tok ve vzorových obrazech. Ty jsou poté porovnány se záznamem ze vstupního zařízení, a pokud je nalezena dostatečná shoda, z výsledků je v druhém kroku získán souřadný systém reálného světa. V této části práce se hlavně zaměříme na první krok

procesu, a to na analýzu zdrojového obrazu, jelikož pokud je již nalezen sledovaný objekt v cílové scéně, rekonstrukce pozice kamery je triviální.

1.2.1 Detekce příznaků

V rámci počítačového vidění, a tedy i rozšířené reality, se jedná o analýzu zdrojových obrazů a rozhodování v rámci každého pixelu, zda se na daném místě nachází nějaký příznak. Nalezené příznaky jsou poté seskupovány do větších celků dle konkrétního požadavku.

Příznak Ačkoliv se jedná o možná nejpoužívanější termín v poli rozšířené reality, neexistuje přesná definice, co vlastně příznak je. Jedná se o zajímavou část obrazu, jejíž přesný tvar je určen konkrétním požadavkem, a která dokáže být analyzována. Na základě nalezení těchto příznaků je následně možno provádět komplexnější analýzu daného obrazu.

1.2.2 Typy příznaků v obrazech

Hrany Hranové body jsou takové body, ve kterých je ostrý přechod mezi dvěma oblastmi obrazu. Obecně, hrany jsou množiny bodů v obraze, které mají velkou hodnotu gradientu. Většina algoritmů pro detekci hran tyto body dle určitých pravidel (například minimální gradient či tvar) seskupují do ucelených množin reprezentující danou hranu. Ačkoliv obecně se hrany považují za jednodimenzionální objekt, mohou dosahovat různých tvarů.

Body zájmu Body zájmu jsou definovány jako oblasti mající velkou úroveň křivosti gradientu. Původní výraz zahrnoval termín „rohy“ nebo „rohové příznaky“, což plyne z prvních algoritmů, které používaly k určení těchto bodů algoritmy pro detekci hran – v místech, kde byl nalezena prudká změna směru, byl poté označen tento bod. Nové algoritmy však tyto body detekují i v jiných případech, například v případu jasného místa na jinak tmavém okolí.

Oblasti významných bodů Označovány také jako „blobs“, oblasti významných bodů jsou doplňkem bodů zájmu v rozpoznávání obrazu. Zatímco „rohové body“ označují pouze malou oblast či dokonce jediný bod, tyto oblasti označují větší plochu v daném obraze. Jejich hlavní využití je pro oblasti, které nejsou dostatečně kontrastní pro detekci jednotlivých bodů zájmu.

Tabulka 1.1: Seznam algoritmů pro detekci příznaků a jejich klasifikace

Název metody	Hrany	Rohy	Oblasti
Canny	X		
Sobel	X		
Kayyali	X		
Harris & Stephens / Plessey / Shi–Tomasi	X	X	
SUSAN	X	X	
Shi & Tomasi		X	
Level curve curvature		X	
FAST		X	X
Laplacian of Gaussian		X	X
Difference of Gaussians		X	X
Determinant of Hessian		X	X
MSER			X
PCBR			X
Grey-level blobs			X

1.2.3 Seznam algoritmů pro detekci příznaků

V současné době existuje řada algoritmů pro detekci příznaků v obrazech. Některé jsou specializované pro konkrétní typ příznaků, jiné detekují více typů. V 1.1 je znázorněn seznam hlavních používaných metod a jejich klasifikace.

1.2.4 Popis a vyhledávání příznaků

Ve většině případů je nutno nejenom příznaky detekovat, ale i správně je popsat a uložit, aby bylo poté možno tyto příznaky zpracovávat a, konkrétně v případě rozšířené reality, porovnávat s jinými obrazy, zda se daná množina příznaků nachází i v tomto novém obraze. Většina algoritmů tohoto typu musí nejdříve použít algoritmy na detekci příznaků. Ty se poté analyzují, shlukují do skupin a přiřazují se jím dodatečné informace, jež je činí invariantními vůči různým transformacím.

Jedním z prvních významných algoritmů tohoto typu je SIFT, jehož funkčnost je popsána níže. Tento algoritmus byl základem mnoha dalších algoritmů, jako je například SURF a HOG, které nabídly stejnou robustnost vůči transformacím a navíc přinesly značné zrychlení výpočtu.

Obrázek 1.1: Vstupní obraz algoritmu SIFT (vpravo) a hledané objekty (vlevo) [9]

Obrázek 1.2: Výsledek algoritmu SIFT, hledané objekty jsou barevně zvýrazněny [9]

1.2.4.1 SIFT

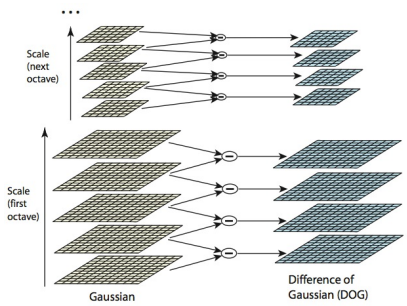
SIFT, neboli Scale-Invariant Feature Transform, je algoritmus pro detekci a popis lokálních příznaků (features) v obrazech. Tento algoritmus byl patentován na University of British Columbia a zveřejněn Davidem Lowe v roce 1999 [9].

Jedním z hlavních použití algoritmu SIFT je identifikace jakéhokoliv objektu v cílovém obrazu. Z obrazu jsou nejdříve získána příznaková data, která jsou poté porovnána s uloženými daty získanými analýzou vzorového obrazu. Objekt je následně rozpoznán na základě přesnosti shod těchto příznaků. Ukázku vstupu a výstupu lze vidět na 1.1 a 1.2. Na prvním obrázku vidíme hledané objekty na levé straně a cílový obraz na pravé. Na druhém obrázku je výstup algoritmu, který identifikoval tyto objekty v obraze, ačkoliv byly některé z nich otočeny či částečně skryty.

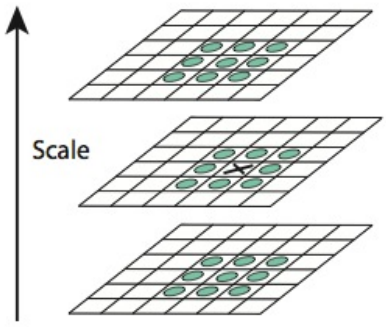
Algoritmus je velmi robustní a příznaková data jsou invariantní k uniformní změně měřítka a rotaci, odolná proti změně osvětlení cílového obrazu a částečně invariantní vůči afinním transformacím.

Popis algoritmu V následující sekci je popsáno základní fungování algoritmu, který se skládá ze čtyř kroků. První krokem je extrakce příznakových dat. Druhý až čtvrtý krok zahrnují vyhledávání v těchto datech, nalezení shody a filtraci potenciálních falešných pozitiv.

Detekce invariantních vlastností obrazu Prvním krokem je nalezení potenciálně významných bodů, z nichž každý je invariantní k posunu, změně velikosti a rotaci vstupního souboru, částečně invariantní ke změně nasvícení obrazu a robustní vůči lokálním změnám geometrie.



Obrázek 1.3: Rozdíl Gaussových funkcí [10]



Obrázek 1.4: Vyhledávání lokálních extrému v 26-okolí [10]

Vstupní soubor je několikrát převzorkován a rozostřen a je vytvořen diskrétní scale-space, což je 3D reprezentace obrazu skládající se z několika vrstev, kde první vrstva odpovídá původnímu obrazu a každá další je tvořena obrazem s vyšším měřítkem. Postup spočívá v konstrukci Gaussovy pyramidy skládající se z obrazů vznikajících konvolucí Gaussova jádra o různém měřítku (šířka jádra) se vstupním obrazem. Scale-space je poté zkonstruován z rozdílových obrazů (jen se nazývají rozdílem Gaussových funkcí 1.3), které vzniknou rozdílem dvou sousedících obrazů v Gaussově pyramidě.

Lokální minima a maxima jsou nalezeny analýzou 26-okolí každého bodu (8-okolí v daném obrazu a 2 x 9 okolí v okolních obrazech), viz 1.4. Pokud má daný bod v tomto okolí největší či nejmenší hodnotu, stává se kandidátem na významný bod. Kandidáti s nízkým kontrastem vůči okolí a body ležící podél hran jsou poté odstraněny. Získané body jsou nyní invariantní ke změně měřítka a jsou nazývány významnými body.

Pro získání invariance vůči rotaci je těmto významným bodům ještě nutno přiřadit dominantní orientaci. Pro každý zkoumaný bod je nejdříve nalezen měřítkově nejbližší obraz (který se nachází v předem sestrojené Gaussově pyramidě), čímž je zajištěna měřítková nezávislost. Pro každý takový obraz je sestrojena velikost a orientace gradientu pomocí rozdílů jasových hodnot. Následně je zkonstruován histogram orientací složený z gradientů z okolí významného bodu. Dominantní orientace je poté určena globálním maximem v tomto histogramu. Poslední fází v tomto kroku je poté sestavení deskriptorů, jež jsou prostorové histogramy obrazových gradientů, které popisují okolí jednotlivých významných bodů způsobem zaručujícím nezávislost na geometrických transformacích a změnách v jasu

obrazu.

Indexace a párování významných bodů Indexování spočívá v ukládání vytvořených SIFT klíčů a identifikaci odpovídajících klíčů z druhého obrazu. Pro efektivní vyhledávání je využito algoritmu Best-bin-first search. Jedná se o aproximační algoritmus pro vyhledávání nejbližšího souseda (nearest neighbour search) v mnohodimenzionálním prostoru. Tento algoritmus používá pro vyhledávání upravený k-d vyhledávací strom, ve kterém je vyhledáváno prioritně od nejmenší vzdálenosti od dotazových souřadnic. Jelikož se jedná o aproximační algoritmus, negarantuje nalezení správného výsledku ve všech případech, avšak toto vyvažuje mnohem efektivnějším časem běhu.

Jako nejlepší kandidát pro každý významný bod je nalezen nejbližší soused v databázi významných bodů vzorového obrazu, kde nejbližší soused je definován jako klíčový bod s nejnižší Euklidovskou vzdáleností od daného významného bodu. Pravděpodobnost správnosti shody lze spočítat jako poměr vzdálenosti nejbližšího souseda proti vzdálenosti druhého nejbližšího. Standardně jsou poté zahozeny veškeré shody, ve kterých je poměr vzdálenosti větší než 0.8, což odstraní přibližně 90% falešných pozitiv, zatímco je zahozeno pouze 5% správných výsledků. Pro urychlení algoritmu je také Best-bin-search utnut po prozkoumání prvních 200 kandidátů, což v databázi o 100.000 klíčích znamenalo zrychlení o dva řády.

Shlukování bodů Ačkoliv bylo v předchozím kroku eliminováno mnoho falešných shod, stále existuje velká pravděpodobnost, že byly nalezeny shody, které nepatří do nalezeného objektu. Z tohoto důvodu je provedeno shlukování příznaků, které náleží jednomu objektu, a hromadné odmítnutí všech shod, jež do tohoto objektu nenáleží. Toto je provedeno Houghovou transformací. Každý bod „hlasuje“ pro množinu konfigurací, jež jsou konsistentní s pozicí, měřítkem a orientací daného bodu. Množiny bodů, jež získají alespoň 3 hlasy, jsou identifikovány jako kandidáti na shodu.

Ověřování metodou nejmenších čtverců Pro každou množinu kandidátů z předchozího kroku je použita metoda nejmenších čtverců pro nejlepší odhad parametrů afinní transformace vzorového obrazu na vstupní obraz. Pokud projekce klíčového bodu použitím těchto parametrů leží v určité toleranci, shoda je ponechána. Pokud po odstranění anomálií zbývá v množině méně než tři body, je shoda zahozena. Metoda nejmenších čtverců je opakována do doby, kdy již není zahozena žádná shoda.

1.2.4.2 SURF

Další z řady algoritmů pro popis a vyhledávání příznaků je částečně inspirován algoritmem SIFT. Prezentován a patentován byl v roce 2006 Herbertem Bay, SURF, neboli Speeded Up Robust Features je několikrát rychlejší než dosavadní SIFT. Pro detekci významných bodů používá tento algoritmus celočíselnou aproximaci Hessiánu (determinant Hessovy matice), což je jeden z algoritmů pro detekci oblastí bodů zájmu. Tato operace vyžaduje pouze tři celočíselné operace pomocí předpočítaného integrálního obrazu. Popisování příznaků je založeno na součtu výsledku Haarových příznaku okolo významného bodu, což může být také spočítáno s využitím integrálních obrazů.

1.2.4.3 HOG

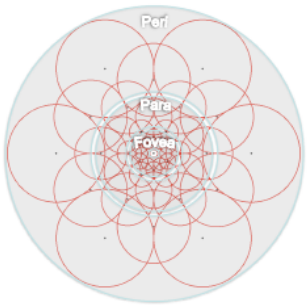
Robert K. McDonnell poprvé předložil základní koncept tohoto algoritmu již v roce 1986 [1]. Do širokého povědomí vstoupil však až v roce 2005, kdy byl prezentován na konferenci CVPR již pod nynějším názvem HOG, neboli Histogram of Oriented Gradients [5]. Tento algoritmus je postaven na faktu, že vzhled a tvar lokálního objektu v obraze lze vyjádřit pomocí distribuce intenzity gradiánů či obecných směrů hran. Nejdříve je obraz rozdělen do menších, spojených oblastí („buněk“). Pro pixely v každé buňce je poté spočítán histogram směrů gradientu. Výsledný deskriptor je poté spojením těchto histogramů. Pro zlepšení invariance vůči změnám v osvětlení mohou být lokální histogramy normalizovány. Nejdříve je spočtena hodnota osvětlení napříč větší oblastí (obsahující více buněk) zdrojového obrazu a toto je poté použito pro normalizaci všech buněk v této oblasti.

Deskriptory získané algoritmem HOG mají několik výhod – jelikož je rozpoznávání prováděno v rámci lokálních oblastí, jsou tyto deskriptory invariantní vůči geometrickým a fotometrickým transformacím. Tento algoritmus je vhodný obzvláště pro rozpoznávání osob v obrazech, jelikož správné vzorkování a fotometrická normalizace dovoluje rozpoznání lidského těla i případě chůze či jiných případů, kdy se tělo pohybuje.

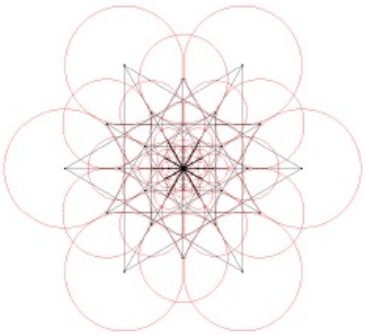
1.2.4.4 FREAK

Binární deskriptory Předchozí algoritmy pro detekci a hledání příznaků používaly histogramy gradiánů. Tyto operace pro každý pixel jsou výpočetně náročné, a ačkoliv SURF toto řešení urychlil pomocí integrálních obrazů, stále se nejedná o dostatečně rychlé řešení pro jisté real-time aplikace.

[1] http://www.google.co.uk/patents/US4567610



Obrázek 1.5: Vzorkovací schéma FREAK[18]



Obrázek 1.6: Kontrolní páry pro kompenzaci orientace FREAK[18]

Z tohoto důvodu byly představeny binární deskriptory. Motivací je značná rychlost jejich porovnávání, jelikož každý deskriptor, jak již název napovídá, je binární řetězec. Pro výpočet Hammingovy vzdálenosti je zde potřebná pouze jedna operace typu XOR mezi dvěma testovanými řetězci.

Každý binární deskriptor se skládá ze třech základních součástí – vzorkovacího schématu, kompenzace orientace a vzorkovaných párů. Pokud chceme vytvořit deskriptor pro určitý klíčový bod, nejdříve vytvoříme množinu bodů okolo tohoto bodu pomocí vzorkovacího schématu. Zde se může jednat například o množinu soustředných kruhů. Poté je vypočítána kompenzace orientace, což zaručí invarianci vůči rotaci cílového obrazu. Třetím krokem je vybrání vzorkových párů z množiny vytvořené použitým vzorkovacím schématem. Počet vybraných párů poté určí délku výsledného řetězce. Páry bodů jsou proti sobě porovnány, a pokud hodnota intenzity větší v prvním bodě, zapíše se do výsledného řetězce „1“, v opačném případě poté „0“. Konkrétní implementace volby vzorkovacího schématu, způsobu výpočtu kompenzace orientace, délky výsledného řetězce a pořadí volby navzorkovaných párů budou záviset na konkrétním algoritmu.

Tímto byl shrnut princip binárních deskriptorů a nyní lze přejít k obecnému popisu algoritmu FREAK [**?**], celým názvem Fast RetinA Keywords, který je jedním z nejnovějších zástupců této skupiny. Snaží se co nejvíce napodobit tvorbu deskriptorů způsobem, který se objevuje v samotném lidském oku. Zvolené vzorkovací schéma je zobrazeno na 1.5. Jedná se o replikaci vzorkování, které napodobuje rozmístění receptorů na lidské sítnici. Schéma se skládá ze soustředných kruhů, kde hustota vzorkovacích bodů je největší kolem středu a exponenciálně se snižuje směrem k okrajům. Každý vzorkovací bod je vyhlazen Gaussovou funkcí, kde průměr každého kruhu

určuje standardní odchylku jádra.

Použitím výsledného vzorkovacího schématu vzniknou desítky vzorkovacích bodů, což odpovídá tisícům možných kombinací vzorkovacích párů, avšak mnoho těchto párů není dostatečně vypovídajících při popisu zkoumané oblasti. FREAK se v tomto ohledu inspiruje algoritmem ORB[1] a snaží se získávat páry s co možná největší odchylkou a upřednostňuje ty, jež spolu nesouvisí. Tento výběr vede v praxi k zajímavým výsledkům. Nejdříve jsou vybrány body ležící především na okrajích zkoumané oblasti, zatímco body v centru jsou vybrány mezi posledními, což odpovídá způsobu, kterým jsou zkoumány objekty v lidské sítnici. Zde se nejdříve využijí receptory v periferní oblasti pro odhad obecné polohy objektu a teprve poté je provedena podrobnější analýza ve středu oblasti, kde je nejvíce fotoreceptorů.

Pro kompenzaci orientace používá FREAK předem definovanou sadu 45 symetrických párů bodů, pro které jsou spočítány a porovnány gradienty, a tímto je spočtena případná rotace mezi objekty 1.6. Pro dodatečné zrychlení algoritmu je při porovnávání dvou binárních řetězců nejdříve porovnáno prvních 128 bitů. Pokud v tuto dobu rozdíl mezi řetězci větší než stanovená mez, kandidát je zahozen. V opačném případě se princip opakuje pro každých 128 bitů až do konce řetězců. Naměřené výsledky ukazují, že 90% kandidátů je zahozeno během prvních 128 bitů porovnávání, čímž je algoritmus podstatně zrychlen.

Kapitola 2

Výběr AR knihovny

V současné době existuje na trhu početná skupina knihoven určených pro použití v augmentované realitě. Jsou zde jak open-source knihovny, tak plně profesionální komerční řešení. Liší ve funkčnosti a samozřejmě také v ceně. V této kapitole učiníme souhrnné srovnání těch nejpodstatnějších zástupců.

2.1 ARToolkit

- Podporované platformy: Windows, Mac OS X, Linux, iOS, Android, Unity
- Licence: Open-Source, GNU Lesser GPL v3.0

ARToolkit je open-source knihovna, jejíž historie spadá až do roku 1999, kdy byla vydána první veřejná verze. Podporuje velké množství platforem a nabízí veškerou základní funkcionalitu, jakožto sledování pozice jedné či dvojice kamer, sledování rovinných obrazců a jakýchkoliv jiných čtvercových vzorů a je dostatečně rychlá pro rozšířenou realitu v reálném čase. Zásadní nevýhodou pro tento projekt je ovšem neschopnost knihovny sledovat pokročilejší objekty, jako například válcovité a kvádrovité tvary. Ačkoliv se jedná o open-source projekt, díky velké popularitě a množství projektů implementovaných v této knihovně je k nalezení mnoho návodů a diskuzních fór pro hledání případné pomoci.

2.2 EasyAR

- Podporované platformy: Windows, Mac OS X, iOS, Android, Unity

- Licence: Zdarma

EasyAR je menší a jednodušší knihovna v porovnání s ostatními. Přístup ke knihovně je volný, vývojář však spolu s užitím této knihovny musí souhlasit s licenční smlouvou, jenž je uvedena na webové stránce knihovny (nejedná se o žádnou „oficiální“ licenci). Zároveň s tímto je nutno se registrovat k používání, vygenerovat licenční klíč a tento poté ve výsledné aplikaci používat. Podporovány jsou všechny významné platformy a funkcionalita je podobná knihovně ARToolkit, avšak zde je nabízeno také rozpoznávání obrazu pomocí cloudové služby, jenž snižuje výkonovou náročnost aplikace výměnou za zvýšené nároky na přenos dat. Stejně jako v předchozí knihovně zde ale chybí možnost sledovat 3D objekty, avšak tato funkcionalita je dle informací plánována v následující verzi. Velkou nevýhodou je avšak dostupná dokumentace, respektive její absence. Společnost vyvíjející EasyAR je čínského původu a bohužel určité stránky podpory a dokumentace jsou pouze v čínštině, což poněkud znesnadňuje orientaci.

2.3 Vuforia

- Podporované platformy: Windows, iOS, Android, Unity, HoloLens, ODG, Epson Moverio

- Licence: Zdarma + komerční verze

Vuforia je zástupcem komerčních AR knihoven. Základní verze lze používat zadarmo, avšak pro nasazení aplikace je již potřeba zakoupit komerční licenci. Zde se je poté rozdílné, zda aplikace provádí výpočty lokálně v zařízení, či je rozpoznávání vykonáváno použitím cloudové služby. V prvním případě se jedná o jednorázový poplatek, v druhém o měsíční platby. Pro rozsáhlé projekty a podniky je zde poté nabízena individuální smlouva. Funkcionalitou se Vuforia zařazuje k tomu nejlepšímu, co lze nabídnout – podporováno je sledování jak planárních obrazů, tak základních 3D objektů jako jsou válcová a kvádrová tělesa, rozpoznávání textu (z vestavěného anglického slovníku, či vlastního dodaného), ale i vlastních 3D objektů naskenovaných pomocí speciálního pluginu. Dále jsou nabízeny funkce pro konstrukci virtuálního terénu z reálné scény, o který poté virtuální objekty mohou kolidovat a jinak s ním interagovat, a funkce pro rozšířené sledování cíle. To umožňuje sledování virtuální scény i v případě, že sledované kontrolní obrazce již nejsou v přímém záběru kamery. Jelikož se jedná o velký komerční projekt, je zde také výhoda ve větší podpoře v případě jakýchkoliv nejasností či problémů.

2.4 CraftAR

- Podporované platformy: Windows, iOS, Android, Cross-platform Cordova, Web, Unity

- Licence: Zdarma + komerční verze

Další z řady komerčních knihoven, CraftAR je přímým konkurentem knihovny Vuforia. Stejně jako v předchozím případě, je zde možnost vyvíjet bez poplatku, avšak s omezením na počet sledovaných vzorů a s přítomností vodoznaku na výsledném obraze. Rozdílem v business modelu zde je, že v základní verzi je veškeré rozpoznávání prováděno pomocí cloudové služby – včetně zkušební verze zdarma. Základní cenové sazby se poté odvíjí od počtu rozpoznaných cílů v obraze a množství provedených rozpoznávání za měsíc. Pokud by uživatel chtěl využívat rozpoznávání přímo v zařízení, toto je také možné, avšak za velmi strmý jednorázový poplatek. Nabízenými funkcemi CraftAR nijak nezaostává za konkurencí, nabízí veškeré funkce, které by vývojář mohl požadovat. Zároveň cílí i na méně zkušené uživatele a nabízí online dostupné prostředí pro tvorbu jednoduchých scén, které je poté možno přímo nasazovat. Další nabázenou službou je poté možnost využití jednoduché webové služby přímo od CraftAR, pro spravování a dodávání jednoduchého obsahu pro webové projekty.

2.5 Wikitude

- Podporované platformy: Windows, iOS, Android, Unity, Cordova, Titanium, Xamarin

- Licence: Zdarma + komerční verze.

Poslední ze vybraného seznamu komerčních knihoven pro rozšířenou realitu je Wikitude. Jedná se o plnohodnotnou knihovnu na stejné úrovni jako její konkurenti. Podmínkou bezplatné licence je registrace a zároveň přítomnost vodoznaku ve výsledném obraze. Přechod na plnohodnotnou verzi je poté zpoplatněn a to buď jednorázovým poplatkem, či ročním předplatným. Nutno dodat, že cenově vychází Wikitude nejhůře ze všech testovaných knihoven. Zarážející je i fakt, že i při přechodu na placenou verzi není v ceně jakákoliv podpora kromě veřejných diskuzních fór.

Název knihovny	Použitý algoritmus
ARToolkit	FREAK
EasyAR	SURF
CraftAR	SIFT
Vuforia	SIFT
Wikitude	ORB, BRIEF

Tabulka 2.1: Pravděpodobné použitých rozpoznávacích algoritmů AR knihovnami

2.6 Srovnání použitých algoritmů

Každá z těchto knihoven používá jeden či více konkrétních implementací algoritmů pro detekci a sledování příznaků v obraze. V rámci porovnání těchto knihoven byl učiněn krátký výzkum, které knihovny používají jaké algoritmy. Z pochopitelných důvodů ne všechny (především komerční) knihovny tyto informace veřejně poskytují, tudíž v těchto případech byly na základě jednoduchých testů jednotlivé algoritmy odhadnuty. V tabulce 2.1 lze vidět výsledný odhad dat. K rozhodování byly použity i naměřená data srovnávající rychlost a přesnost daných algoritmů.[7] Pouze open-souce knihovna ARToolkit veřejně deklaruje použitý algoritmus, čímž je metoda FREAK [2] používající binární deskriptory. EasyAR tuto informaci nenabízí, avšak dle analýzy návrhu systému a výsledků sledování vzorů je pravděpodobně využit algoritmus SURF, který nabízí sledování velkého množství cílů a zároveň je invariantní vůči změně měřítka a velmi odolný vůči změnám osvětlení. Knihovny soustředící se na podporu co největšího počtu sledovaných cílů se zpracováním přímo v cílovém zařízení jako například Wikitude potřebují rychlé zpracování, nejlépe s využitím binárních deskriptorů jako například ORB a BRIEF. Vuforia poté sice nabízí zpracování přímo v mobilním zařízení, avšak je zde nastaven poměrně nízký limit na počet současně sledovaných cílů ve scéně.

2.7 Shrnutí a výběr

Na první pohled mají samozřejmě navrh knihovny EasyAR a ARToolkit, jelikož jejich stav open-source projektů je žádaný z licenčního hlediska. Bohužel ale ani jeden tento projekt nepodporuje v dostatečné míře sledování základních 3D objektů. EasyAR má navíc špatnou dokumentaci a relativně malou komunitu. Z tohoto důvodu je vůběr zůžen na komerční knihovny.

Zde bylo již rozhodování těžší, avšak ve výsledku byla zvolena knihovna Vuforia. Hlavními důvody jsou dobrá integrace do množství platforem, možnost počítat sledování objektů přímo v mobilním zařízení a v případě spokojenosti a přechodu na placenou verzi i příjemné cenové podmínky. Hlavními nevýhodami konkurentů zde bylo v případě CraftAR neustálá nutnost spojení s cloudovou službou a v případě Wikitude nedostatečná podpora i při přechodu na placenou verzi. V obou případech jsou poté i mnohem vyšší cenové náklady pro nasazení aplikace.

Kapitola 3

Výběr mobilní platformy

V současné době existuje mnoho způsobů, jak vyvíjet pro mobilní zařízení. Kromě klasických nativních SDK existuje i řada multiplatformních frameworků pro tvorbu těchto aplikací. Tento projekt je cílen především na platformu Android, avšak samozřejmě je bonusem, pokud by byla aplikace bez větších úprav použitelná i na ostatní operační systémy. V seznamu níže krátce představíme tři hlavní způsoby vývoje:

3.1 Nativní vývoj pro OS Android

Jak již název napovídá, nativní vývoj se soustředí na jednu konkrétní platformu. Výhodou je především na míru vyvíjená aplikace, která je řádně otestována v rámci daného OS a veškerý kód je psán přímo pro danou platformu. V tomto případě se jedná v vývoj v prostředí Android Studio, což je prostředí je přímo navržené pro vývoj aplikací pro OS Android a nabízí veškeré funkce k tomu nutné. Nevýhodou poté je nutnost vývoje stejné aplikace pro ostatní platformy, často v jiném programovacím jazyce, tudíž s nemožností znovupoužitelosti kódu. V případě dodržení zásad softwarového inženýrství je sice vývoj urychlen již existujícím návrhem architektury, vzhledu uživatelského rozhraní atd., avšak stále je zde časová náročnost mnohem vyšší.

3.2 Unity Engine

Unity je mutiplatformní herní engine, primárně využívaný k vývoji her a simulací pro PC, herní konzole a mobilní platformy. Založen byl v roce 2005 pouze pro OS X, nyní je dostupný pro 27 různých platforem. Unity

Engine je založen na přenositelnosti a podporuje množství grafických API pro všechny hlavní platformy. Zároveň podporuje množství grafických standardů, jakožto například kompresi textur, nastavení rozlišení pro každou podporovanou platformu, normálové, odrazové, parallax mapování, SSAO a další. Zároveň dokáze zvolit ideální shader pro každou hardwarovou výbavu cílového stroje, popřípadě zvolit záložní nastavení, jenž upřednostní co možná nejlepší výkon. Unity nabízí 4 stupňující se typy licence. Základní osobní licence je zdarma a již v této verzi nabízí veškerou funkcionalitu celého enginu. Nedostupné jsou pouze pokročilé služby jako ladění výkonu, přizpůsobení „splash" obrazovky při startu aplikace, přístup k zdrojovému kódu enginu a prémiová podpora. Pro vývoj projektů je k dispozici samostatný Unity Editor, který spravuje základní strukturu projektu. Pro tvobru skriptů je zde poté možnost využít externí editor, jako například Miscrosot Visual Studio.

3.3 Xamarin

Další ze zástupců multiplatformního vývoje, Xamarin byl založen v roce 2011 vývojáři stojícími za vznikem Mono, což je multiplatformní implementace CLI a CLS, známé jako Microsoft .NET. Xamarin umožňuje vývoj nativních aplikací pro systémy Android, iOS a Windows Mobile. Pro samotný vývoj lze použít samostné Xamarin Studio, či v poslední době více doporučovanou možnost pluginu pro Microsoft Visual Studio. Veškerý sdílený kód je psán v jazyce C#, kdy ačkoliv Xamarin umožňuje přímý vývoj pouze pro mobilní platformy, tato sdílená část kódu je znovupoužitelná i na ostatních platormách, jako například Windows či macOS. Framework dále nabízí i využití tzv. Xamarin.Forms, které umožňují použití sdíleného kódu také na tvorbu ovládání výsledné aplikace napříč platformami.

3.4 Porovnání a výběr

Ačkoliv je cíleno především na OS Android, je v tomto projektu velkou výhodou mít aplikaci připravenou i pro ostatní platformy. Zde se nemusí jednat pouze o ostatní mobilní platformy jako iOS a Windows Mobile, avšak bylo by přínosem mít projekt připraven k nasazení i v rámci webové služby, jako součást hlavního eshopu pomocí technologie WebGL. Dalším kritériem výběru je zde snadnost vývoje a integrace rozšířené reality do aplikace. V tomto ohledu ztrácí multiplatformní framework Xamarin, který je zdaleka nejméně podporován současnými AR knihovnami. Nejlepší vývoj zde nabízí

Unity Engine, který podporuje většina AR knihoven a který nabízí rychlé a jednoduché testování projektu pomocí možnosti spuštění projektu přímo v editoru a změny konfigurace za běhu. V omezené míře toto nabízí i prostředí Android Studio, avšak zde je třeba testovat přímo na mobilním telefonu, či využít možnosti emulátoru, který avšak zdaleka tak výkonný, obzvláště na náročné aplikace tohoto typu. Z důvodů rozsáhlé podpory AR knihovnami, přehledném testování během vývoje, možnosti rychlého nasazení na velké množství platforem a rozsáhlé dokumentaci byl pro tento projekt ve výsledku zvolen Unity Engine.

Kapitola 4

Návrh systému

4.1 Analýza existujících dat

Jeden z hlavních požadavků na tento projekt je co největší využití již existujících dat, především fotografie již nafocených hodinek z otočného stolku, a vývoj aplikace, která tato data umí v rozumné míře použít. Prvním krokem je tedy analýza existujícího internetového obchodu, rozbor prezentovaných dat a diskuze o znovupoužití 360°fotografií.

4.1.1 Informace z internetového obchodu

Existující eshop[20] nabízí velké množství zboží a kromě náramkových hodinek také různé druhy šperků a příslušenství. Tato data nejsou pro tento projekt podstatná a nebudou tedy zahrnuta do analýzy a následné implementace. Samotné hodinky se dělí do těchto základních kategorií:

- Značky A-Z
- Pánské hodinky
- Dámské hodinky
- Dětské hodinky
- Dle typu
- VÝPRODEJ hodinek

Každá z těchto dalších kategorií má poté na výběr navíc seznam podkategorií, které se liší dle základní kategorie – ve všech případech kromě jedné se jedná o třídění dle typu a určení daných hodinek, například:

- Analogové
- Barevné
- Digitální
- Keramické
- Levné
- Luxusní
- atd..

Pouze v případě kategorie „Značky A-Z“ je zde očekávaný seznam výrobců hodinek, dle kterého se dá dále třídit.

Nutno dále dodat, že nelze podkategorie kombinovat, například nelze zvolit „Pánské“, „Sportovní“ a „Digitální“ hodinky z hlavní nabídky kategorií. K tomuto slouží sloupec pro podrobné filtrování na levé straně stránky, kde lze specifikovat svoji volbu pro jakoukoliv kombinaci parametrů. Dále je zde možnost řadit výsledné hodinky dle ceny, popularity či data přidání do internetového obchodu a seznam je samozřejmě stránkován.

4.1.2 Detail produktu

Při zobrazení detailu každého produktu zde nalezneme podrobné informace o každých hodinkách a kromě obecných informací o možném financování a například možností přikoupení záruky nás zajímají hlavně tyto informace:

- Obrázek hodinek zepředu ve vysoké kvalitě
- 3D prezentace hodinek pomocí série snímků pořízených na otočném podstavci
- Cena
- Podrobný popis
- Seznam vlastností hodinek s jejich hodnotami

Všechny tyto informace jsou použitelné do mobilní aplikace, kde obrázek hodinek z přední strany může sloužit jako náhled v seznamu všech modelů a ostatní data plnit stejnou roli jako zde, tudíž detailní informace pro prezentaci daného modelu

Obrázek 4.1: Existující fotografie hodinek pořízené pomocí otočného podstavce.

4.1.3 Analýza pořízených fotografií

Pro každé hodinky byla pořízena série fotografií vyobrazující dané hodinky v rozsahu 360°. Toho bylo docíleno použitím otočného podstavce a fotoaparátu na stativu, jenž zajistil zobrazení hodinek ve stejném místě v obraze, díky čemuž je možno je bez úprav zobrazit jako prezentaci. Kvalita fotografií je na dobré úrovni co se rozlišení a celkové ostrosti obrazu týče, avšak pro použití pro 3D model je situace horší.

První problém je v pozadí a v materiálu jistého typu hodinek. Zatímco některé tmavé hodinky z matného materiálu vynikají proti bílému pozadí dostatečně, je zde mnoho stříbrných či zlatých hodinek, které nejen že již svojí barvou splývají s pozadím, ale negativní roli zde hraje i nasvícení. Aby bylo dosaženo dobrého vzhledu a zářivosti materiálu na výsledných fotografiích, nachází se u těchto lesklých hodinek mnoho odlesků, které ještě více pomáhají se splynutím s okolím. Dalším problémem je poté úhel fotografování. Ačkoliv jsou pořízené fotografie do 360°okolo hodinek, jsou pořízeny pouze v jedné rotační ose, tudíž spodní část hodinek (řemínku) je zakrytá a tento povrch se tedy musí rekonstruovat.

Posledním potenciálním problémem je poté úhel fotografování, jelikož hodinky nejsou foceny „vodorovně“ s podložkou. Výsledkem je záběr lehce svrchu, což opět nenahrává jednoduché extrakci informací z fotografie, jelikož je povrch zkreslený.

Vzorové fotografie v současném stavu budou moci být použity, avšak kvalitou budou v některých ohledech zaostávat, zejména u lesklých modelů, kde se ztratí informace o původní barvě. Tato fotografie, pokud se namapuje na nějaký 3D model, bude mít na daném místě konstantní odlesk, který bude působit nepřirozeně, jelikož se nebude měnit s úhlem pozorování.

4.2 Vymezení funkčnosti aplikace

Než přejdeme k samotné implementaci, je nutno vymezit přesnou funkcionalitu celého systému. Základním předpokladem pro tento projekt je, že výsledná mobilní aplikace se nesnaží plně zastoupit webový obchod – ačkoliv by bylo možné tuto funkcionalitu do aplikace přidat, pro pozitivní výsledek by musela být kvalita výsledných 3D modelů velmi vysoká, aby měl zákazník iniciativu hodinky přímo z mobilní aplikace zakoupit. V opačném případě by hrozil opačný efekt – zákazník vidí jen přibližný vzhled hodinek a pokud bude tlačen ke koupi, radši od aplikace odstoupí. Primární cíl aplikace bude sloužit jako technologická prezentace, ukázka všech možností a „navnazení“ zákazníka na větší zájem o daný produkt a zároveň zlepšení povědomí o daném obchodníkovi.

4.2.1 Funkcionalita mobilní aplikace

Mobilní aplikace v tomto případě poskytuje velmi zjednodušenou verzi hlavního obchodu. Uživatel může filtrovat výsledky dle kategorie a podkategorie, řadit výsledky dle daných parametrů a zobrazit detailní informace o každém modelu a následně si daný model virtuálně zkusit na vlastní ruce. Obsahuje také odkaz na přejití do plnohodnotného internetového obchodu, kde již zákazník bude možnost zboží zakoupit. Níže jsou shrnuty základní funkční a nefunkční požadavky na mobilní aplikaci:

4.2.1.1 Funkční požadavky

F1. Výběr kategorie hodinek

- **F1.1.** Výběr podkategorie hodinek

F2. Řazení seznamu hodinek

- **F2.1.** Dle ceny
- **F2.2.** Dle popularity
- **F2.3.** Dle data přidání

F3. Zobrazení seznamu hodinek

- **F3.1.** Zobrazení detailních informací
- **F3.2.** Zobrazení augmentovaného 3D modelu v reálné scéně
- **F3.3.** Možnost přejít do hlavního obchodu

F4. Nastavení šířky pásku

F5. Zobrazení instrukcí ohledně použití aplikace

4.2.1.2 Nefunkční požadavky

- Mobilní telefon s operačním systémem Android, alespoň verze 5.0
- Dostupné internetové připojení
- Dotykový display s rozlišením alespoň 800x480px
- Zadní kamera s rozlišením alespoň 5MPx

4.3 Společné části systému

Než bude možné implementovat jakoukoliv konkrétní část systému, je nutno rozhodnout o společných částech tak, aby bylo v pozdějším vývoji přesně definované API mezi mobilní aplikací a webovou službou.

4.3.1 Zobrazení virtuálních hodinek

Možných řešení, jak hodinky reprezentovat na ruce uživateli je více, ačkoliv se všechna pohybují na ose s nepřímou úměrou kvalita/kvantita. Nejkvalitnějším způsobem je reprezentace vlastním 3D modelem pro každé hodinky. Každý produkt se individuálně vymodeluje, otexturuje a poté zobrazí uživateli unikátní pro každý model hodinek. Zde by se poté muselo také rozhodnout, zda lze použít již hotové fotografie, nebo nafotit nové. Toto by samozřejmě zabralo velké množství času, jelikož každý kompletní model by mohl zabrat i hodiny času a pro realizaci až tisíců hodinek by toto bylo velmi časově/finančně náročné.

Naopak nejrychlejší variantou je použít pouze jeden univerzální model, na který poté bude nanášena jiná textura pro každé hodinky. Toto řešení se samozřejmě neobejde bez kompromisů, jelikož existuje několik zásadně rozličných typů hodinek, které by nevypadaly věrohodně, pokud by se jejich textury nanesly na nějaký univerzální model. Kompromisem je zvolit a vytvořit malé množství modelů, které budou sloužit jako předlohy určitým typům hodinek. Vytváření nových 3d hodinek si poté obsluha vybere, který z nabídky modelů použít a na základě tohoto poté dodat vhodnou texturu.

Po prostudování existující databáze hodinek bylo pro účel tohoto projekto rozhodnuto o vytvoření třech univerzálních modelů,které budou sloužit jakožto podklad pro zobrazování hodinek. Konkrétně se jedná o typ dám-

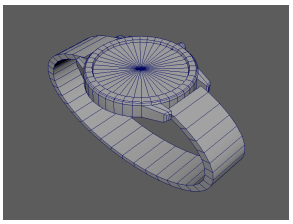
ských hodinek, pánských hodinek a sportovních hodinek. Tyto se zásadně liší ve svém provedení, velikosti náramku a ciferníku a jsou proto vhodné jako základní prototypy pro reprezentaci většiny modelů hodinek. Jakožto rozšíření funkcionality je poté možné 3D modely přidávat dynamicky do webové služby, kde mobilní aplikace poté dodatečné modely stáhne za svého běhu.

4.3.2 Návrh struktury dat

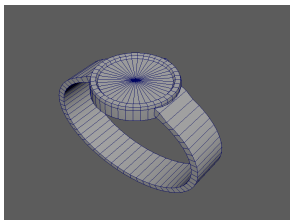
Na základě výše stanovených požadavků na mobilní aplikaci a předchozí analýzy existujících dat je nyní možno navrhnout základní datový objekt pro reprezentaci každého modelu hodinek. Každý objekt hodinek bude obsahovat minimálně základní informace

- Výrobce
- Model
- Podrobný popis
- Seznam vlastností hodinek
- Seznam kategorií produktu
- Seznam podkategorií produktu
- Odkaz na do Eshopu na tento produkt
- Cena
- Popularita
- Datum přidání
- 3D model
- Textura pro zobrazení na modelu
- Náhledový obrázek

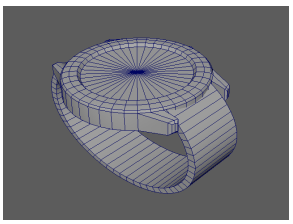
Kromě základních údajů o hodinkách (výrobce, model, popis, seznam vlastností, odkaz do hlavního obchodu a cena) je potřeba pro dosažení požadované funkčnosti uchovat i další data. Seznam kategorií a podkategorií je použit pro filtrování v seznamu hodinek, zatímco popularita a datum přidání hodinek jsou atributy nutné pro implementaci stejného řazení, jako



Obrázek 4.2: Pánské hodinky



Obrázek 4.3: Dámské hodinky



Obrázek 4.4: Sportovní hodinky

nabízí internetový obchod. 3D model je konkrétní model hodinek, který se zobrazuje na uživatelově ruce společně s přiloženou texturou a náhledový obrázek slouží pro zobrazení vzhledu hodinek v seznamu produktů.

4.3.3 Tvorba 3D modelů

Při návrhu modelu pro náramkové hodinky je nutno rozhodnout o detailnosti výsledku, na který bude poté nanášena textura. V rozhodování platí přímá úměra, kde detailnější model se rovná nejenom více práce se samotným modelováním, avšak také s následným texturováním. V případě tvorby unikátního modelu pro každý model hodinek by toto nebylo velkým problémem, avšak při použití několik univerzálních modelů pro všechny hodinky by se časová náročnost texturování hodinek zvýší řádově. Dalším problémem je poté použití již existujících fotografií, které jsou dostačující pro zobrazení na jednodušším modelu, avšak pro detailnější model by mapování fotografií na texturu bylo mnohem složitější z důvodu nutnosti komplexnějších UV map.

Pro tři základní typy hodinek bylo rozhodnuto o vytvoření relativně jednoduchých modelů, především z důvodu znovupoužitelnosti již pořízených fotografií. Výsledné modely jsou vidět v pánské variantě na 4.2, dámský model viz 4.3 a sportovní hodinky viz 4.4. Ve všech případech model skládá z dvou základních částí – tělo hodinek a pásek.

Pásek je vytvořen jakožto jednoduchý válec s výsekem na místě, kde se nalézá tělo hodinek. V původním návrhu zde byla na dolní straně vymodelována i spona pásku, avšak od tohoto rozhodnutí bylo ustoupeno, jelikož výsledné, otexturované objekty nevypadaly přesvědčivě. Tělo hodinek je tvořeno modifikací válcového primitiva, kde šířka okraje okolo ciferníku a celého těla jsou dány typem hodinek. Dále je zde vymodelováno jednoduché otočné nastavení hodinek a u pánských a sportovních hodinek také výstupy pro uchopení pásku.

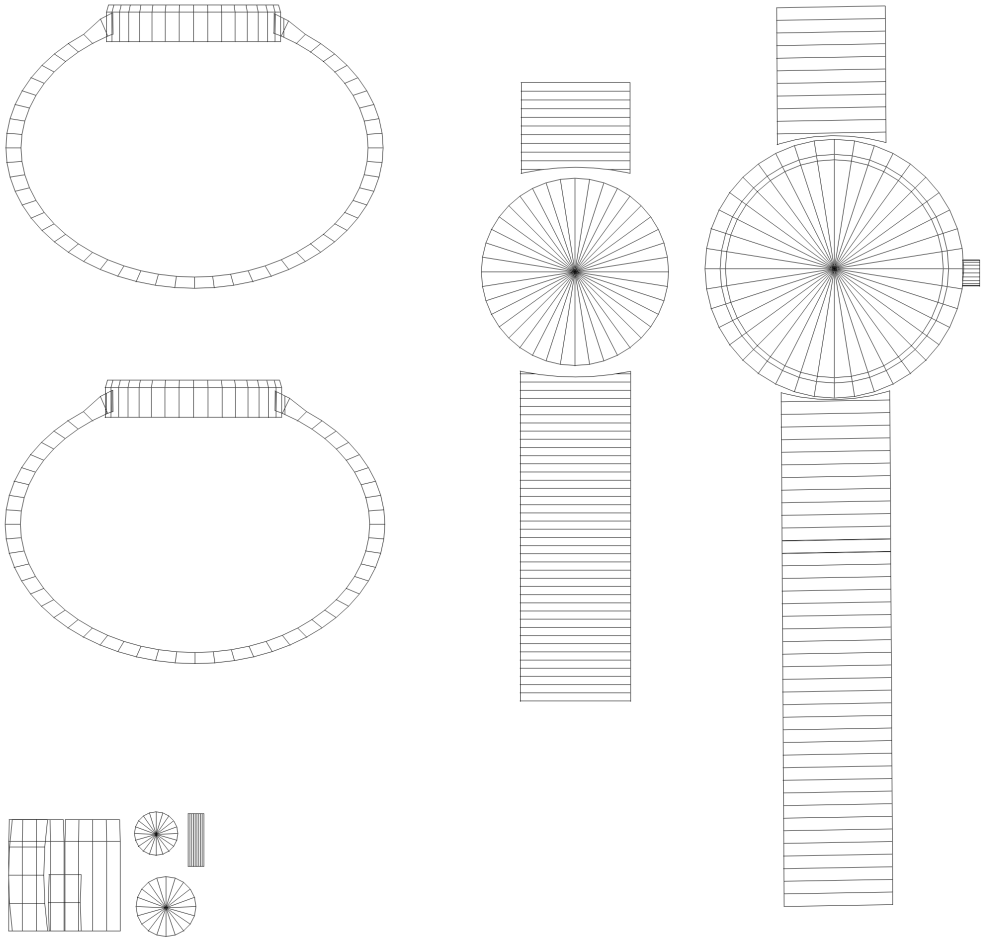
4.3.4 Tvorba UV map

Hlavním požadavkem na vkládání nových modelů hodinek do databáze je rychlost a znovupoužitelnost pořízených dat, čemuž se přizpůsobila i tvorba UV souřadné mapy pro každý model. Předlohou zde byly 4 konkrétní fotografie z každé série 360°fotografií a to ze všech čtyř základních úhlů – z přední strany, ze zadní strany a z každého boku. Na základě tohoto byla provedena série planárních projekcí pro každý model a vytvořeny mapy pro základní typy hodinek (na 4.5 lze vidět navrženou mapu pro dámské hodinky). Na těchto mapách jsou na levé straně projekce ze stran, na pravé straně je projekce vnějšího povrchu hodinek a mezi nimi poté vnitřní povrch hodinek. V dolním levém rohu jsou poté seskupeny plošky, které nejsou přímo viditelné. Tyto plochy a vnitřní strana jsou uzpůsobeny pro otexturování obecnou barvou daných hodinek, jelikož tyto data nejsou přímo dostupné z pořízených fotografií. Jelikož na fotografiích chybí také část informace o pásku hodinek (neexistuje fotografie shora a zezdola), je pro kompletní texturu nutno část pásku zrekonstruovat. Na 4.6 lze vidět hotovou a připravenou texturu pro hodinky dámského typu.

4.3.5 Návrh komunikace s webovou službou

Mobilní aplikace musí čerpat vždy aktuální data z webové služby, je proto nutné stanovit alespoň základní typ komunikace mezi aplikací a webovým serverem. Pro největší efektivitu bude komunikace probíhat pomocí REST příkazů. Následující požadavky jsou obecnými předpoklady pro komunikaci:

- Styl požadavků splňuje požadavky RESTful
- Formát odpovědi je strukturovaný JSON dokument
- Odpověď kromě samotných dat obsahuje také označení stavu – zda požadavek proběhl v pořádku
- Komunikace je alespoň základním způsobem zabezpečena
- Podpora minimálně následujících typů dotazů
- Stáhnout informace o kategoriích
- Stáhnout seznam hodinek
- Stáhnout dodatečné soubory (textury, náhledový obrázek, 3D model)



Obrázek 4.5: UV texturovací mapa pro dámské hodinky

Obrázek 4.6: Výsledná textura pro dámské hodinky dle navržené UV mapy

Kapitola 5

Webová služba

5.1 Výběr frameworku a technologie

Webová služba pro použití v tomto projektu musí splňovat následující kritéria:

- Využití SQL databáze pro vkládání a administraci vložených dat produktů
- Jednoduché uživatelské rozhraní pro správu těchto dat
- Snadná rozšiřitelnost
- Efektivní a rychlé poskytování dat mobilní aplikaci s využitím REST protokolu
- Možnost nasazení na jakýkoliv webový server

Tyto požadavky nejsou nijak náročné a splňuje je většina dnešních frameworků. Jakožto možní kandidáti byli vybráni tito zástupci PHP frameworků: [8]

- Yii
- Symfony2
- Zend 2
- Nette

Po zvážení požadavků a dosavadních zkušeností s těmito frameworky bylo rozhodnuto pro použití Symfony2 jakožto základ pro server. Tento framework obsahuje již v základu ORM Doctrine pro mapování objektů a práci s SQL databází a je již odzkoušen jakožto kvalitní a jednoduchý základ pro REST server [11].

5.2 Popis zvoleného frameworku a struktury webové služby

Symfony2 je full-stack PHP framework pro webové aplikace. Základní použitá architektura je typu MVC, neboli Model-View-Controller, což je třívrstvá architektura založena na následujícím principu:

Model – datová vrstva poskytující data a zajišťující jejich zpětnou persistenci View – prezentační vrstva, skládající se z HTML šablon naplněných daty Controller – logická vrstva aplikace, všechny příchozí požadavky procházejí skrz Controller, který se dotazuje Modelu na případná data a jakožto odpověď vrací View objekt.

5.2.1 Struktura projektu

Celý projekt se skládá z několika hlavních komponent a typů tříd. Níže jsou tyto základní stavební kameny popsány do většího detailu.

5.2.1.1 Bundles

Bundles je v Symfony frameworku označení přenositelné části projektu, pluginu, jenž se dá nezávisle přenášet a případně sdílet mezi více projekty. V Symfony je Bundlem i jádro frameworku, vše se zkládá z těchto částí. Obecně každý Bundle plní nějakou základní funkcionalitu a dá se to projektu snadno integrovat. Součástí každého Bundle jsou všechny soubory, jenž jsou třeba pro jeho správnou funkčnost, včetně PHP souborů, HTML šablon, stylovacích souborů, testů a mnoho dalšího.

5.2.1.2 Základní nastavení aplikace

Pro základní nastavení projektu slouží v Symfony adresář „/app/config“, který obsahuje zdroje sloužící ke konfiguraci, routování a bezpečnosti aplikace.

Routing.yml Soubor obsahující globální instrukce pro směrování přicházejících požadavků. Zde jsou především definovány cesty do jednotlivých Bundles a Controllerů, stejně tak jako jiné ostatní cesty (například přihlášení a odhlášení uživatele). Je možno zde definovat také cesty pro konkrétní akce v Controllerech, avšak od tohoto přístupu se upouští, vzhledem k snadnějšímu použití anotací přímo v daném Controlleru.

Security.yml Zde se nalézají veškeré informace o přístupu do aplikace. Lze zde nalézt způsob šifrování uživatelských hesel, popis a hierarchii možných uživatelských rolí, zdroj dat o uživatelích pro přihlašování (lze buď poukázat na databázi, nebo vložit konkrétní údaje přímo do kódu) a poté hlavně omezení přístupu pro nepřihlášené uživatele a způsob autentifikace uživatelů pro přístup do konkrétních oblastí webu.

Services.yml Definice aplikačních Services se nalézají právě zde. Jedná se o třídy, jenž jsou po svém definování přístupné odkudkoliv z aplikace, tudíž jsou vhodné pro funkcionalitu, jenž není vázána na konkrétní situaci. Definice se skládá z názvu nové Service, odkaz na třídu, jenž má být použita a také seznam vstupních argumentů, jenž budou předány při konstrukci konkrétní Service.

Parameters.yml Zde se nachází definice systémových parametrů-konstant, jenž jsou použity například pro spojení s SQL databází – konkrétně parametry, jenž jsou stejné pro celou aplikaci, ale rozdílné pro nasazení na cíl (fyzický server).

Config.yml Hlavní třída nastavení celého serveru, zde se importují veškeré ostatní konfigurační soubory a nastavují se zde hlavní parametry celé aplikace, jakožto například volba konkrétního šablonovacího enginu, validace, správa sessions, nastavení připojení k SQL databázi a e-mailového rozšíření.

5.2.1.3 Resources

Tato část webového serveru obsahuje veškeré šablony pro použití pro zobrazení uživateli. Standardní šablonovací systém dodávaný s frameworkem je Twig, jenž označuje výsledné html šablony koncovkou „.twig“. V tomto systému je možná prakticky jakákoliv základní manipulace s daty, jenž jsou šabloně předány, stejně tak jako všechny základní konstrukce programovacích jazyků, jako jsou například „for“ cykly, podmíněné bloky, formátování textu atd.

Následující zdroje patří již vždy do konkrétního Bundle, nejsou tedy ve společném umístění pro celý projekt a dají se tedy v případě potřeby sdílet například do jiných projektů.

Controller Controllery jsou prvky aplikace, jenž vyřizují požadavky od uživatelů a vrací zpět odpověď ve formě HTML vygenerovaných z dostupných šablon. Každý Controller má definovány „akce“, jenž dokáže obsloužit. Tyto jsou popsány danou funkcí a zároveň k nim musí být definována URL cesta – toho může být docíleno buď globálně v souboru routing.yml, či lokálně pomocí anotací u konkrétní funkce pomocí „@Route”, ve které se definuje URL vzor, při kterém se má tato akce vykonat a také dobrovolně název cesty, pro její případnou referenci v jiném kontextu. Do každé akce lze z URL a query parametrů předat proměnné do této akce, kde pokud jako vstupní parametr akce je definována konkrétní entita, Symfony se pokusí vzít korespondující parametr z URL za předpokladu, že se jedná o ID objektu a najít v databázi objekt požadovaného typu s daným ID. Po provedení akce je poté vyžadováno vrácení nějaké šablony pro zobrazení uživateli.

Entity Zde se nacházejí objekty pro reprezentaci samotných entit v projektu. Ačkoliv se může jednat i o obyčejné objekty bez jakýchkoliv vazeb, hlavní určení je pro objekty využívající ORM pro ukládání do použité databáze. K definici těchto vazeb lze použít řadu možných přístupů, avšak nejvyužívanější v současné době je použití anotací přímo k daným atributům objektu. Lze zde definovat jak základní sloupce v databázi, tak i cizí klíče a jakékoliv jiné vazby a omezení použité v definici databázových tabulek. Dále je zde možno využít také různých „lifecycle“ callbacků, například „prePersist“ a „postPersist“ pro úpravu dat před vložením do databáze – například při nahrání souboru se tento soubor před persistencí do databáze uloží na webový server a do databáze se uloží pouze jeho název a případně celá cesta k němu.

Form Formuláře se v Symfony2 frameworku definují do vlastních tříd s koncovkou „Type“, které jsou poté uloženy do tohoto adresáře. V definici každého formuláře se poté nachází přesný popis, jaká pole a případně jakou entitu použít jakožto kontejner pro data z tohoto formuláře. Při použití defaultních názvů atributů dané entity Symfony správně atribut přiřadí a vygeneruje popisek, avšak lze definovat i nová pole a například vložit i vlastní SQL dotaz pro filtrování dostupných možností.

Repository Repositáře jsou třídy používané pro získávání dat z použité databáze. Každá entita má standardně vlastní prázdný repozitář, který avšak dědí všechny základní příkazy z generického Repository. Zde je poté možno psát vlastní SQL příkazy pro rychlé vyhledávání dle parametrů, které se mohou opakovat, separovat logiku a nepoužívat tedy zbytečně místo v Controllerech a jiných místech. Jedna částečná výhoda i nevýhoda Doctrine je v případě definice funkce s názvem „FindBy{název_atributu}($atribute)“, kde Doctrine použije „magic“ doplnění funkce a automaticky vyhledá entity dle názvu atributu v názvu funkce, tudíž tyto klíčová slova nelze použít pro vlastní dotazové funkce.

5.3 Implementace

Dle požadavků by se mělo jednat o jednoduchý webový server, jenž bude poskytovat data o hodinkách mobilní aplikaci a tímto simulovat reálný internetový obchod.

5.3.1 Správa uživatelů

Základní částí implementace byla samotná kostra aplikace. V tomto případě se jednalo o vytvoření jednoduchého systému pro přihlašování/správu uživatelů, kteří poté mohou editovat data. Pro tento účel bylo vytvořeno samostatné UserBundle, jenž obsahuje pouze jednu entitu User. Tato entita obsahuje základní data o novém uživateli, jakožto uživatelské jméno, heslo, email a datum registrace. Dále zde byla implementována logika pro přihlašování a odhlašování. V globálním nastavení aplikace byly poté vymezeny jednotlivé role uživatelů a jejich pravomoce:

- ROLE_USER – standardní uživatel, smí data prohlížet, ale nesmí nic editovat či mazat
- ROLE_ADMIN – správce systému, který má všechna práva, smí tedy i data upravovat/mazat

Pro uživatele byly poté vytvořeny odpovídající webové stránky pro jejich správu a byl nastaven základní CSS styl aplikace.

5.3.2 Hlavní server

Pro hlavní část serveru byla založena nová Bundle s názvem AppBundle. Zde je poté umístěna veškerá další funkčnost.

5.3.2.1 Návrh entit

Pro hlavní část serveru bylo nejdříve nutné navrhnout datové objekty pro reprezentaci hodinek a splnění všech požadavků na systém. Níže následuje seznam veškerých entit, jejich popis, důvod implementace a funkčnost.

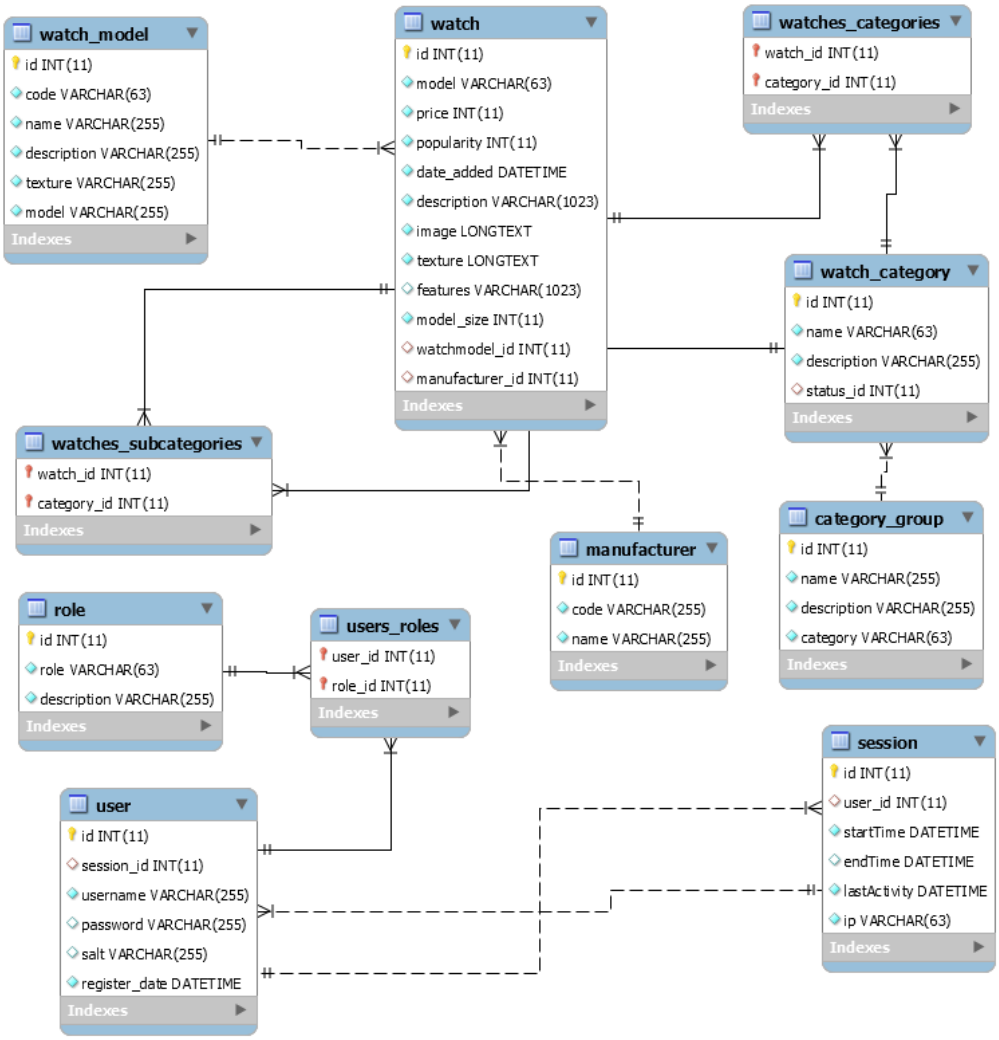
WatchCategory Pro reprezentaci kategorie hodinek je zde třída WatchCategory. Uživatel nejdříve na serveru vyplní seznam těchto kategorií, aby je poté mohl přiřazovat daným hodinkám. Nutno dodat, že tato třída plní funkci jak základních kategorií („Pánské“,“Dámské“,...), tak podkategorií/typů/určení hodinek, například Sportovní. Digitální, Analogové. Pro rozlišení o jaký typ kategorie jde, je zde atribut category, jenž speficikuje daný typ zařazení.

CategoryGroup Pomocná třída sloužící k detailní specifikaci dané kategorie, avšak může být použita i na jiné univerzální účely v budoucnu.

Manufacturer Jeden z prvotních návrhů pracoval s výrobcem hodinek pouze jako s obyčejným textovým atributem entity Watch. Jelikož ale jednou z vlastností mobilní aplikace je filtrování dle výrobce, tvorba separátní třídy toto řešení dělá robustnější díky absenci možných překlepů či rozlišování mezi výrobcem napsaným kapitálkami a druhým s normálními znaky.

WatchModel Pro reprezentaci 3D modelů použitých v mobilní aplikaci je zde třída WatchModel. Zde uživatel deklaruje jak její základní informace, tak i nahraje 3D model pro zobrazení v mobilní aplikaci. Zároveň pro jednoduchost je uživatel požádán také o vzorový obraz s UV pro pozdější tvorbu jednotlivých textur pro konkrétní hodinky.

Watch Hlavní třída pro uchovávání informací o uložených hodinkách je třída Watch. Zde jsou deklarovány všechny atributy pro použití v mobilní aplikaci. Kromě základních atributů jakožto model, výrobce, cena a popis jsou zde i atributy použité čistě pro řazení v seznamu hodinek u uživatele na mobilním zařízení, tedy datum přidání a popularita – ta je v reálném scénáři počítána komplikovaným způsobem založeném na kombinaci shlédnutí položky a počtu nákupů. Zde, jelikož žádná takováto metrika není k dispozici, je vstupem pouze celé číslo určující výslednou hodnotu. Dále zde uložené objekty obsahují i popis kategorií, ve kterých se dané hodinky nachází – jak kategorie, tak podkategorie jsou implementovány vztahem „Many to



Obrázek 5.1: Entity-relationship model navržené SQL databáze

Many“, tudíž jedny hodinky mohou patřit do více kategorií i podkategorií. Důležitými parametry jsou zde atributy vztahující se k použití na 3D modelu. Jedná výběr, na jaký model se hodinky mají zobrazit (z modelu dostupných a vložených na tento server), odkaz na soubor textury, která se na tento model má zobrazit a parametr určující korekci velikosti výsledného modelu (v případě, že by některé hodinky byly výrazně menší nebo větší).

5.3.3 Návrh uživatelského rozhraní

Uživatelské rozhraní této služby je navrženo jako jednoduché administrační rozhraní pro správu několika základních typů dat. Při prvním vstupu na web je uživatel přesměrován na přihlašovací stránku. Jelikož tento server neposkytuje žádná veřejná data, všechny výsledné stránky jsou chráněny a dostupné pouze přihlášeným uživatelům. Po úspěšné autentizaci je uživatel přesměrován na stránku „Jak použít tento web“, kde nalezne stručné informace k jednotlivých stránkám a jak správně nahrát data pro hodinky, včetně textur a modelů. Další stránky slouží především pro správu dat uložených na serveru:

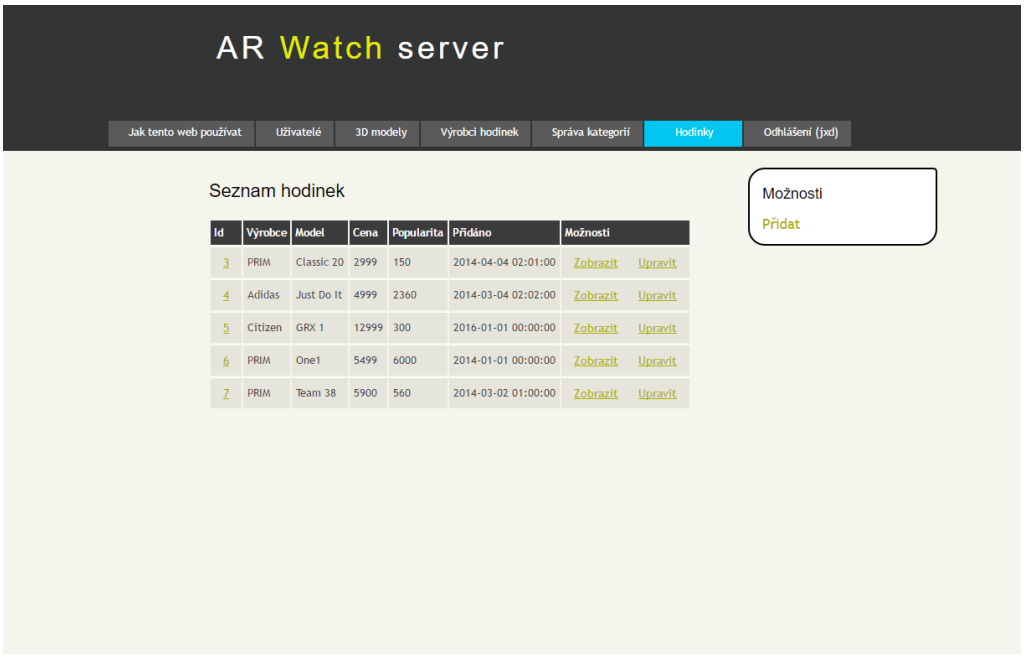
- Uživatelé – zde může již přihlášený uživatel vytvořit účty pro případné další správce s přístupem na tento server
- 3D modely – zde se nachází seznam nahraných 3D modelů s možností jejich administrace
- Výrobci hodinek – při tvorbě nových hodinek je nutno použít výrobce, který je již vložen do tohoto seznamu
- Správa kategorií – zde může uživatel měnit a přidávat kategorie/podkategorie
- Hodinky – správa samotných hodinek. Při vytváření nových je třeba mít již vyplněná data ostatních entit, aby mohly být vybrány pro tvorbu nových hodinek.

5.3.4 Návrh REST komunikace

Pro efektivní stahování dat do mobilní aplikace bylo rozhodnuto o použití REST dotazů na tento server. Výsledná data budou poskytována ve formátu JSON. Pro zajištění funkcionality mobilní aplikace jsou na webový server vyžadovány následující tři typy dotazů – zjištění kategorií hodinek, zjištění modelů a stažení samotných dat hodinek. Pro stahování obrazových souborů či modelů bude dotazem vždy poslán odkaz URL na stažení samostatně, kdy bude potřeba. Zvolená definice těchto dotazů je následujícím

actionGetModels

- HTTP typ: GET
- URL parametry: žádné



Obrázek 5.2: Uživatelské rozhraní webové služby

- Popis: Tato akce je používána aplikací pro získání informací o 3D modelech, které bude potřebovat za svého chodu. Jakožto odpověď vrací seznam všech 3D modelů na serveru, přičemž soubory vrací jen jako jejich odkazy na webu.

actionGetCategories

- HTTP typ: GET
- URL parametry: žádné
- Popis: Při každém spuštění aplikace se nejdříve zeptá serveru na všechny dostupné kategorie a podkategorie, které teprve poté zobrazí uživateli. Odpovědí je tedy seznam rozdělený na dvě skupiny s jednotlivými seznami kategorií.

ActionGetWatches

- HTTP typ: GET
- URL parametry: category, subcategory, page, items

- Popis: Hlavní metoda při komunikaci se serverem, zde se aplikace dotazuje na dostupné hodinky. Parametry „category“ a subcategory“ filtrují výstupní seznam hodinek podle kategorie a podkategorie, zatímco „page“ určuje, kolikátou stránku má server vrátit a „items“, kolik položek má každá stránka. Toto se využívá při větších objemech dat pro omezení náročnosti na síťovou komunikaci. Pokud kterýkoliv z parametrů chybí, výsledky se nefiltrují.

Pro zajištění základní bezpečnosti komunikace je přístup k REST API omezen nutností autentikace. Jelikož se nejedná o citlivá data (uživateli jsou všechna přenesená data později stejně zobrazena), je přenos uživatelských údajů zajištěn pomocí standardu HTTP Basic Authentification, kde je uživatelské jméno a heslo zabaleno a „zašifrováno“ pomocí Base64. Toto není zdaleka robustní a bezpečný přenos, ale vzhledem k typu přenášených dat a faktu, že v tomto API nejsou implementovány žádné metody, které by jakkoliv uložená data měnily, je toto řešení pro tuto službu dostačující.

Kapitola 6

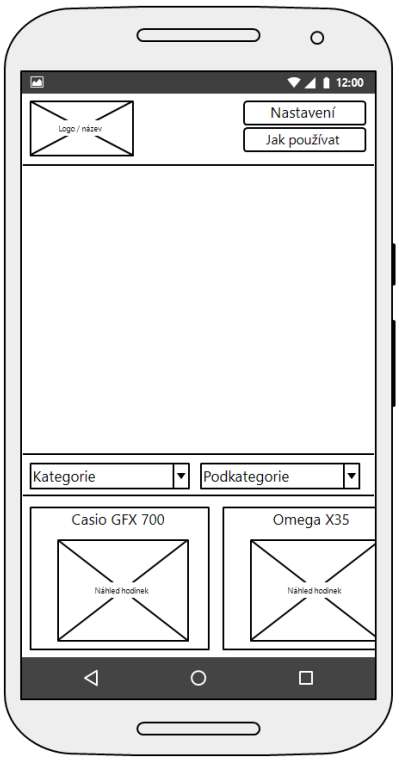
Mobilní aplikace

6.1 Návrh uživatelského prostředí

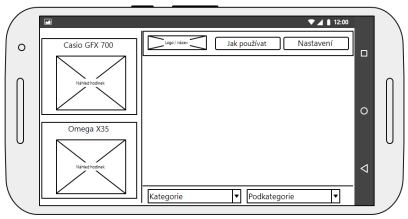
Hlavními požadavky mobilní aplikace tohoto typu jsou uživatelská přehlednost a jednoduchost ovládání, od čehož se odvíjí i návrh rozhraní. V rámci návrhu byly nejdříve vytvořeny wireframy potenciálního uživatelského prostředí aplikace, viz. 6.1 a 6.2. Jedním z prvních designových rozhodnutí bylo, jakou orientaci displaye podporovat, zda pouze vertikální „portrait“, horizontální „landscape“, či obojí. Jelikož uživatel má při plném používání aplikace k dispozici pouze jednu ruku (kde na druhé má zobrazeny hodinky před fotoaparátem), je nutno zvolit variantu, která umožňuje co nejjednodušší používání pouze tímto způsobem.

Jako logická varianta se tedy nabízí portrétní režim. Základní návrh prostředí (viz. 6.2) by mohl podporovat i landscape orientaci, kde by prvky uživatelského rozhraní byly umístěny po stranách obrazovky, to se avšak v praktickém testu prokázalo jako mnohem hůře ovladatelné, než návrh portrétního rozložení, viz 6.1.

Aplikace se skládá z jedné základní obrazovky, kde se informace zobrazují do tří oblastí. V dolní části je vždy přítomen panel s výběrem hodinek. V tomto panelu jsou v horní části nejdříve tři dropdown seznamy na výběr kategorie, podkategorie a stylu řazení výsledků. V dolní části se nachází samotný seznam hodinek na základě daného výběru. Seznam hodinek je dále scrollovatelný do stran, pokud je výsledků více, než lze zobrazit. Tento styl je volen tak, aby uživatel mohl celý panel ovládat co nejsnáze jednou rukou, tj. V tomto případě palcem ruky, která drží telefon. Alternativní řešení byla záměna pozice seznamu hodinek a výběru kategorií/řazení, to se však prokázalo jako neideální, jelikož výběr kategorie se nacházel v naprostém rohu obrazovky a byl proto těžko dosažitelný.



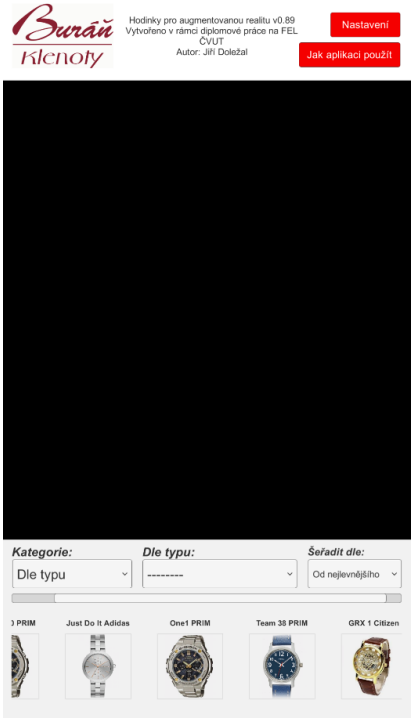
Obrázek 6.1: Návrh uživatelského prostředí v portrétním režimu



Obrázek 6.2: Návrh uživatelského prostředí v landscape režimu

V horní části je v základním režimu zobrazen panel se základními informacemi o aplikaci, to znamená logo společnosti provozující internetový obchod, popis aplikace a tlačítka pro zobrazení návodu pro pomoc s použitím aplikace a zobrazení nastavení. Tento horní panel je navržen tak, aby co nejméně překážel ve viditelnosti aplikace a zároveň sloužil jako náhrada obrazovky „O aplikaci"(viz 6.3).

Pokud uživatel vybere nějaký model hodinek, standardní horní panel je poté nahrazen panelem s informacemi o konkrétních hodinkách (viz 6.4). Zde se nachází název hodinek (výrobce + model), scrollovatelná oblast s obecným popisem hodinek, seznam vlastností hodinek („features“), cena v korunách českých a tlačítko s odkazem do eshopu. Celý tento panel je lehce větší než původní informační. Toto je způsobeno nejenom nutností zobrazit více dat uživateli, ale rozdíl velikostí je zde implementován i z důvodu větší vizuální odlišnosti obsahu. Uživatel je více upozorněn, že byla vykonána akce, jelikož se změní velikost celého horního obsahu. Uživateli,



Obrázek 6.3: Prostředí aplikace - hlavní obrazovka

Obrázek 6.4: Prostředí aplikace - detail hodinek

který se v tuto dobu soustředil na dolní část obrazovky, je tedy poskytnuta větší odezva. V prostřední části obrazovky se standardně nachází oblast zobrazující obraz z fotoaparátu zařízení a případně zobrazené hodinky na detekovaném vzoru. Alternativně se zde zobrazuje dialogové okno návodu k použití aplikace (viz 6.6) či nastavení aplikace (viz 6.5).

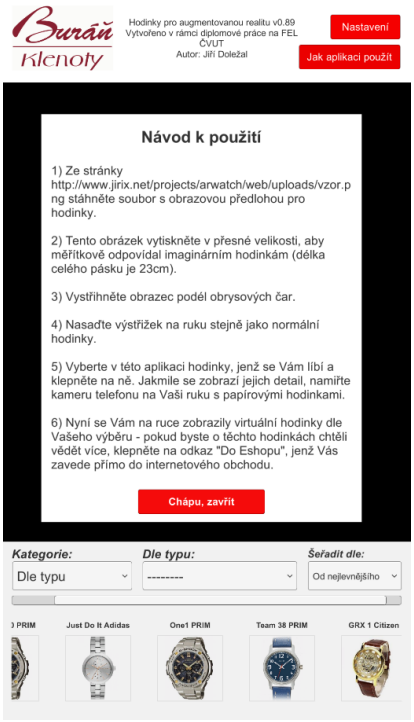
6.2 Struktura aplikace

Mobilní aplikace se skládá z několika vrstev a nezávislých kompoment, které mezi sebou komunikují a poskytují si data. Na 6.7 lze vidět diagram hlavních tříd implementovaných v rámci této aplikace.

6.2.1 Prezentační vrstva

Základem uživatelského rozhraní je ve frameworku Unity prvek Canvas, pod který spadají veškeré UI elementy, a tento prvek řídí jejich správné vykreslování. V rámci této aplikace má každý hlavní prvek uživatelské roz-

Obrázek 6.5: Prostředí aplikace - uživatelské nastavení

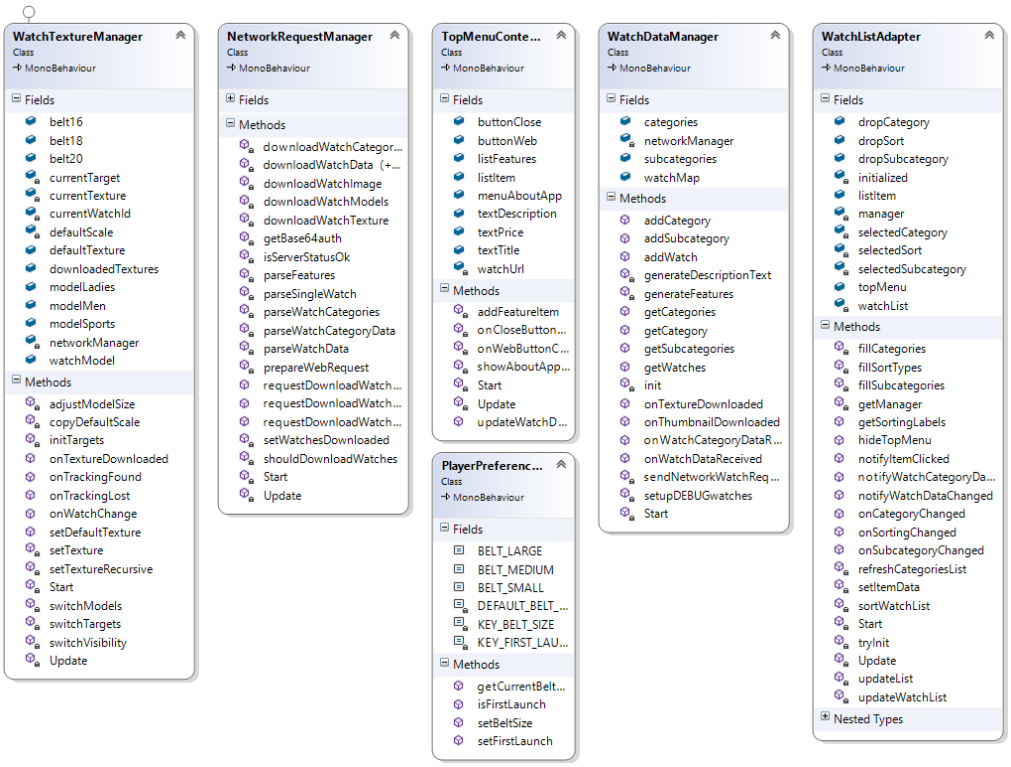


Obrázek 6.6: Prostředí aplikace - jak aplikaci použít

hraní svoji vlastní třídu, která zajišťuje správné zobrazení uživateli. Níže je uveden seznam všech těchto komponent a vysvětlení jejich funkčnosti

BottomMenu Jak již název napovídá, jedná se o panel ve spodní části obrazovky. Hlavní části tohoto panelu jsou tři dropdown pole a scrollovatelný seznam hodinek. Celá logika je zde postavena na skriptu WatchListAdapter, který má na starost zobrazování seznamu hodinek, reakce na změny volby kategorií a zároveň notifikací horního menu v případě výběru konkrétních hodinek.

TopMenu Horní panel pro zobrazení informací o hodinkách. Hlavním ovládacím skriptem je zde TopMenuContentManager, který spravuje zobrazení horního panelu a přepíná mezi zobrazením informací o konkrétním objektu, a pokud žádný není vybrán či uživatel panel zavřel, obecnými informacemi o aplikaci.



Obrázek 6.7: Diagram hlavních tříd aplikace

TopAboutMenu Panel zobrazující obecné informace o aplikaci v případě, že nejsou vybrány žádné hodinky. Jeho zobrazení spravuje skript TopMenuContentManager, stejně jako v předchozím případě.

6.2.2 Síťová vrstva

Pro komunikaci se vzdáleným serverem je k dispozici NetworkRequestManager. Tato třída poskytuje veškeré rozhraní pro současně implementované dotazy na webovou službu. Všechny webové dotazy jsou prováděny asynchronně a volající třída je notifikována, jakmile je obdržena odpověď. Jelikož je přístup k webové službě chráněn proti neautorizovanému přístupu, obsahují všechny dotazy také vložení vlastní hlavičky s HTTP Basic autorizací. Aby se zamezilo opakovanému stahování dat při stejném dotazu, je zde také implementován systém pro zapamatování již stažených dat. Kdykoliv přijde dotaz na stažení nových dat, nejdříve je zjištěno, zda se stejná data již nestáhla v minulosti – pokud ano, volající třída je na tento fakt upozorněna a síťový dotaz není vykonán. V opačném případě je vyvolán dotaz na

webovou službu a pokud je obdržená odpověď v pořádku (očekávaný tvar JSON a zároveň kladný status kód od serveru), je toto poznamenáno. Zároveň tato třída také zpracovává přijatá data a jakožto výsledky již vrací hotové objekty k dalšímu použití. V současné době podporuje následující typy dotazů:

Stažení seznamu kategorií a podkategorií

- Tvar url: /rest/watchcategory
- Typ dotazu: GET
- URL parametry: žádné
- Formát odpovědi: JSON

Základní dotaz pro stažení dat kategorií. Odpovědí je zde strukturovaný JSON seznam rozdělený dle typu data, který je na místě zpracován a jakožto výsledek jsou vráceny dva seznamy obsahující objekty typu WatchCategory.

Stažení dat hodinek pro konkrétní kategorii/podkategorii

- Tvar url: /rest/watch
- Typ dotazu: GET
- URL parametry: category, subcategory, page, items
- Formát odpovědi: JSON

Tento dotaz poskytuje přístup k datům hodinek na serveru. Parametry určují, pro jakou kombinaci kategorie a podkategorie se mají data stáhnout, a zároveň je zde připravena funkcionalita pro stránkování výsledků – parametr „page“ určuje, kolikátou stranu výsledků stáhnout, a „items“ určuje počet položek na stránku. Výsledek je poté zpracován a vráceny jsou konkrétní objekty typu Watch.

Stažení náhledového obrázku pro konkrétní hodinky

- Tvar url: /uploads/images/<název souboru>
- Typ dotazu: GET
- URL parametry: žádné

- Formát odpovědi: byte data

Jelikož při stažení dat hodinek nejsou automaticky staženy dodatečné soubory, je potřeba tyto získat dodatečně samostatně pro každé hodinky. K tomu slouží tento dotaz. V příchozích datech z dotazu na data hodinek je obsažena relativní cesta a název souboru náhledového obrázku. Ten se v této metodě stáhne a konvertuje do typu Texture2D, který je poté navrácen volajícímu objektu.

Stažení textury pro 3D model konkrétních hodinek

- Tvar url: /uploads/textures/<název souboru>
- Typ dotazu: GET
- URL parametry: žádné
- Formát odpovědi: byte data

Stejně jako v dotaze výše, i zde je potřeba získat soubor obsahující texturu hodinek dodatečně. Objekt Watch obsahuje relativní cestu a jméno cílového souboru, který je stažen a konvertován do typu Texture, jenž je poté předán k dalšímu zpracování.

6.2.3 Uložení a správa dat

Aby nebylo nutné data získávat z webové služby při každé změně v uživatelském rozhraní (výběr jiné kategorie), je v aplikaci implementována třída WatchDataManager. Tato třída slouží jako prostředník mezi seznamem hodinek v uživatelském prostředí a službou pro získávání dat z webového serveru a uchovává stažená data pro jejich použití skrz aplikaci. Typický příklad využití zahrnuje WatchListAdapter dotazující se na data, WatchDataManager poskytne dostupná data a dotáže se NetworkRequestManager na nová data. Pokud je dotaz vykonán, tak je WatchDataManager notifikován, nová data jsou uložena a zároveň je WatchListAdapter notifikován a jsou mu předána aktualizovaná data. Data jsou uchovávána dle unikátních ID objektů, zvolené datové struktury jsou typu HashMap (nazývané Dictionary v jazyce C#).

6.2.4 Uložení a správa uživatelských nastavení

Pro persistentní uložení uživatelských preferencí je implementována třída PlayerPreferenceManager. Tato třída umožňuje uložení dat i mezi jednotlivými spuštěními aplikace, což umožňuje například předvyplnit stránku s

uživatelským nastavením na již existující hodnoty. V současné době je tato třída využita pro dva konrétní účely, a to zjištění, zda se jedná o první spuštění aplikace na daném zařízení, a pro uložení velikosti uživatelova pásku pro lepší detekci virtuálních hodinek.

6.3 Rozšířená realita

6.3.1 Obecný princip AR frameworku Vuforia

Pro implementaci rozšířené reality pomocí tohoto frameworku musí uživatel postupovat dle následujících obecných kroků:

- Vytvořit nový uživatelský účet
- Vygenerovat licenční klíč
- Vytvořit novou databázi sledovaných cílů v online administraci
- Vybrat konkrétní způsob způsob trackování (viz níže) a nahrát potřebná obrazová data
- Předchozí bod lze opakovat pro vytvoření více cílů v jedné databázi
- Stáhnout výslednou databázi a tu importovat do mobilního projektu
- Importovat Vuforia SDK do projektu a nastavit dodaný objekt ARCamera jakožto hlavní kameru scény
- Přidat do scény projektu objekt TargetBehaviour, jenž je součástí SDK, a nastavit sledovaný cíl
- Při spuštění aplikace bude nyní automaticky rozpoznáván nastavený cíl

6.3.2 Metody trackování

Framework Vuforia AR nabízí několik možných způsobů, jak trackovat AR objekty. Ve všech případech uživatel vybere jeden z následujících způsobů a nahraje obrazové vzory, které se mají sledovat. Výsledný objekt je poté zpracován a jsou pro něj vytvořeny speciální datové sady pro samotné vyhledávání. Tyto objekty jsou poté importovány do lokálního projektu a nastaveny jako parametr proprietárnímu objektu, jenž je součástí SDK. Tento objekt poté již provádí automaticky vyhledávání a umožňuje notifikovat o nalezení/ztrátě hledaného vzoru.

6.3.2.1 Dostupné způsoby sledování cíle

Framework Vuforia AR nabízí několik možných způsobů, jak trackovat AR objekty. Ve všech případech uživatel vybere jeden z následujících způsobů a nahraje obrazové vzory, které se mají sledovat. Výsledný objekt je poté zpracován a jsou pro něj vytvořeny speciální datové sady pro samotné vyhledávání v aplikaci.

Image Target Základní ze všech typu trackování, Image Target funguje na principu hledání rovinného obrazu v dané scéně. Uživatel nahraje obrázek jakýchkoliv poměrů a ten je poté vyhledáván ve scéně.

Cube target Pokročilejším trackováním je poté Cube target recognition, jenž umožňuje sledovat objekt ve tvaru kvádru – uživatel zadá rozměry/poměry stran tohoto objektu a poté nahraje obrazové vzory pro stěny, podle niž má být objekt rozpoznán. Tento celý objekt je poté pomocí perspektivně hledám ve scéně.

Cylinder target Podobně jako Cube target, Cylinder target umožňuje hledat 3D objekt ve virtuální scéně, avšak v tomto případě se jedná o válec. Uživatel zadá průměr a výšku válce a poté doplní zdrojové obrazce, které se budou sledovat.

3D object tracking Speciální případ trackování, tato možnost je využita spolu se 3D skenováním cílového objektu. Z současné době bohužel Vuforia framework nepodporuje nahrání vlastního 3D modelu. Pomocí proprietárního modulu je naskenován a připraven speciální 3D model, jenž poté slouží jako cíl trackování.

6.3.2.2 Výběr vhodné metody trackování

Pro tento projekt sledování náramkových hodinek na lidské ruce jsou vhodné především dva z výše uvedených způsobů. Image target poskytuje jednoduchou a spolehlivou variantu sledování, jelikož vzorový obraz by bylo možno zakomponovat jako virtuální ciferník hodinek. Nevýhodou tohoto způsobu je ale fakt, že by trackování fungovalo pouze z přední strany hodinek. Pokud by se uživatel chtěl podívat na hodinky z boku či ze zadní strany, vzorový obrazec by již nebyl viditelný, a virtuální hodinky by tedy nebyly vykresleny.

Vylepšením tohoto způsobu by poté bylo 4 nezávislé Image targets, kde každý by byl nadefinován z jedné strany hodinek. Zde by poté platilo, že

ač by uživatel sledoval cíl z jakéhokoliv úhlu (za předpokladu kolmosti k vlastní ruce), minimálně jeden tento vzor by vždy byl v dostatečné kvalitě na rozpoznání, a tudíž by se 3D model hodinek dokázal trackovat neustále. Jediný krajní případ by byl v úhlu 45°od jakýchkoliv dvou vzorů, kde by se musela zajistit dostatečná čitelnost.

Alternativou pro tento projekt se ovšem přímo nabízí použití tzv. Cylinder target, jenž bude reprezentován celým páskem okolo uživatelovy ruky. Jedinou nevýhodou je nemožnost poskytnutí obrazových vzorů pro celý válec, jelikož pásek se bude skládat pouze z pláště, tudíž rozpoznávací schopnost frameworku může být nižší než u standardního Image target.

6.3.3 Návrh vhodného obrazového vzoru

Pro nejlepší výsledky při použití aplikace je nutné navrhnout vhodný obrazový vzor pro co nejlepší sledování v obraze kamery. Pro tento projekt se jedná o válcový vzor bez horního a dolního podstavce, kde plášť je co nejmenší šířky, aby byl pokud možno celý překryt virtuálními hodinkami. Jelikož se jedná o potenciálně komerční produkt, je zde také žádoucí, aby byl výsledný vzor vzhledově adekvátní a pokud možno také reprezentoval obchodníka.

Jakožto potenciální součást vzoru se nabízí použití loga společnosti. V konkrétním případě se jedná a dvouslovné textové logo, kde první část je okrasným písmem a druhá standardním. Toto logo bylo podrobeno analýze na kvalitu z hlediska množství značek, výsledek je možno vidět na obrázku 6.8. Zde lze vidět, že logo obsahuje dostatečné množství sledovacích značek distribuovaných napříč obrázkem. Jediná méně kvalitní oblast je poté velké okrasné „B“, které nemá dostatek ostrých hran, a z tohoto důvodu není vzorově tak kvalitní. Obecně však platí, že použití tohoto loga jako součást vzoru je z hlediska rozšířené reality možné. Jelikož se jedná o hlavní identifikační značku společnosti, umístíme logo na přední stranu pásku do místa, kde se bude zobrazovat ciferník hodinek.

Dále je nutné navrhnout vzhled pásku jako takový. Jelikož analýza textového loga se prokázala jako úspěšná, první návrh celého pásku zahrnuje textové označení horní a dolní části spolu s několika pomocnými obrazci, viz 6.9. Tento vzor má několik zásadních nedostatků – obsahuje velké prázdné úseky, které neobsahují žádné značky na sledování, a zároveň i vložený text neobsahuje tolik značek, aby bylo zajištěno bezproblémové rozpoznávání. Toto je reflektováno také ve výsledném bodovém hodnocení ze strany Vuforia, kde byl tento vzor hodnocen pouze dvěma body z pěti, což značí nedostatečnou kvalitu a možné problémy s rozpoznáváním.

Obrázek 6.8: Příznaková analýza loga společnosti

Obrázek 6.9: Příznaková analýza základního návrhu pásku

Na základě předchozího návrhu byl vytvořen nový, opět založený na využití textového obsahu. V tomto návrhu je použito větší množství textu, ideálně aby zabíral co největší plochu náramku a zároveň je zde implementována možnost proměnlivé délky pásku, kdy uživatel může nastřihnout pásek na jednom ze tří uvedených míst na levé straně, čímž lze ve spojení s nastřižením v pravé části pásek jednoduše ukotvit kolem ruky. Do volného prostoru byly dále přidány snadno sledovatelné obrazce pro podpoření kvality rozpoznávání. Tento vzor již obsahuje mnohem více záchytných bodů po celé své délce a finální hodnocení je čtyři body z pěti.

Jakožto testovací případ pro zkoušku jiného stylu náramku byl vytvořen vzor inspirovaný vzhledem QR kódů. QR kódy jsou „maticové čárové kódy“ navržené tak, aby nesly nějakou informaci a byly snadno strojově čitelné. Z tohoto důvodu byl vytvořen vzor, jen je vidět na 6.11. Ponecháno zde

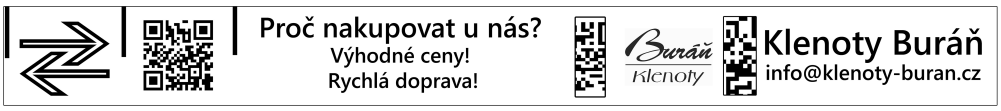
Obrázek 6.10: Příznaková analýza kontrastní fotografie přírody

Obrázek 6.11: Návrh AR pásku s využitím stylu QR kódu

bylo pouze logo společnosti, ostatní texty a jiné objekty byly nahrazeny vzorem inspirovaným z QR kódů. Ačkoliv teoreticky tento vzor obsahuje velké množství ostrých přechodů a zdánlivě se neopakuje, analýza dosáhla výsledku dvou bodů z pěti. Důvodem je fakt, že ačkoliv jednotlivé ostré přechody jsou unikátní, algoritmus seskupuje skupiny těchto bodů dohromady a tvoří větší celky, které se dále porovnávají. Tyto celky se v tomto vzoru již opakují, a to například otočené o násobek 90°, což není na první pohled vidět. Z tohoto důvodu byl tento návrh zavržen.

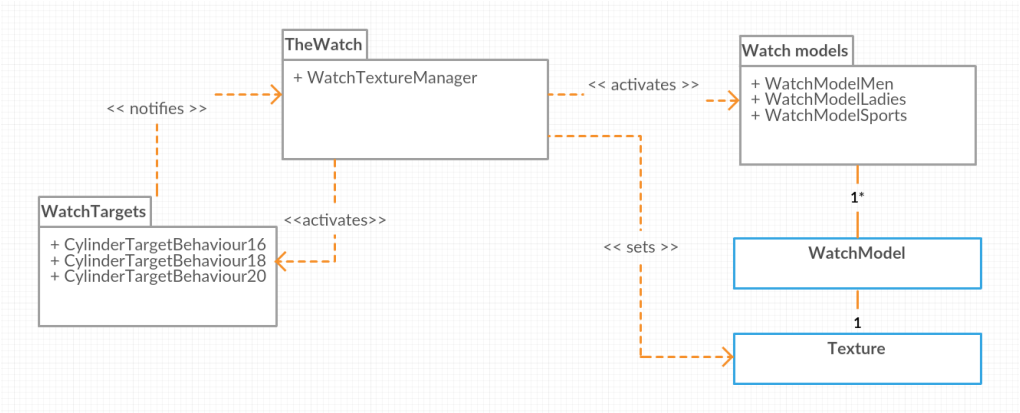
Ačkoliv původní náramek již obsahoval dostatečné množství značek a jeho hodnocení bylo nadprůměrné, v zájmu co nejlepší přesnosti je možné vzor dále vylepšit. Ideální vzory pro rozpoznávání jsou neopakující se kontrastní vzory. Proto je možné celému náramku přidat pozadí, které by tato kritéria splňovalo. Nabízí se například fotografie přírody - v těchto fotografiích bývá velké množství kontrastních přechodů a ty se zpravidla neopakují, proto lze použít celou scénu jako zdroj sledovacích značek. Výsledek takového pokusu je vidět na 6.10. Zde byla použita pestrá fotografie přírody obsahující řadu různých tvarů a detekční algoritmus našel velké množství bodů, což mělo za výsledek hodnocení pěti bodů z pěti.

Tento vzor byl použit i na již existující návrh obsahující logo společnosti spojené s několika reklamními texty. Nicméně problém zde nastal ve faktu, že aby bylo pozadí rozpoznatelné a přidávalo na rozpoznávací hod-



Obrázek 6.12: Návrh výsledného AR pásku s použitím reklamních textů

Obrázek 6.13: Návrh výsledného AR pásku s použitím vhodné fotografie



Obrázek 6.14: Uspořádání AR objektů

notě, muselo být dostatečně kontrastní, což způsobovalo hrubou nečitelnost dalšího obsahu. Proto byl náramek jen lehce vylepšen několika drobnými změnami, pozadí však zůstalo původní, viz 6.12. Tento vzor má již plné hodnocení snadnosti rozpoznávání a neměl by být problém s jeho použitím. Alternativně byl navržen také pásek bez reklamních textů, pouze s logem společnosti a několika pomocnými obrazci, kde většinu výplně pokrývá již zmíněná fotografie; ten je vidět na 6.13.

6.3.4 Integrace do aplikace

Na 6.14 je vidět hierarchické uspořádání a komunikace objektů v ovládání rozšířené reality v aplikaci. Hlavní třídou je zde *WatchTextureManager*. Tento objekt má k dispozici odkazy jak na *TargetBehaviour* objekty poskytnuté SDK Vuforia, tak na výsledné 3D modely pro zobrazení na uživatelově

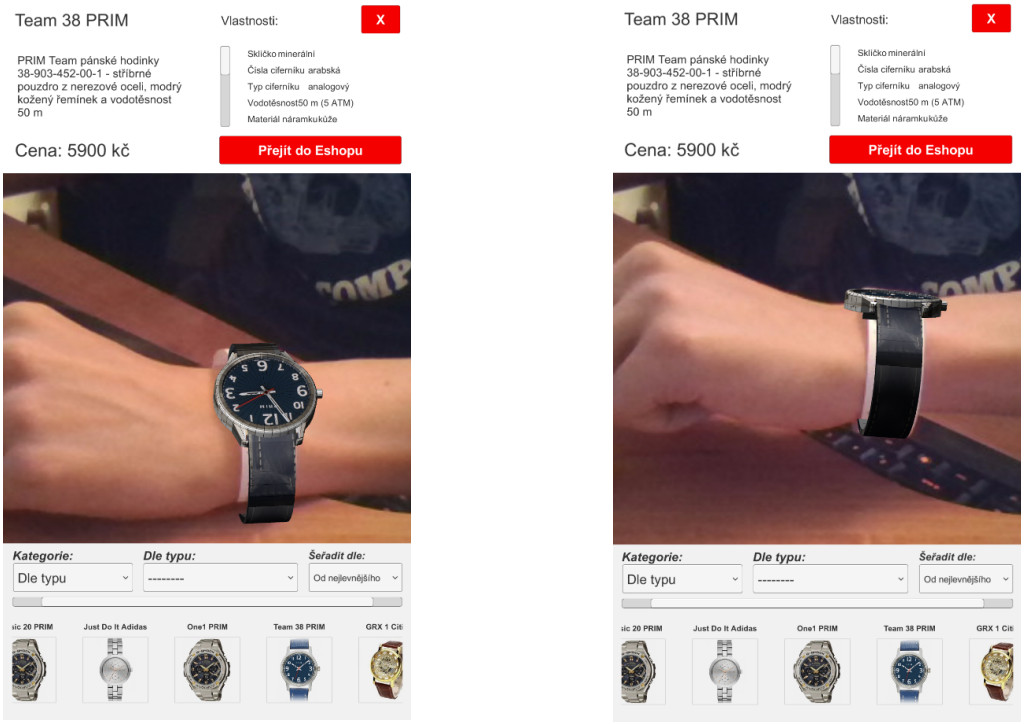
ruce.

TargetBehaviour objekty Jedná se o seskupení speciálních objektů dodaných v rámci SDK pro rozšířenou realitu. Každý tento objekt, pokud je aktivní, v reálném čase analyzuje obraz z kamery zařízení a vyhledává nastavený vzor. V případě nalezení tohoto vzoru je vyslán signál a všichni potomci daného objektu jsou aktivováni. V tomto projektu se nachází tyto objekty tři – pro každou šířku náramku jeden a vždy právě jeden je aktivní. Bohužel není zřejmé, zda SDK Vuforia umožňuje měnit rozpoznávaný cíl za běhu aplikace, v opačném případě by bylo dostatečné použít jeden tento objekt a pouze měnit nastavený cíl.

3D modely hodinek Zobrazované modely jsou uloženy v této složce. V základní verzi aplikace se zde nacházejí tři modely, pro každý typ hodinek jeden (pánské, dámské, sportovní). Všechny modely jsou velikostně připraveny na zobrazení na uživatelově ruce a při spuštění aplikace jsou neaktivní. Zároveň je zde umístěn speciální válcový objekt s komponentou*DepthMask* procházející skrz modely hodinek, která při zobrazování hodinek emuluje lidskou ruku, tudíž není vidět celý model hodinek včetně vnitřní strany pásku.

WatchTextureManager Hlavní třída zodpovědná za zobrazování virtuálních hodinek v této aplikaci. Při startu aplikace je zjištěno konkrétní nastavení velikosti sledovaného opasku a vybrán odpovídající *TargetBehaviour*, který je aktivován. Tato akce je poté vykonána i při jakékoliv změně nastavení velikosti opasku. Zároveň je nastaven defaultní 3D model pro připravení k zobrazení. Kdykoliv objekt *TargetBehaviour* detekuje či ztratí přítomnost sledovaného vzoru, *WatchTextureManager* je notifikován a konkrétně nastavený 3D model je zobrazen či skryt. Pokud uživatel vybere jakékoliv hodinky, jsou data hodinek nahrána do této třídy, 3D model je vyměněn za aktuální k daným hodinkám a je uskutečněn požadavek na existenci textury pro dané hodinky. Pokud tato konkrétní textura ještě nebyla stažena (například nevybral-li uživatel tyto hodinky již v minulosti), je vznesen požadavek na třídu *NetworkRequestManager*. Jakmile je dokončeno stahování textury, tato třída je notifikována a defaultní bílá textura je vyměněna za tuto nově staženou a zároveň uložena pro budoucí použití.

Výsledná funkčnost Výslednou funkčnost zobrazování virtuálních hodinek lze vidět na 6.15. Sledované *TargetBehaviour* objekty jsou pozicovány



Obrázek 6.15: Výsledné virtuální hodinky na uživatelově ruce

tak, aby se virtuální hodinky zobrazily přímo kolem nich. Kvalita rozpoznávání se bohužel ukázala velmi závislá na okolním prostředí. Obzvláště v prostředí s ostrým světlem je cíl špatně rozpoznáván a je potřeba chvíli počkat, popřípadě zkusit natočit pásek pod jiný úhel. Jakmile je ovšem pásek rozpoznán, sledování je plynulé, a pokud uživatel nedělá prudké pohyby (které způsobí rozostření vstupního obrazu, a tedy ztrátu zaměření pásku), hodinky jsou dobře sledovány okolo celé ruky.

Kapitola 7

Testování

V rámci kontroly kvality implementované aplikace bylo provedeno testování s vybranou skupinou uživatelů. Testovány byly mobilní telefony s operačním systémem Android splňující požadavky stanovené výše. V tabulce 7 jsou popsány tyto konkrétní modely a jejich hardwarové specifikace. Účastníci testování byli vybráni na základě potenciální cílové skupiny. Jedná se o muže a ženy ve věku 24 až 50 let s alespoň minimální technickou znalostí mobilních zařízení (zkušenosti s nastavením telefonu, instalací nových aplikací, používání fotoaparátu, odesílání emailových zpráv apod.). Samotné testování se poté skládalo ze dvou částí – z testování uživatelského rozhraní a kvality samotné rozšířené reality.

7.1 Uživatelské prostředí

S účastníky testování byla nejdříve odzkoušena uživatelská přívětivost aplikace. Uživatelům byla aplikace nainstalována, vysvětlen její základní koncept a byli pozorováni, jak se v ní dokáží orientovat. Většina uživatelů byla schopna aplikaci plně ovládat, pouze v jednom případě se účastník zadrhl a jeden účastník měl zápěstí větší velikosti, než se kterým bylo počítáno při

Výrobce, model	Disp.	Procesor	Verze OS	Fotoaparát
Samsung Galaxy S5	5.1’	4x 2.5 GHz	Android 6.1	16MP, f/2.2
Samsung Galaxy S4 Active	5.0’	4x 1.9 GHz	Android 5.0	8MP
Samsung Galaxy S7	5.1’	4x2.3 GHz, 4x1.6 GHz	Android 6.1	12 MP, f/1.7
Huawei P8 Lite	5.2’	4x2.1 GHz, 4x1.7 GHz	Android 7.0	12 MP, f/2.0
Asus Zenfone GO	5.0’	4x 1.3 GHz	Android 5.1	8 MP, f/2.0
Xiaomi Redmi 4 PRO	5.5’	8x 2.0 GHz	Android 6.0	13 MP, f/2.0

Tabulka 7.1: Specifikace mobilních zařízení použitých v testování

Výrobce, model	Celková kvalita	Poznámka
Samsung Galaxy S5	2/5	Velké množství odlesků, problém se zaostřováním
Samsung Galaxy S4 Active	1/5	Nízké rozlišení obrazu z fotoaparátu
Samsung Galaxy S7	4/5	Odlesky
Huawei P8 Lite	4/5	
Asus Zenfone GO	1/5	Nízké rozlišení, trhaný obraz
Xiaomi Redmi 4 PRO	2/5	Problémy se zaostřováním

Tabulka 7.2: Výsledky mobilních zařízení v testování kvality AR

návrhu pásku, tudíž zde musel být pásek upevněn dodatečně. Obecně bylo uživatelské rozhraní hodnoceno pozitivně, avšak bylo poukázáno na několik nedostatků. Především se jedná o nedostatečná vizuální vodítka ohledně scrollovatelného obsahu, ať už v samotném seznamu hodinek, či při zobrazení detailních informací popis hodinek a seznamu jejich vlastností. Dále uživatelům chyběla možnost zavřít okno „Nastavení“ bez uložení změn, či ohraničení textových oblastí v rámci zobrazení detailu konkrétních hodinek. Dále bylo na určitých mobilních telefonech problémové samotnou aplikaci zavřít, jelikož odezva na tlačítko Zpět systému Android není standardně implementována v rámci aplikací vyvinutých v Unity Engine.

7.2 Kvalita AR

Cílem tohoto testu bylo zjistit, jak se liší kvalita implementace rozšířené reality na různých mobilních zařízení. Výsledná data ukazují, že spolehlivost rozpoznávání cíle se velmi liší dle použitého zařízení. Na vině zde byla hlavně špatná kvalita živého přenosu z vestavěné kamery. Obraz byl často přesvícený a především při umělém osvětlení se na papírovém pásku tvořily odlesky, jež bránily správnému rozpoznávání obrazu. Telefony měly problémy také s automatickým zaostřováním - u některých telefonů trvalo dlouhou dobu a případně zaostřilo na jiný objekt. V tabulce **??** lze vidět shrnutí naměřených výsledků dle použitého mobilního telefonu.

7.3 Zhodnocení

Nedostatky týkající se uživatelského prostředí byly analyzovány a bylo rozhodnuto o následujících změnách:

- Přidání horizontálního scrollbaru nad seznam dostupných hodinek.
- Přidání vertikálních scrollbarů po stranách detailního popisu hodinek a seznamu jejich vlastností.
- Přidání tlačítka pro zavření menu „Nastavení“ bez uložení dat.
- Zrychlení řazení a celkově zobrazování seznamu hodinek.
- Ohraničení detailního popisu a seznamu vlastností hodinek bylo přidáno, avšak opět odstraněno, jelikož narušovalo celkový vizuální styl aplikace a tato změna se neprojevila jakožto zlepšení ve viditelnosti daných prvků.

Nedostatečně kvalitní rozpoznávání vzorového pásku u některých mobilních telefonů je významný problém. Pro zlepšení těchto schopností bylo odzkoušeno několik menších změn jako například testování různých režimů automatického ostření u postižených telefonů, avšak tento problém zůstává nevyřešen a je proto jedním z hlavních cílů pro budoucí rozšíření a nasazení aplikace.

Závěr

Výsledkem této práce je mobilní aplikace a webový server poskytující API a webové administrační rozhraní.

Webový server je po funkční stránce připraven pro nasazení s připravenými testovacími daty. Server je vizuálně uzpůsoben požadovanému určení, jednoduchý barevný styl s minimem rušivých prvků. Implementovány jsou zde všechny vytyčené funkce pro administraci a poskytování dat mobilní aplikaci, zároveň je připravena funkcionalita pro nahrání a zobrazení jakéhokoliv modelu hodinek, což může být do budoucna využito pro prémiové produkty, kde každý bude mít svůj vlastní, detailní model.

Mobilní aplikace byla plně implementována za použití všech standardních technik dostupných v Unity Engine, včetně asynchronního načítání dat z webového serveru a jejich uchovávání. Aplikace byla otestována s uživateli, v současné verzi podporuje veškerou vytyčenou funkcionalitu, jakožto filtrování náramkových hodinek dle kategorií, třídění výsledků dle parametrů a samozřejmě zobrazování výsledného modelu na uživatelově ruce. Pro rozšířenou realitu byl navrhnut speciální pásek pro vytisknutí a vystřižení uživatelem, který poté slouží jako vzor pro zobrazení virtuálních hodinek. Pásek je dále navrhnut pro tři různé obvody uživatelova zápěstí a toto nastavení lze pro co nejlepší výsledky nastavit i v mobilní aplikaci.

Další vývoj spočívá především v integraci s reálným internetovým obchodem, ať již přesunem veškeré funkcionality současné webové služby, či implementací API pro automatický import dat z obchodu do implementované služby.

Literatura

[1] Antti Hietanen, J.-K. K., Jukka Lankinen: A Comparison of Feature Detectors and Descriptors for bject Class Matching. Tampere University of Technology.

[2] Bell, D.: FREAK Image Features. [online], 2016, [cit. 2017-05-20]. Dostupné z: `https://artoolkit.org/blog/2016/05/freak-image-features`

[3] Benjamin Wah, e.: *Encyclopedia of Computer Science and Engineering*. Hoboken, New Jersey: John Wiley and Sons, čtvrté vydání, 2008.

[4] Cathroom: CraftAR: Augmented Reality and Image Recognition toolbox. [online], 2017, [cit. 2017-05-20]. Dostupné z: `https://catchoom.com/product/craftar/augmented-reality-and-image-recognition/`

[5] Dalal, N.; Triggs, B.: Histograms of Oriented Gradients for Human Detection. In *Conference on Computer Vision and Pattern Recognition*, INRIA Rhone-Alps, ˆ 655 avenue de l'Europe, Montbonnot 38334, France, 2005.

[6] Google Inc.: Develop | Android Developers. [online], Květen 2013, [cit. 2013-03-23]. Dostupné z: `http://developer.android.com/develop`

[7] Karami, E.: Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images. In *Newfoundland Electrical and Computer Engineering Conference*, IEEE, 2015.

[8] Knesl, J.: Nette 2, Symfony 2, Zend Framework 2. [online], Květen 2013, [cit. 2017-05-20]. Dostupné z: http://www.knesl.com/articles/view/nette-symfony-zend-2

[9] Lowe, D.: *Distinctive image features from scale-invariant keypoints.* International Journal of Computer Vision, 2006.

[10] O'Shea, D.: SIFT and Object Recognition. [online], [cit. 2017-05-20]. Dostupné z: http://www.cs.princeton.edu/courses/archive/spr08/cos598B/Lectures/SIFTandObjectRecognition.pdf

[11] Potencier, F.: Why Symfony2. [online], Září 2012, [cit. 2017-02-13]. Dostupné z: http://fabien.potencier.org/article/65/why-symfony

[12] PTC Inc.: Vuforia library. [online], 2017, [cit. 2017-05-20]. Dostupné z: https://library.vuforia.com/

[13] nyatla Twnkls BV, D. A. C. L. G. J. I. I. T. S. M. R. S. L. M. C. L.: Open Source Augmented Reality SDK | opentoolkit.org. [online], 2017, [cit. 2017-05-20]. Dostupné z: https://artoolkit.org

[14] Ulrich Neumann, S. Y.: Natural Feature Tracking for Augmented-Reality. [online], [cit. 2017-05-20]. Dostupné z: http://graphics.usc.edu/cgit/publications/papers/ARNFieee_mac_final.pdf

[15] Unity Technologies: Unity | Game engine, tools, multiplatform. [online], 2017, [cit. 2017-05-20]. Dostupné z: https://unity3d.com/unity

[16] VisionStar Information Technology (Shanghai) Co. Ltd.: EasyAR. [online], 2017, [cit. 2017-05-20]. Dostupné z: https://www.easyar.com/

[17] Wikitude GmhB: Wikitude SDK documentation. [online], 2017, [cit. 2017-05-20]. Dostupné z: https://www.wikitude.com/documentation/

[18] A tutorial on binary descriptors – part 5 – The FREAK descriptor. [online], 2013, [cit. 2017-05-20]. Dostupné z: https://gilscvblog.com/2013/12/09/a-tutorial-on-binary-descriptors-part-5-the-freak-descriptor

[19] Xamarin Inc.: Xamarin: Mobile App Development and App Creation Software. [online], 2017, [cit. 2017-05-20]. Dostupné z: https://www.xamarin.com/

[20] ©Klenoty-Buran.cz: Klenoty-Buráň.cz. [online], 2017, [cit. 2017-05-20]. Dostupné z: https://www.klenoty-buran.cz/

Příloha A

Seznam použitých zkratek

AR Augmented Reality

URL Uniform Resource Locator

UI User Interface

FREAK Fast REtinA Keypoints

HOG Histogram of Oriented Gradients

SIFT Scale-Invariant Feature Transform

SURF Speeded-Up Robust Features

ORB Oriented FAST and Rotated BRIEF

FAST Features from Accelerated Segment Test

BRIEF Binary Robust Independent Elementary Features

REST REpresentational State Transfer

API Application Programming Interface

PHP Hypertext PreProcessor

JSON JavaScript Object Notation

SQL Structured Query Language

SDK Software Development Kit

BLOB Binary Large OBject

CRUD Create, Read, Update, Delete

ORM Object-Relational Mapping

HTTP HyperText Transfer Protocol

Příloha B

Obsah přiloženého datového archivu

```
readme.txt ..... stručný popis obsahu projektu
aplikace ..... adresář s implementací mobilní aplikace
android.apk ..... spustitelná soubor mobilní aplikace
webserver ..... adresář s implementací webového serveru
thesis.pdf ..... text práce ve formátu PDF
```