VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÁ GALERIE S ROZPOZNÁVÁNÍM OSOB

WEB GALLERY WITH PERSON IDENTIFICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — LUKÁŠ HUBL
AUTHOR

VEDOUCÍ PRÁCE — MICHAL HRADIŠ, Ing., Ph.D.
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Hubl Lukáš**
Obor: Informační technologie
Téma: **Webová galerie s rozpoznáváním osob**
Web Gallery with Person Identification
Kategorie: Uživatelská rozhraní

Pokyny:

1. Vytvořte si přehled o současných metodách a nástrojích pro rozpoznávání tváří pomocí konvolučních hlubokých neuronových sítí.
2. Seznamte se s dostupnými nástroji pro tvorbu webových aplikací
3. Navrhněte webovou aplikaci s automatickým rozpoznáváním osob a vyberte nástroje, které použijete.
4. Vytvořte navrženou aplikaci.
5. Vytvořte si datovou sadu blízkou cílovému zaměření aplikace a vyhodnoťte na ní úspěšnost rozpoznávání osob.
6. Vyhodnoťte vlastnosti uživatelského rozhraní aplikace.
7. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
8. Vytvořte stručné video prezentující vaši práci, její cíle a výsledky.

Literatura:

- Taigman et al.: DeepFace: Closing the Gap to Human-Level Performance in Face Verification. CVPR 2014.
- Parkhi, Omkar M., Andrea Vedaldi, and Andrew Zisserman. "Deep face recognition." Proceedings of the British Machine Vision 1.3 (2015): 6.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Hradiš Michal, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce se zabývá tvorbou aplikace sloužící jako webová galerie pro ukládání a prohlížení fotografií s možností automatické filtrace na základě obličejů. Probírá nástroje pro tvorbu jednotlivých částí aplikace včetně nástrojů pro automatickou detekci obličejů ve fotografiích a následného automatického rozpoznaní identity, včetně jejich implementace do vytvořené webové aplikace. Ukazuje výsledky testů detekce obličejů a rozpoznání osob na datasetu, který byl vytvořen konkrétně pro tuto práci. Na závěr jsou popsány možnosti dalšího postupu pro vylepšení jednotlivých prvků výsledné aplikace.

Abstract

This bachelor thesis focuses on the creation of an application which serves as a web gallery for saving and viewing photographs with the possibility of automatic filtration of faces. The application chooses its instruments for creating particular parts as well as including the instruments for the automatic identity recognition, and its implementation into the created web application. The test results of the face's detection and the recognition of particular persons are shown on dataset, which was created intentionally for this work. Last chapter describes further possibilities for improving particular elements the resulting application.

Klíčová slova

Webová aplikace, HTML, CSS, Bootstrap, JavaScript, Twisted, Ajax, OpenFace, Dlib, HOG, detekce obličejů, konvoluční neuronové sítě, identifikace obličeje.

Keywords

Web application, HTML, CSS, Bootstrap, JavaScript, Twisted, Ajax, OpenFace, Dlib, HOG, face detection, convolutional neural network, face recognition.

Citace

HUBL, Lukáš. *Webová galerie s rozpoznáváním osob*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Hradiš Michal.

Webová galerie s rozpoznáváním osob

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Michala Hradiše, Ing., Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Lukáš Hubl
15. května 2017

Poděkování

Děkuji vedoucímu této bakalářské práce jimž byl pan Michal Hradiš, Ing., Ph.D. za věnovaný čas při seznamování se s principy strojového účení a konvolučních neuronových sítí, za poskytnutou odbornou pomoc a nasměrování při řešení této práce. Dále bych chtěl poděkovat těm, kteří mi dali souhlas požít fotografie, na nichž se nachází, pro experimenty a testování této práce.

Obsah

Kapitola 1

Úvod

Webových galerií pro prohlížení fotografií byla vytvořena již celá řada, ale smyslem této práce bylo vytvořit webovou galerii pro konkrétní cílovou skupinu uživatelů a ke konkrétnímu účelu. Hlavním cílem bylo, za použití konvolučních neuronových sítí, implementovat automatickou detekci obličejů ve fotografiích a nasledně jejich automatickou identifikaci, pomocí které má uživatel možnost vyfiltrovat si a zobrazit pouze fotografie, na kterých jsou zvolené osoby.

Existuje mhoho frameworků pro tvorbu webových aplikací. Jednak serverové části, tyto frameworky jsou stručně popsány v kapitole 3, tak i pro tvorbu klientské části aplikace. Na tyto frameworky je zaměřena kapitola 4. Z těchto frameworků byly pro účely tohoto projektu použity ty, které umožňuji vytvořit aplikaci pracující asynchronně, což je nutnost s ohledem na předpoklad, že danou aplikaci bude využívat více uživatelů ve stejný okamžik.

Jak již bylo zmíněno, tak tato práce se věnuje využití a implementaci neuronových sítí konkrétně pro zpracování obrazu v galerii fotografií. Projektů na zpracování obrazu již také vznikla celá spousta. Je velké množství způsobů jak nahlížet na zpracování obrazu. Na obraz se dá nahlížet jako na celek, a tak vzikají například projekty na transformování jednoho obrazu podle obrazu druhého, ale také se dá pohlížet na jeho jednotlivé části (osoby, zvířata, věci a mnoho dalších) a to má za následek vzik například detektorů libovolnych prvků obrazu.

Kapitola 5 věnuje pozornost detekci obličejů ve fotografiích. Bliže jsou zde popsány nejvíce používané algoritmy pro detekci obličejů a poté knihovny, které tyto algoritmy implementují. V kapitole 6 je bližší pohled na strukturu a základní vlastnosti umělých neuronových sítí včetně konvoluční neuronové sítě, která byla použita v rámci této práce. Postup, podle kterého je postupováno při detekci a identifikaci obličejů je popsán v kapitole 7.

Implementaci každé z části webové aplikace (serverová čast a klientská část) jsou věnovány kapitoly 8 a 9.

Pro účely vyhodnocení úspěšnosti a funkčnosti jednotlivých částí (detekce, identifikce a výsledná webová galerie) bylo vhodné vytvoření vlastního datové sady fotografií (dataset). Dataset bylo nutné vytvořit dostatečně velký, aby bylo možné výsledky vyhodnocení považovat za věrohodné a platné. Testovací dataset byl vytvořen tak, aby se co nejvíce podobal fotografiím, které budou uživatelé nahrávat do vytvořené galerie. Vyhodnocení detekce a identifikace je popsáno v kapitole 11. Kapitola 12 je zaměřena na testování výsledné webové aplikace.

Kapitola 2

Cíl aplikace

V následujících řádcích je popsána cílová skupina uživatelů, pro které byla aplikace vyvíjena, myšlenka pro vytvoření aplikace vytvářené v této práci a cíle aplikace z hlediska funkčnosti.

2.1 Hlavní myšlenka

Hlavní myšlenka pro tvorbu webové aplikace v rámci této práce je vytvořit aplikaci, která bude sloužit k nahrávání fotografií na server, kde bude možné tyto fotografie prohlížet, spravovat, filtrovat a sdílet mezi uživatelovi známé.

Aplikace by měla mít jednoduché a intuitivní ovládání, jednoduché rozhraní a vzhled, aby nepřehlcovala uživatele zbytečnými prvky. Od prvního spuštění by mělo být uživateli jasné, k čemu aplikace slouží a jak se s ní pracuje.

2.2 Cílová skupina uživatelů

Cílovou skupinu uživatelů tvoří zejména moji klienti, pro které pořizuji fotografie ať už ze svateb, oslav, večírků nebo z libovolné události kterou chtějí zaznamenat. Samozřejmě by aplikace měla sloužit i pro jiné uživatele, jenž si chtějí galerii využít.

Od svých klientů vím, že ve spoustě případů, kdy jim předám zhotovené fotografie z nějaké události, chtějí vybrat a následně vyvolat fotografie, na kterých je konkrétní osoba nebo osoby, aby právě této osobě či osobám mohli fotografii věnovat. Tato skutečnost byla impulzem pro vytvoření právě této aplikace.

2.3 Funkčnost aplikace

Jak již bylo naznačeno, aplikace by měla umožnit uživateli nahrávat fotografie do webové galerie, kde si je bude moci prohlížet a hlavně bude moci fotografie filtrovat podle označené osoby.

Základem je, aby si uživatel mohl vytvořit uživatelský účet, skrz který bude do galerie přistupovat. Po vytvoření uživatelského účtu by se uživateli vytvořila galerie, do které si může přidat vlastní fotografie.

Hlavní prvek aplikace by však tvořila možnost nahrané fotografie filtrovat podle obličejů osob, které jsou na fotografii. Filtrování by mělo probíhat automaticky, aby uživatel musel pouze vybrat osoby, podle kterých se budou fotografie filtrovat. To by mohlo mým klientům usnadnit výběr fotografií, jenž si budou chtít vyvolat.

Další funkce aplikace by měla umožnit uživateli sdílet svoji galerii mezi své známe. Těm následně zpřístupnit galerii bez nutnosti mít vlastní uživatelský účet, ale zpřístupnit ji pouze s právem prohlížet si fotografie a filtrovat je.

Jelikož hlavní skupinu uživatelů budou tvořit klienti, jenž svoje fotografie obdrží vždy na DVD, bude aplikace vyvíjena pro použití v desktopovém prostředí.

2.4 Návrh řešení

Jelikož je počítáno s přístupem několika uživatelů ve stejný okamžik, je nutnost vytvořit asynchronní neblokující server, na kterém aplikace poběží. Pro automatické filtrování fotografií podle obličejů osob se hodí konvoluční neuronové sítě, jenž se například používají pro identifikaci osob. Dále je vhodné použít databázový systém, který bude zajišťovat správu uživatelů, ukládat informace o fotografiích a propojovat uživatele s fotografiemi.

Klientské rozhraní aplikace bude dynamické, proto musí být použity vhodné techologie jako jsou šablonovací systémy či technologie pro dynamickou komunikaci mezi serverem a klientskou částí.

Programovací jazyk vhodný pro vytvoření serverové části aplikace je jazyk Python. Pro klientskou část jsou to HTML, JavaScript a CSS.

Kapitola 3

Webová aplikace - serverová část

V následující kapitola je věnována tvorbě jedné z částí webových aplikací, serverové části. Jsou zde přiblíženy nastroje pro tvorbu aplikačních serverů, a také relační databázové systémy.

Obecně lze říci, že server by měl mít na starosti, v rámci webové aplikace, správu uživatelů (jejich registraci, přihlašování a odhlašování), přijimání požadavků z klientské části a následně jejich vyřízení a běh samotné aplikace (spouštění skriptů, renderování stránek podle šablon a za pomoci šablonovacího systému, vytváření soketů pro připojení).

V sekci 3.1 je blíže popsán framework Twisted, který je použit pro tvorbu aplikačního serveru v této práci.

3.1 Twisted

Twisted umožňuje tvorbu asynchronních serverů. Je to event-driven[1] (událostmi řízený) framework. Podporuje obrovskou řadu síťových protokolů, například TCP[2], UDP[3], HTTP[4], IMAP[5] a spoustu dalších protokolů pro síťovou komunikaci. Je šířen pod MIT licencí[6]. Příklad serveru v Twisted je v ukázce 3.1.

Zdrojový kód 3.1: Ukázka serveru v Twisted

```
1 from twisted.internet import protocol, reactor, endpoints
2 class Echo(protocol.Protocol):
3     def dataReceived(self, data):
4         self.transport.write(data)
5 class EchoF(protocol.Factory):
6     def buildProtocol(self, addr):
7         return Echo()
8 endpoints.serverFromString(reactor, "tcp:1234").listen(EchoF())
9 reactor.run()
```

[1]Event-driven https://en.wikipedia.org/wiki/Event-driven_programming
[2]TCP https://en.wikipedia.org/wiki/Transmission_Control_Protocol
[3]UDP https://en.wikipedia.org/wiki/User_Datagram_Protocol
[4]HTTP https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
[5]IMAP https://en.wikipedia.org/wiki/Internet_Message_Access_Protocol
[6]MIT licence https://opensource.org/licenses/mit-license.php

Pro asynchronní chod serveru Twisted využívá tzv. *Deferred objekty*. Tyto objekty řídí volání callbacků[7]

Pomocí tzv. *Site objects* je propojený port, na kterém jsou přijímány HTTP požadavky ze strany klienta, s tzv. *Resource objects*. Resource objekty mohou být organizovány do DOM stromu a reprezentují jednotlivé URL segmenty stránky. Do DOM stromu se Resource objekty vkládají pomocí funkce **`putChild`**. Pro snažší pochopení problematiky týkající se Resource objektů slouží ukázka 3.2. Při takovémto použití budou validní následující stránky:

- http://example.com/
- http://example.com/fred
- http://example.com/bob
- http://example.com/fred/
- http://example.com/bob/

Zdrojový kód 3.2: Ukázka použití Resource objects

```
1 from twisted.web.resource import Resource
2
3 root = Hello()
4 root.putChild('fred', Hello())
5 root.putChild('bob', Hello())
6 reactor.run()
```

Pro každou listovou část DOM stromu ('fred', 'bob') může být, v aplikaci vytvořené v rámci této práce tomu tak je, přiřazen jiný Resource objekt. Při této implementaci jsou jednolitvým URL segmentům přiřazovány různé Resource objekty. Výše uvedený příklad by byl upraven tak, jak je vidět v ukázce 3.3. V tomto případě by při zobrazení stránky 'http://example.com/fred' byla vytvořen Resource objekt 'Bye', v kterém by byly vyřizovány požadavky právě z této stránky. [19]

Zdrojový kód 3.3: Ukázka použití Resource objects

```
1 from twisted.web.resource import Resource
2
3 root = Hello()
4 root.putChild('fred', Bye())
5 root.putChild('bob', Goodbye())
6 reactor.run()
```

3.2 Databázové systémy

Databázový systém slouží k definici dat, která mají být ukládána v databázi. Definuje vztahy mezi ukládanými daty, způsob přístupu k nim a jaké operace je možné s uloženými

[7] Callback https://en.wikipedia.org/wiki/Callback_(computer_programming)

daty realizovat. Také řeší oprávnění k přístupu a manipulaci s daty včetně správy uživatelů jež mohou s daty manipulovat.

Databázové systémy velice usnadňují práci s daty a jejich zabezpečením, jelikož není nutné vytvářet vlastní zabezpečovací systém pro přístup k datům. Programovací jazyky umožňují velice snadné propojení databázového systému s programovou částí, což je další velké usnadnění například při tvorbě webových aplikací, které pracují s daty. [23]

3.2.1 SQLite

SQLite je *relační* databázový systém. Je implementován jako knihovna v jazyce C. Narozdíl od databázových systémů, jako je například *MySQL*, kdy jde o systém na bázi klient-server a databázový server je spuŠtěn jako samostatný proces, je SQLite pouze knihovna, kterou stačí v aplikaci naimportovat a za použití jednoduchého rozhraní využívat.

Celá databáze vytvořená pomocí SQLite je uložena v jediném souboru, jenž není závislý na platformě, což umožňuje použít využít SQLite v libovolném operačním systému. Podpora programovacích jazyků je také velmi široká. Tento databázový systém lze využít ve všech nejpoúživanějších programovacích jazycích. Implementuje téměř celý jazyk SQL. [13]

Kapitola 4

Webová aplikace - klientská část

V této kapitole jsou popsány nástroje použité pro vytvoření klientské části aplikace.

4.1 HTML

HTML (HyperText Markup Language) je značkovací jazyk používaný pro tvorbu webových stránek, vzájemně propojených hypertextovými odkazy. HTML je nejpoužívanější značkovací jazyk pro tvorbu webových stránek.

Jeho nejnovější stabilní verze je HTML5. HTML byl vyvinut z dříve používaného, rozsáhlého značkovacího jazyka SGML (Standard Generalized Markup Language). Stejně jak se vyvíjí webové prohlížeče, vyvíjí se i jazyk HTML.

Je charakterizován množinou značek a jejich atributů. Mezi tyto značky se uzavírají části obsahu dokumentu, což určuje význam obsaženého textu a tvoří tzv. element dokumentu. Značky jsou, až na několik vyjímek, párové. Koncová značka je shodná s počáteční, ale s rozdílem, že před názvem koncové značky je lomítko. Podle jistých pravidel je možné jednotlivé elemnty vkládat do sebe. Z významového hlediska se značky dělí na:

- **Strukturální značky** - rozvrhují strukturu dokumentu, například nadpisy (`<h1>`, `<h2>`) nebo odstavce (`<p>`)
- **Popisné (sémantické) značky** - popisují povahu obsahu prvku, například nadpis (`<title>`) nebo adresa (`<address>`)
- **Stylistické značky** - při zobrazení určují vzhled prvku, například tučné písmo (`<b>`), od těchto značek se téměř upustilo a byly nahrazeny kaskádovými styly CSS, které jsou popsány v sekci 4.2.

Struktura HTML dokumentu má předepsanou strukturu. Tvoří ji:

- **Deklarace typu dokumentu** - například značka (`<!DOCTYPE html>`)
- **Kořenový element** - prvek `html` (značky `<html>` a `</html>`)
- **Hlavička dokumentu** - prvek `head` (značky `<head>` a `</head>`)
- **Tělo dokumentu** - prvek `body` (značky `<body>` a `</body>`)

[3]

4.2 CSS

CSS (Cascading Style Sheets) je jayzk pro definici způsobů zobrazení elementů webových stránek, psaných v jazycích HTML, XHTML[1] a XML[2]. Umožňuje autorům webových stránek oddělit definici vzhledu dokumentu od jeho struktury a obsahu. V porovnání se stylistickými značkami v HTML nabízí kaskádové styly rozsáhlejší možnosti formátování dokumentu.

Definice kaskádových stylů se skládá z posloupnosti několika pravidel. Na začátku každého pravidla je tzv. selektor, který specifikuje element, popřípadě skupinu elementů, na který budou daná pravidla aplikována. Za selektorem následuje blok deklarací, určujících vzhled vybraného elementu, či skupiny elementů. Blok deklarací je uzavřen do složených závorek. Jednotlivé deklarace jsou odděleny středníkem. Elementům v HTML dokumentu jsou přiřazovány identifikátory nebo jsou řazeny do tříd, které obvykle obsahují více elementů. Pomocí selektorů se přistupuje právě k těmto identifikátorům a třídám.

Nejnovější verze kaskádových stylů je CSS3. [20]

4.3 Bootstrap

Bootstrap je webový framework určený pro vývoj webových aplikací. Za použití HTML, CSS a JavaScriptu, které implementuje, umožňuje vytvářet aplikace pro většinu webových prohlížečů jednak v desktopové verzi, tak i ve verzi pro mobilní zařízení.

Jeho výhodou je jednoduché zpracování libovolného uživatelského rozhraní back-endových i front-endových částí aplikací. Podporuje tzv. responzivní design, což znamená, že se rozložení stránky dynamicky přizpůsobuje zařízení, na kterém je stránka zobrazována.

Bootsrap je modulární a sestává ze série stylopisů, které implementují různé komponenty. Pro použití Bootsrapu je nutné stáhnout jeho komponenty a nalinkovat je do hlavičky HTML dokumentu. Původně byl Bootstrap vyvinut jako framework pro Twitter. [21]

4.4 Javascript

Jedná se o multiplatformní, objektově orientovaný scriptovací jazyk. Obvykle jsou pomocí JavaScriptu ovládány různé interaktivní prvky uživatelského rozhraní, například tlačítka nebo textová pole. Slouží také pro tvorbu animací a různých efektů obrázků.

Co se týče syntaxe, jedná se o podobný jazyk jako je rodina jazyků C/C++/Java. Sémanticky je však odlišný. Program v JavaScriptu se obvykle spouští na straně klienta, až po načtení webové stránky ze serveru. Do HTML dokumentu se vkládá buďto přímo, pomocí párové značky `SCRIPT`, nebo pomocí odkazu na externí soubor obsahující program v jazyce JavaScript.

Části programu v JavaScriptu jsou obvykle spouštěny během interakce s uživatelem a to jako reakce na událost, kterou uživatel vykonal, například kliknutí na některý element, stisknutí klávesy nebo přejetí kurzorem myši přes element. Pro každý element může být definováno několik událostí, při kterých se spouští část programu. Jedná se o tzv. *obsluhu událostí*, kdy je pro určité operace s elementy definováno, jaké části programu se mají vykonat. [10]

[1] XHTML https://cs.wikipedia.org/wiki/Extensible_HyperText_Markup_Language
[2] XML https://cs.wikipedia.org/wiki/Extensible_Markup_Language

4.5 Ajax

Ajax (Asynchronous JavaScript and XML) je knihovna pro jazyk JavaScript. Vyžaduje použití novějších verzí webových prohlížečů. Ajax pro svoji činnost využívá následující technologie:

- HTML (případně XHTML) a CSS pro prezentaci informací
- DOM[3] a JavaScript pro zobrazování a dynamické změny prezentovaných informací
- XMLHttpRequest[4] pro asynchronní výměnu dat s webovým serverem

Největší výhodou Ajaxu je možnost vytvářet interaktivní webové aplikace, které mění svůj obsah, bez nutnosti jejich kompletního opětovného načítání, za pomoci asynchronního přenosu dat. Například pokud uživatel klikne na tlačítko, které má aktualizovat část obsahu na stránce, tak bez použití Ajaxu by se musela přenačíst celá stránka, ale za použití Ajaxu vše proběhne na pozadí. Server zašle pouze ty části stránky, které se změnily a pouze ty se uživateli na stránce aktualizují.

Jako nevýhoda Ajaxu se dá považovat zvýšení počtu vyměňovaných HTTP požadavků mezi uživatelem a serverem, a třebaže se přenaší nižší množství dat, tak pří špatné implementaci zátíženost serveru neklesne. [11]

[3] DOM https://cs.wikipedia.org/wiki/Document_Object_Model
[4] XMLHttpRequest https://cs.wikipedia.org/wiki/XMLHttpRequest

Kapitola 5

Detekce obličejů v obraze

Tato kapitola je zaměřena na algoritmy používané pro detekci obličejů v obraze, knihovny určené pro detekci a je zde také popsán postup při detekci obličejů v obraze.

5.1 Algortimy pro detekci obličejů v obraze

Přístupů k detekci obličejů v obraze je celá řada. Je možné využít například tzv. znalostní metody, kdy se obličej detekuje na základě vzorů. Při detekci na základě vzorů se může vycházet z faktu, že obličej má dvě oči a jedny ústa. Mnoho znalostních metod je založeno na naučení detektoru podle sady vzorů.[14]

5.1.1 Viola - Jones detektor

Jedna z nejpopulárnějších metod pro detekci obličejů v obraze byla vytvořena Violou a Jonesem v roce 2001. Metoda pracuje s obrazy ve stupních šedi a pracuje ve 3 krocích během zpracování obrazu. Využívá integrálního obrazu[1] pro rychlé počítání příznaku z obrazu, Haarovy příznaky[2], zobrazené na obrázku 5.2, které jsou odvozeny od obdélníků, které se dělí na hranové, čárové a středové. Toto pojmenování je odvozeno od informace, kterou detekují. Jako vyhodnocovací algoritmus využívá AdaBoost[3].[27] Tato metoda je implementovaná v OpenCV.

5.1.2 Histogram orientovaných gradientů

Histogram orientovaných gradientů (dále pouze HOG) je algoritmus vycházející se směru a síly gradientů, což jsou úhly hrany v každém bodě vstupního obrazu. Dokaže pracovat jak s obrazy ve stupních šedi, tak i s barevnými obrazy. Nejprve jsou vytvořeny tzv. HOG descriptory, což jsou příznaky, které jsou následně předány vyhodnocovacímu algoritmu. HOG descriptor se skládá z množství histogramů vypočítaných pro každý pixel vstupního obrazu.

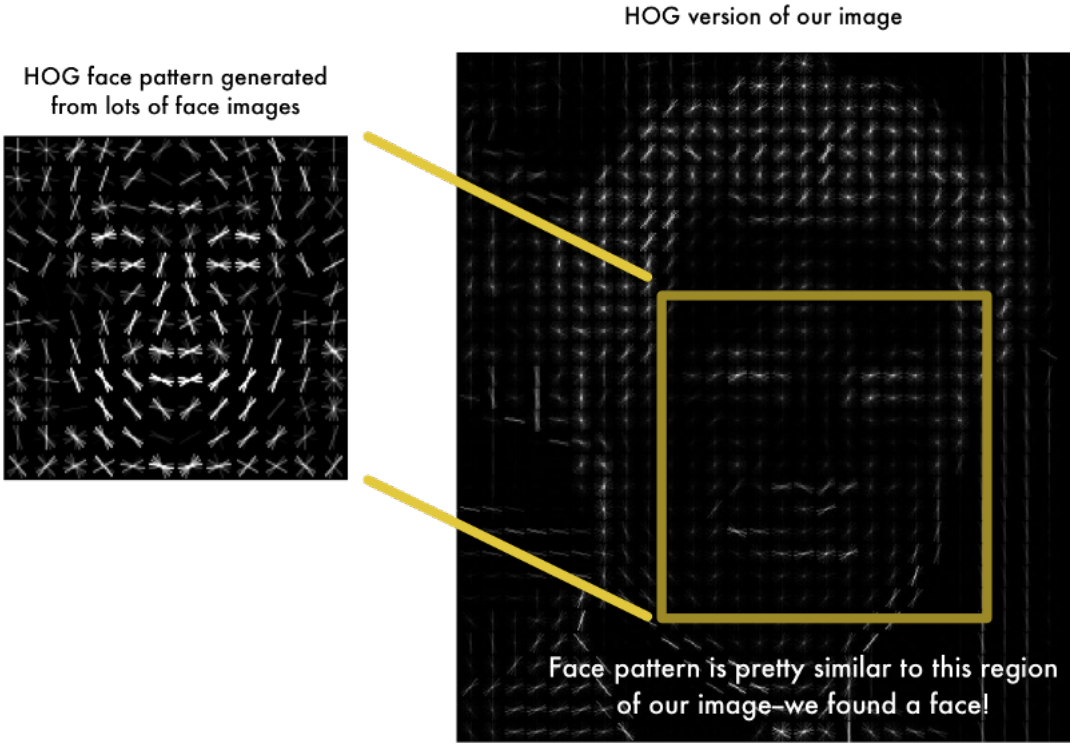
Vyhledání objektu v obraze probíhá v několika krocích. Nejprve se provede konvoluce obrazu pomocí vhodného jádra pro výpočet derivace ve směru x a y. Poté se vypočtou gradienty pro všechny pixely vstupního obrazu. Následuje segmentace obrazu na menší částí. Pro

[1] Integral image https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/

[2] Haar features http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html

[3] AdaBoost http://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/

každou část se následně vypočítá vážený histogram. Nasledně proběhne segmentace obrazu na oblasti, které se vzájemně překrývají. Nad těmito oblastmi proběhne normalizace jejich histogramů. Nakonec je součet normalizovaných histogramů odeslán pokročilému klasifikátoru. V tomto případě klasifikátoru SVM.[9] Tato metoda je implementována v knihovně Dlib[4]. Na obrázku 5.1 je vidět použití metody HOG.



Obrázek 5.1: Použití metody HOG. Vlevo je vygenerovaná šablona, která vznikla natrénováním na množství obrázků obličejů. Vpravo je HOG verze vstupního obrazu. Po porovnání se šablonou se dá konstatovat, že na obrázku vpravo je obličej. Převzato z [12]

5.1.3 SVM klasifikátor

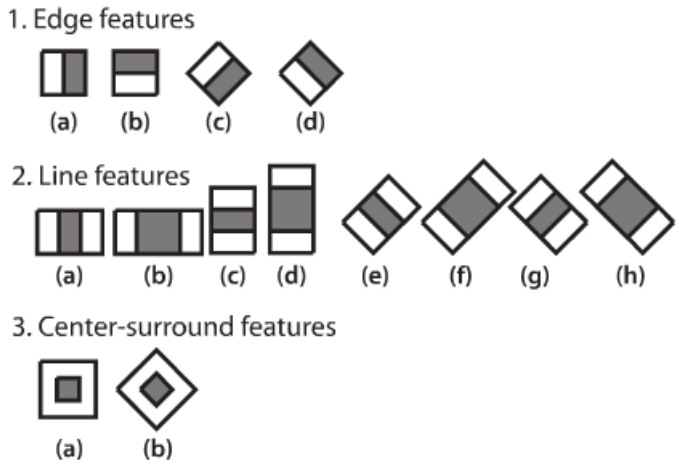
SVM - Suport Vector Machine je jeden z algoritmů strojového učení, který může být použit jak pro klasifikaci, tak pro regresi. Nejvíce je však využíván právě pro klasifikaci. Rozděluje vstupní data do skupin bodů, které je možné následně rozdělit do n-dimenzionálního prostoru. To má za následek vznik možnosti využití složitých nelineárních funkcí. [9]

5.2 Knihovny pro detekci obličejů v obraze

5.2.1 OpenCV

Je to volně dostupná knihovna. OpenCV podporuje implementaci v programovacích jazycích C[5], C++, Python, Java[6] a MATLAB[7]. OpenCV je multiplatformová knihovna a

[4] Dlib http://dlib.net/
[5] C http://www.cprogramming.com/
[6] Java http://www.oracle.com/technetwork/java/index-138747.html
[7] MATLAB https://www.mathworks.com/products/matlab.html



Obrázek 5.2: Haarovy příznaky, převzato z [5]

podporuje operační systémy Windows[8], Linux[9], Android[10] a Mac OS[11]. Je šířena pod licencí BSD[12]. V OpenCV je implementována metoda Viola-Jones pro detekci obličejů v obraze, která byla zmíněna v sekci 5.1.1. Je to jedna z nejpoužívanějších knihoven buďto pro detekci obličejů v statických obrazech nebo pro detekci obličejů ve videu.[8]

5.2.2 Dlib

Dlib je stejně jako OpenCV volně dostupná hknihovna. Byla vytvořena v programovacím jazyce C++. Primárně je určená pro programovací jazyk C++, nicméně má rozhraní i pro Python. Dlib je také multiplatformová, tudíž podporuje operační systémy Windows, Linux, MacOS, Solaris[13], BSD[14] a HP-UX[15]. Je šířen pod licencí Boost[16]. Oproti knihovně OpenCV, která je určena pro zpracování obrazu, má Dlib mnohem širší využití. Dlib lze využít například při řešení numerických problémů, v oblasti strojového učení nebo pro zpracování obrazu. Také implementuje nástroje z oblasti síťových služeb, tudíž umožňuje vytvářet například TCP sokety[17]. Za použití knihovny Dlib lze spouštět procesy v oddělených vláknech, jelikož implementuje tzv. threading[18]. Dlib při detekování objektů v obraze využívá metodu HOG popsanou v sekci 5.1.2.[17]

Knihovna Dlib je použita v rámci této bakalářské práce pro účely detekování obličejů ze statického obrazu.

[8]Windows https://www.microsoft.com/cs-cz/
[9]Linux https://www.linux.com/
[10]Android https://www.android.com/
[11]MacOS https://www.apple.com/cz/macos
[12]BSD https://en.wikipedia.org/wiki/BSD_licenses
[13]Solaris https://www.oracle.com/solaris/solaris11/index.html
[14]BSD operating systemhttps://www.freebsd.org/
[15]HP-UX http://www8.hp.com/us/en/products/servers/hp-ux.html
[16]Boost licence http://www.boost.org/LICENSE_1_0.txt
[17]TCP socket https://www.scottklement.com/rpg/socktut/introduction.html
[18]Threading http://dlib.net/api.html#threads

Kapitola 6

Umělá neuronová síť

V této kapitole je popsána jedna z částí strojového učení, neuronové sítě, které mají využití při identifikaci osob v obraze a byly tudíž využity v této práci. Dále je zde zmíněn princip činnosti neuronových sítí, jejich využití, základní stavební prvek neuronových sítí (neuron) a jejich učení.

6.1 Neuronová síť

Neuronová síť je algoritmus, který je inspirovaný stavbou lidského mozku a jeho činností. Stejně jak v lidském mozku, je i v umělé neuronové síti základním kamenem nervová buňka - neuron. Zatímco v mozku jsou neurony vzájemně propleteny synapsemi a řízeny elektrickými impulzy, v umělé neuronové síti jsou propojeny spoji, které jsou ohodnoceny váhami. Nejdůležitější vlastností neuronu v oblasti umělé inteligence je schopnost, na základě vstupních dat, adaptovat již zmíněné váhy a z toho vyplývá, že neuronová síť tvořená právě těmito neurony je schopna se učit. Tím poskytuje širokou škálu možností využití v oblasti analýzy dat. Dále budou v práci umělé neuronové sítě označovány pouze jako neuronové sítě. [25]

6.2 Využití neuronových sítí

Existuje mnoho druhů neuronových sítí a dá se konstatovat, že každá z nich se hodí k řešení odlišného problému. Neuronové sítě tak najdou využití například v letecké dopravě, ve zdravotnictví, v oblastni zpracování obrazu či zvukových signálu, ale třeba také v oblasti finančních služeb, či zdravotnictví.

6.3 Neuron

Jak již bylo řečeno, neuron je základem neuronových sítí. Formální neuron má zpravidla n vstupů a jeden výstup, což vyplývá z obrázku 6.1. Podle tohoto modelu lze neuron popsat vztahem:

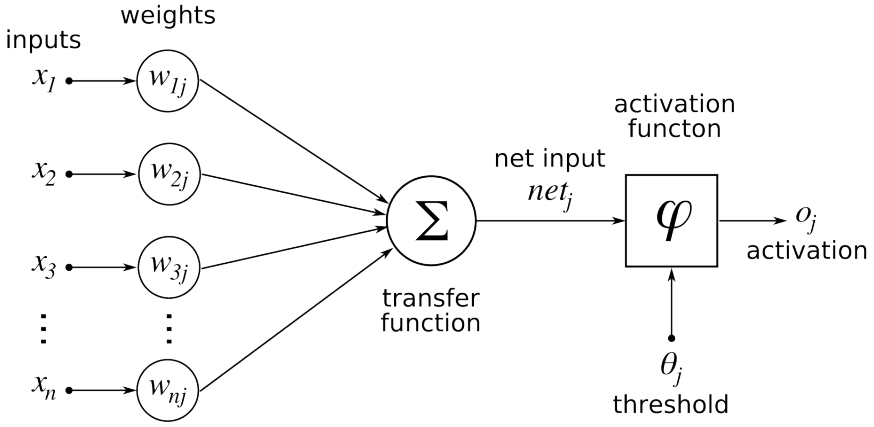
$$y = \varphi\left(\sum_{i=1}^{n} \omega_i x_i + \theta\right) \tag{6.1}$$

, kde:

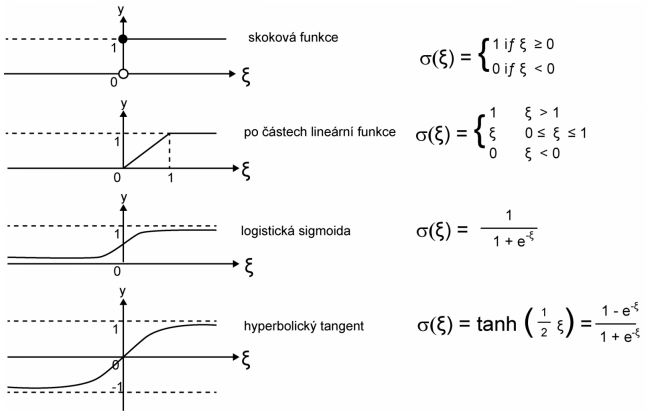
- y je výstup neuronu

- φ značí aktivační funkci neuronu
- x_i značí vstup neuronu
- ω_i jsou váhy, kterými je násoben vstup x_i
- θ značí prahovou hodnotu aktivační funkce

Nejčastěji používané aktivační funkce jsou zobrazeny na obrázku 6.2. Neurony v neuronových sítích jsou vzájemně propojeny a uspořádány do vrestev. [16]



Obrázek 6.1: Model neuronu, převzato z [26]



Obrázek 6.2: Nejpoužívanější aktivační funkce, převzato z [4]

6.4 Učení neuronových sítí

Učení neuronové sítě lze zjednodušeně popsat jako opakovaný proces, kdy se neuronové síti předkládají vstupní vzorky a na základě výstupu z neuronové sítě se zpětně upravují váhy tak, aby se co nejvíce minimalizovala výstupní chyba. Využívá se algoritmu zpětné propagace chyby. Tento algoritmus je založen na výpočtu chyby při porovnání výstupu s očekávaným výstupem a tato chyba se zpětně šíří celou neuronovou sítí. Tím se váhy,

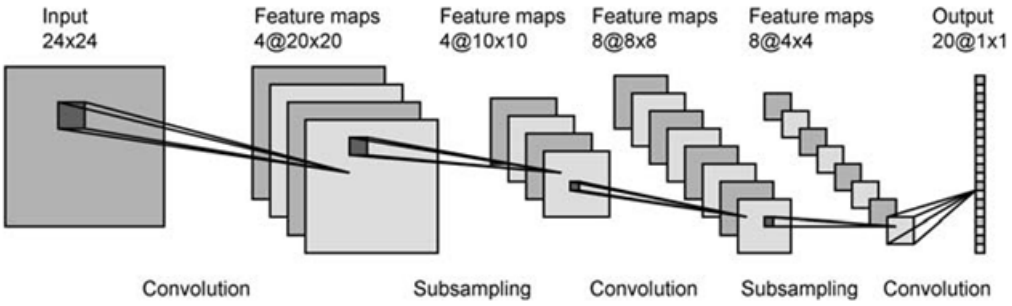
kterými se násobí vstupy upravují, aby bylo dosaženo lepších výstupů a tím pádem menší chyby.[24]

Stav, kdy celá trénovací sada jednou prošla neuronovou sítí se nazývá epocha. Trénování sítě trvá mnoho epoch a s rostoucím počtem epoch, které byly vykonány se natrénování sítě v ideálním případě zlepšuje. Ukončení trénování může být nastaveno například počtem epoch, popřípadě dosažením požadovaného minima chybové funkce. Neúspěch trénování neuronových sítí může být způsoben nedostatečným množstvím vstupních dat, popřípadě tím, že ve vstupních datech není obsažena specifická informace, z které by neuronová síť mohla vyvodit požadovaný výstup. [4]

6.5 Konvoluční neuronová síť

Konvoluční neuronové sítě se řadí mezi vícevrstvé neuronové sítě, které se v praxi používají pro zpracování obrázků či videa. Neurony v konvolučních sítích mezi sebou sdílejí váhy. Mohou mít různou architekturu, ale ve výsledku jsou velmi podobné. Schéma typické konvoluční sítě je znázorněno na obrázku 6.3. Je zde vidět, že konvoluční síť je složena z několika vrstev, z nichž první je konvoluční vrstva.[6]

Konvoluční vrstvy bývají často následovány vrstvami subsamplingovými. Subsamplingová vrstva je blíže popsána v sekci 6.5.2. Za těmito vrstvami může následovat vrstva plně propojená, která propojuje všechny výstupy předchozí vrstvy se všemi svými vstupy. Výstupem konvoluční neuronové sítě bývá jednorozměrný vektor. Tento vektor se přivadí do neuronové sítě, která vypočítá konečný výstup.[18]



Obrázek 6.3: Konvoluční neuronová síť, převzato z [1]

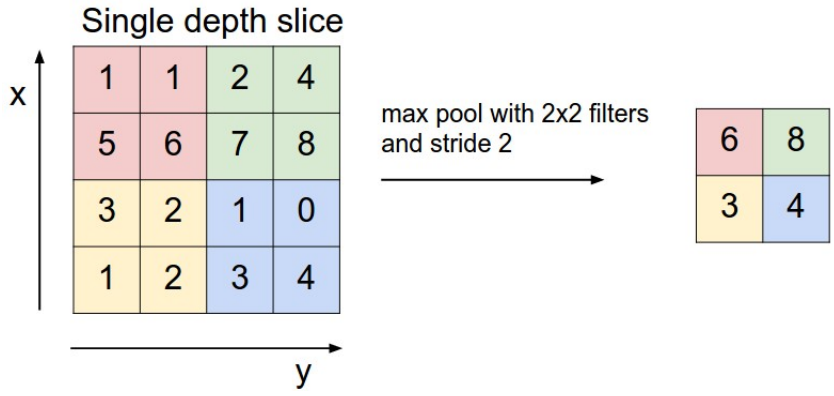
6.5.1 Konvoluční vrstva

Konvoluční vrstva se skládá z několika příznakových map. Tyto příznakové mapy jsou vytvářeny pomocí sady filtrů (jádra). Příznakové mapy jsou v podstatě výsledkem aplikace daného filtru přes celý vstupní obrázek. Počet příznakových map je závislý na počtu použitých filtrů v dané vrstvě. Každá příznaková mapa si klade za cíl získat ze vstupního obrazu lokální charakteristiky, což je možné chápat tak, že ze vstupního obrazu o velikosti 24x24 bude zpracovávat podobraz o velikosti 20x20. Tento příklad je vidět i na obrázku 6.3.[18]

6.5.2 Subsamplingová vrstva

Subsamplingové vrstvy, uváděné také jako pooling vsrtvy, shlukují okolní pixely do jednoho, čímž snižují velikost vstupního obrazu. Existuje několik variant pooling vrstev. Například

max-pooling, znázorněný na obrázku 6.4, při které se určuje největší hodnota nebo average pooling, kdy se bere průměr hodnot.[18]



Obrázek 6.4: Max-pooling, převzato z [15]

6.6 Frameworky pro práci s neuronovými sítěmi

Frameworků[1] pro práci s neuronovými sítěmi je hned několik. Mezi nejpoužívanější patří například Caffe[2], Keras[3] nebo Torch[4]. Dále jsou však popsány pouze ty, které byly použity v rámci této práce.

6.6.1 Torch

Torch je open-source framework pro strojové učení vytvořený v jazyce *C*. Je určený pro skriptovací jazyk LuaJit[5] s podporou CUDA[6]. Je vhodný pro řešení široké škály problémů z různých oblastí jako je počítačové vidění či zpracování signálu. Má API[7] pro Python - PyTorch[8]. Umí pracovat jak v CPU[9], tak v GPU[10] režimu.[7]

6.6.2 OpenFace

Pro účely aplikace vytvářené v rámci této bakalářské práce, byl vybrán právě tento plugin. Openface je nadstavba frameworku Torch vytvořená jako knihovna pro Python. Je určený ke konkretnímu účelu - identifikaci osob v obraze. Pracuje jak v CPU, tak v GPU režimu. Využíva matematickou knihovnu pro Python Numpy[11] a OpenCV[12], která slouží pro zpracování a transformaci obrazu.

[1] Framework https://cs.wikipedia.org/wiki/Framework
[2] Caffe http://caffe.berkeleyvision.org/
[3] Keras https://keras.io/
[4] Torch http://torch.ch/
[5] LuaJit http://luajit.org/
[6] CUDA http://www.nvidia.com/object/cuda_home_new.html
[7] API https://en.wikipedia.org/wiki/Application_programming_interface
[8] PyTorch http://pytorch.org/
[9] CPU https://cs.wikipedia.org/wiki/Centrální_procesorová_jednotka
[10] GPU https://cs.wikipedia.org/wiki/GPU
[11] Numpy http://www.numpy.org/
[12] OpenCV http://opencv.org/

OpenFace výrazně ulehčuje práci s identifikací obličejů, jelikož jsou v něm již implementovány algoritmy používané pro identifikaci osob, například pro zarovnání obličeje v rámci jeho výřezu. [2]

Kapitola 7

Postup při detekci a identifikaci obličeje

Detekce a identifikace spolu úzce souvisí. Jelikož webová galerie vytvořená v rámci této práce byla vyvinuta pro fotografie, které zpravidla obsahují dva a více obličejů, je nejdříve nutné tyto obličeje detekovat, oříznout z původní fotografie a na nich poté provést identifikaci. Postup při detekci a identifikaci lze popsat v několika krocích, které dále budou více rozebrány. Základní postup, který je dodržován je následující:

- Ze vstupního obrazu detekovat všechny obličeje
- Pro jednotlivé obličeje detekovat základní rysy (oči, ústa, nos atd.)
- Podle detekovaných rysů zarovnat obličej transformováním obrazu, aby tyto rysy byly stále ve stejné pozici
- Pomocí předtrénované neuronové sítě vygenerovat pro každý obličej vektor příznaků, které ho charakterizují
- Porovnáním vektorů příznaků rozhodnout, zda jsou obličeje shodné či nikoliv

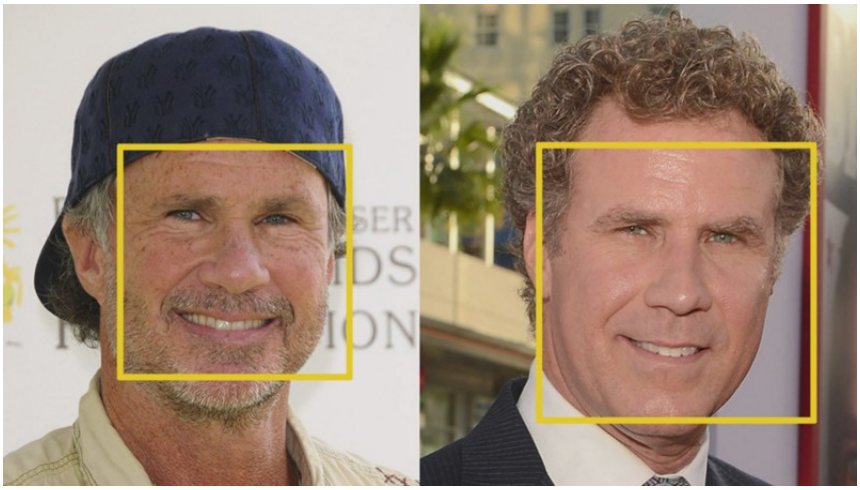
7.1 Detekce obličejů

Pro detekování obličejů v obraze je v prvním kroku nutné načíst samotný obraz, na kterém bude prováděna detekce. To lze provést pomocí libovolné knihovny, pracující s obrázky, například **`scikit-image`**[1]. Ta načte obraz do proměnné, pomocí niž je k obrazu dále přistupováno. Následně je načten detektor z příslušné knihovny. V tomto případě je použita knihovna **`Dlib`**.

Po vložení obrazu do detektoru je spuštěna samotná detekce. Detektor vrací jako výsledek souřadnice všech detekovaných obličejů v obraze. Podle těchto souřadnic lze, za použití již zmíněné knihovny **`scikit-image`**, vyříznout obličej ze vstupního obrazu a pracovat s nim samostatně.

Pro urychlení detekce obličejů je vhodné obraz, který je ve velkém rozlišení, nejprve zmenšit. Výsledek je stejný, ale detekce trvá několikanásobně kratší dobu. Ovšem po zmenšení obrazu je nutné přepočítat výsledné souřadnice, aby odpovídali původní velikosti obrazu.

[1] scikit-image http://scikit-image.org/

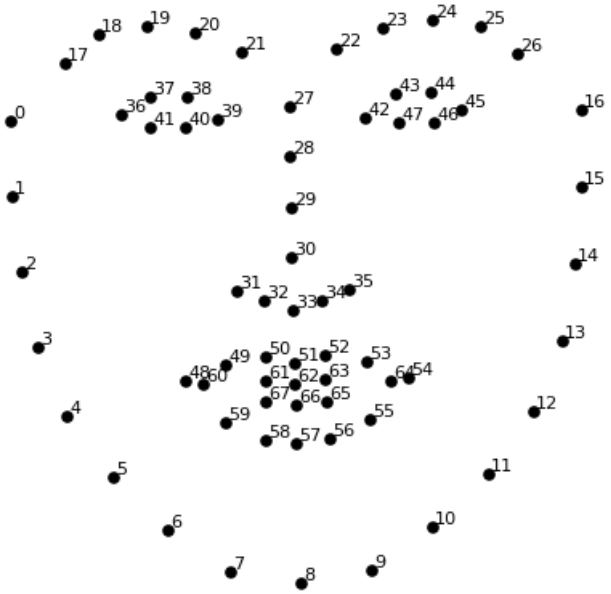


Obrázek 7.1: Detekce obličeje z obrazu, převzato z [12]

7.2 Detekce základních rusů v obličeji

Dále je nutné v obraze detekovat základní rysy obličeje, podle kterých bude obličej zarovnán. K tomu je použit algoritmus *Face landmark estimation*. Základní myšlenkou je najít specifických 68 bodů, zvaných *landmarks*, které jsou obsaženy v každém obličeji (ohraničení oka, linie nosu atd.). Tyto body jsou vidět na obrázku 7.2.

Detekce 68 bodů v obličeji je provedena pomocí stejné knihovny jako při detekci, `Dlib` a předtrénovaného prediktoru.

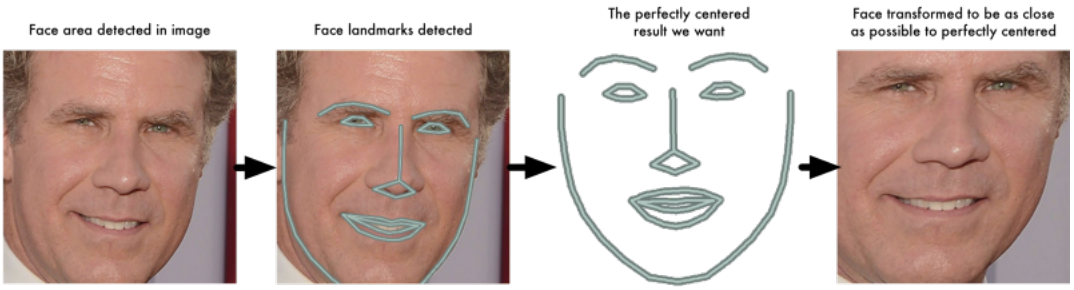


Obrázek 7.2: 68 bodů (*landmarks)* v obličeji, převzato z [12]

7.3 Transformace obrazu

Transformace obrazu je provedena pomocí algoritmu *Affine transformation*[2], jenž je implementován v pluginu OpenFace, který byl použit v této práci.

Pomocí použití algoritmu *Affine transformation* je obličej, nehledě na to jak je natočen, vycentrován v rámci obrazu. To napomáhá samotné identifikaci, aby byla více přesnější, jelikož je každý obličej zarovnán stejně.



Obrázek 7.3: Zarovnání obličeje, převzato z [12]

7.4 Identifikace obličeje

Jakmile je obraz transformován, je vše připraveno k identifikaci. OpenFace implementuje framework Torch, pomocí kterého je vytvořena a natrénována konvoluční neuronová síť pro identifikaci.

Pomocí OpenFace je vložen zarovnaný obličej do načtené neuronové sítě. Výstup ze sítě tvoří 128 dimensionální vektor příznaků, obrázek 7.4. Tento vektor je pro různé vstupy unikátní. Nicméně pro dva různé vstupy se stejným obličejem je tento vektor rozdílný minimálně. Pomocí algoritmů pro porovnání dvou n-dimensionálních vektorů, lze určit do jaké míry se dva vektory liší a na základě této míry je rozhodnuto, zda jsou obličeje různé či totožné.

V této práci byl pro výpočet odlišnosti dvou vektorů použit algotimus pro výpočet *Euklidovy vzdálenosti*[3] dvou vektorů.

[2] Affine transformation https://en.wikipedia.org/wiki/Affine_transformation
[3] Euklidova vzdálenost https://en.wikipedia.org/wiki/Euclidean_distance

Input Image

128 Measurements Generated from Image

0.097496084868908	0.045223236083984	-0.1281466782093	0.032084941864014
0.12529824674129	0.060309179127216	0.17521631717682	0.020976085215807
0.030809439718723	-0.01981477253139	0.10801389068365	-0.00052163278451189
0.036050599068403	0.065554238855839	0.0731306001544	-0.1318951100111
-0.097486883401871	0.1226262897253	-0.029626874253154	-0.0059557510539889
-0.0066401711665094	0.036750309169292	-0.15958009660244	0.043374512344599
-0.14131525158882	0.14114324748516	-0.031351584941149	-0.053343612700701
-0.048540540039539	-0.061901587992907	-0.15042643249035	0.078198105096817
-0.12567175924778	-0.10568545013666	-0.12728653848171	-0.076289616525173
-0.061418771743774	-0.074287034571171	-0.065365232527256	0.12369467318058
0.046741496771574	0.0061761881224811	0.14746543765068	0.056418422609568
-0.12113650143147	-0.21055991947651	0.0041091227903962	0.089727647602558
0.061606746166945	0.11345765739679	0.021352224051952	-0.0085843298584223
0.061989940702915	0.19372203946114	-0.086726233363152	-0.022388197481632
0.10904195904732	0.084853030741215	0.09463594853878	0.020696049556136
-0.019414527341723	0.0064811296761036	0.21180312335491	-0.050584398210049
0.15245945751667	-0.16582328081131	-0.035577941685915	-0.072376452386379
-0.12216668576002	-0.0072777755558491	-0.036901291459799	-0.034365277737379
0.083934605121613	-0.059730969369411	-0.070026844739914	-0.045013956725597
0.087945111095905	0.11478432267904	-0.089621491730213	-0.013955107890069
-0.021407851949334	0.14841195940971	0.078333757817745	-0.17898085713387
-0.018298890441656	0.049525424838066	0.13227833807468	-0.072600327432156
-0.011014151386917	-0.051016297191381	-0.14132921397686	0.0050511928275228
0.00936793349683 28	-0.062812767922878	-0.13407498598099	-0.014829395338893
0.058139257133007	0.0048638740554452	-0.039491076022387	-0.043765489012003
-0.024210374802351	-0.11443792283535	0.071997955441475	-0.012062266469002
-0.057223934680223	0.014683869667351	0.05228154733777	0.012774495407939
0.023535015061498	-0.081752359867096	-0.031709920614958	0.069833360612392
-0.0098039731383324	0.037022035568953	0.11009479314089	0.11638788878918
0.020220354199409	0.12788131833076	0.18632389605045	-0.015336792916059
0.0040337680839002	-0.094398014247417	-0.11768248677254	0.10281457751989
0.051597066223621	-0.10034311562777	-0.040977258235216	-0.082041338086128

Obrázek 7.4: Výstup z neuronové sítě, převzato z [12]

Kapitola 8

Implementace serverové části aplikace

Následující kapitola popisuje implementaci serverové části aplikace. Každá sekce kapitoly popisuje implementaci jedné z částí serveru. Celá serverová část aplikace byla vytvořena v jazyce Python za použití knihoven určených pro tento jazyk.

8.1 Twisted server

Pro vytvoření serveru pomocí Twisted je použita stejnojmená knihovna Twisted, která je volně dostupná.

Při spuštění serveru je vytvořen TCP socket na portu 8888. Pokud je požadován jiný port, je nutné ho ručně změnit v jednotlivých modulech.

Samotný server je potom spuštěn ve dvou krocích. V prvním kroku se k tzv. *reactoru* připojí vytvořený TCP socket na příslušném portu a následně se v dalším kroku *reactor* spustí příkazem `reactor.run()`, což spustí aplikační server.

Při spuštění serveru také proběhne připojení k databázi vytvořené pomocí frameworku SQLite. Spojení s databází je navázáno za použití modulu, z knihovny Twisted, `adbapi`[1]. Dále je složka, obshaující šablony HTML stránek připojena k šablonovacímu systému Jinja2.

8.2 Registrace uživatele

Při žádosti o registraci je na server odeslán požadavek, jehož parametry jsou jednotlivé položky formuláře, které uživatel vyplnil. Kontrola platnosti vyplněných údajů je prováděna na straně klienta, ještě před odesláním požadavku o registraci serveru. Ve chvíli, kdy údaje projdou kontrolou na klientské straně jsou odeslány na server.

Nejprve je nutné ověřit, zda uživatel s daným uživatelským jménem již neexistuje. V případě, že již existuje, odešle se zpět klientské části aplikace odpověď s informací, že daný uživatel je již registrován. V opačném případě se na serveru vytvoří pro nového uživatele složka, která slouží pro ukládání nahraných fotografií, včetně složky, kam se ukládájí ořezy obličejů, které jsou detekovány ve fotografiích. Uživateli se také při registraci vygeneruje unikátní kód, sloužící pro snadné sdílení jeho galerie mezi jeho známé.

V posledním kroku registrace se údaje, vyplněné uživatelem včetně vygenerovaného kódu pro sdílení vloží do databáze.

[1] Adbapi https://twistedmatrix.com/documents/15.3.0/core/howto/rdbms.html

8.3 Přihlášení a odhlášení uživatele

Oproti registraci probíhá kontrola vyplněných údajů v přihlašovacím formuláři pouze na straně serveru. Kontrola je provedena pokusem o vyhledání přihlašovacích údajů v databázi.

Jestliže údaje nejsou validní, odešle se, stejně jako v případě registrace, zpět klientské části aplikace odpověď, ve které je obsažena informace, zda bylo špatně vyplněno uživatelské jméno, heslo popřípadě oba údaje.

Jakmile je uživatelské jméno i heslo vyplněno správně proběhne vytvoření tzv. *session*. Ta je pro každého přihlášeného uživatele unikátní a pomocí této techniky se při přijetí libovolného požadavku na serveru rozlišuje, od kterého uživatele daný požadavek přišel. Jakmile se uživatel, který není přihlášen, tudíž pro něj není vytvořena *session*, pokusí zadat adresu, směrující přímo do galerie, proběhne kontrola *session* s negativním výsledkem a uživatel je automaticky přesměrován na přihlašovací stránku. Nakonec se klientské části odešle zpráva o úspěšném přihlášení a dále je přesměrován přímo do galerie.

Ve chvíli kdy na server příjde požadavek na odhlášení uživatele, tak je nejprve zjištěno, jaký uživatel požadavek odeslal a následně je ukončena *session* daného uživatele a poté je přesměrován na přihlašovací stránku.

8.4 Renderování galerie

Renderování galerie probíhá, stejně jako v případě všech ostatních HTML stránek v aplikaci, za pomoci šablonovacího systému Jinja2.

Nejprve proběhne kontrola, jak již bylo zmíněno na konci sekce 8.3, zda je uživatel přihlášen. Pokud ano, uloží se jeho uživatelské jméno do lokální proměnné pro větší přehlednost kódu. Z databáze se podle uživatelského jména načte seznam fotografií, které uživatel již dříve nahrál na server. V případě, že jde o první přihlášení je tento seznam prázdný a renderuje se prázdná stránka.

Nakonec se z databáze načte i unikatní kód uživatele, který byl vygenerovaný při jeho registraci.

Jakmile jsou všechny data načteny z databáze, uloží se do šablonovacíhc proměnných. Poté proběhne samotné renderování stránky pomocí šablonovacího systému s šablonovacími proměnnými.

8.5 Nahrávání a mazání fotografií

Fotografie jsou na server odesílány v asynchronním reřimu, což znamená, že při nahrávání více fotografií najednou se fotografie odesílají samostatně. Nejprve se načte jméno nahrávaného souboru a následně je načtena samotná fotografie v binární podobě. Binární data jsou v zakódovaná v hexadecimálním tvaru, tudíž je nejdříve nutné tyto data dekódovat zpět do formátu typického pro fotografie (jpg, png aj.).

Co se týče názvu souboru, je nutné ověřit, zda už daný uživatel nemá na serveru uloženou fotografii se stejným nazvem. Pro ověření slouží funkce `ifexist`, které je jako parametr předán název souboru. Jestliže je název unikátní, funkce vrací stjný název. V opačném případě se před název souboru vygeneruje číslice a poté je nový název opět ověřen. Funkce nakonec vrací unikátní název souboru.

Fotografie je poté uložena na serveru pod jedinečným názvem. Pro každou fotografii je také uložen záznam do databáze.

Požadavek na smazání fotografie je odesílán vždy s parametrem obsahujícím název fotografie, která má být odtraněna. Po obdržení žádosti o odstranění fotografie se odstraní samotná fotografie ze serveru, včetně všech ořezů obličejů vytvořených z této fotografie. Následně jsou z databáze odstraněny všechny záznamy spojené s odstraňovanou fotografií.

8.6 Sdílení galerie

Pro umožnění uživateli sdílet galerii mezi své známé bylo nutné vytvořit dvě role uživatelů s odlišnými právy. Uživatel jež se zaregistroval a tím získal práva vytvořit si galerii, musí mít možnost ji spravovat (nahrávat a mazat fotografie). Zatímco uživatel, kterému byla daná galerie nasdílena musí mít pouze právo prohlížet a filtrovat galerii.

Uživateli jenž se zaregistruje, jak již bylo zmíněno v sekci 8.2, je vygenerován unikátní kód, který mu umožňuje nasdílet svoji galerii. Vygenerovaný kód se zobrazuje po kliknutí na příslušné tlačítko umístěné v ovládacím panelu jeho galerie. Tento kód může poslat známému, kterému chce galerii nasdílet.

Jakmile na server příjde žádost o přístup do galerie prostřednictvím kódu, je tento kód ověřen v databázi. Pokud zadaný kód není v databázi uložen, uživatel přistupující do galerie obdrží zprávu, že zadaný kód není validní. V opačném případě je vyrenderovaná stránka podobná té, která se zobrazí registrovanému uživateli po přihlášení, ale s tím rozdílem, že chybí ovládací prvky galerie pro nahrání fotografií, mazání fotografií a sdílení galerie.

8.7 Modul Detection

V modulu Detection je implementována detekce obličejů z fotografií nahraných uživateli. Tento modul je spuštěn jakmile některý z uživatelů dokončil nahrávání fotografií na server. Detekce je prováděna jen na fotografiích, které byly aktuálně nahrané a probíhá zvlášť pro každého uživatele.

V prvním kroku se z databáze načtenou cesty k fotografiím, které mají být zpracovány a poté se uloží do proměnné **`images`** typu **`array`**, z které jsou dále postupně načítány. Následně je načten detektor **`frontal_face_detector`** z knihovny Dlib, který provadí samotnou detekci obličejů. Pomocí cyklu se poté prochází proměnná **`images`**, z které je v každé iteraci cyklu načtena jedna cesta k fotografii.

Podle cesty je načtena samotná fotografie, která se převede do formátu RGB[2]. Pro urychlení detekce obličejů se ověří rozlišní vstupní fotografie. Pokud je některý z parametrů rozlišení (šířka, výška) vetší než 500 pixelů, tak se u vstupní fotografie zmenší rozlišení tak, aby větší rozměr fotografie dosahoval maximální velikosti právě 500 pixelů. Jestliže je proces zmenšení rozlišení proveden, vypočítá se poměr, v kterém byl vstupní obraz zmenšen. Tento poměr je vypočten podělením původního rozměru konstantou 500, což odpovídá novému rozměru. Poměr je uložen do proměnné pro přepočítání souřadnic detekovaných obličejů, aby odpovídali původnímu rozměru fotografie. Po tomto procesu je spuštěna samotná detekce.

Detektor po dokončení detekce vrací souřadnice včech obličejů, které dokázal detekovat z fotografie. Souřadnice jsou postupně procházeny v cyklu. Ze vstupní fotografie se za použití knihovny PIL[3] vyřízne detekovaný obličej , který se uloží do samostatného souboru

[2]RGB https://cs.wikipedia.org/wiki/RGB
[3]PIL http://www.pythonware.com/products/pil/

ve formátu *jpg*. Ořez je v jedné z dalších fází použit pro identifikaci. Jednotlivé souřadnice jsou uloženy do databáze.

S cestou k oříznutému obličeji, jako parametrem je zavolána funkce z modulu *Identification*, která provede identifikaci. Návratová hodnota této funkce je 128 dimenzionální vektor. Jestliže se identifikace nezdaří, například z důvodu chybné detekce, návratová hodnota funkce je prázdný vektor. V tomto případě je ožez, dříve uložený na serveru, smazán. Vektor, navrácený funkcí pro identifikaci, je uložen do souboru *Identification.csv*, jehož formát je popsán v sekci 8.9.

8.8 Modul Identification

Pro identifikaci obličeje je použita knihovna OpenFace. Pomocí této knihovny je načtena neuronová síť natrénovaná právě pro identifikaci obličejů. Modul obsahuje jedinou funkci, volanou z modulu Detection. Této funkci je, jako parametr, předána cesta k ořezu obličeje, který má být identifikován.

Nejprve je soubor obsahující obličej načten a převeden do formátu RGB. Soubor ve správném formátu je vložen do funkce, jako parametr, která v obličeji detekuje základní rysy (oči, obočí, ústa, nos aj.) a poté obličej zarovná způsobem, aby byl vycentrovaný. Zarovnání je velmi důležité pro správnou identifikaci obličeje, jelikož v případě, kdy by byl stejný obličej v jiné pozici, v rámci ořezu, může neuronová síť vrátit odlišné hodnoty, což by mohlo vest k závěru, že obličeje jsou různé.

V poslední fázi funkce je zarovnaný obličej vložen do předtrénované neuronové sítě. Návratová hodnota funkce je výstup z neuronové sítě, což je 128 dimensionální vektor obsahující příznaky pro daný obličej.

8.9 Soubor Identification.csv

Do souboru *Identification.csv* jsou ukládány cesty k souborům, jenž obsahují ořezy obličejů a k nim jsou přiřazené 128 dimensionální vektory vzniklé jako výstup z neuronové sítě.

Soubor csv je souborový formát určený pro výměnu tabulkových dat. Je tvořen řádky, jehož položky jsou odděleny znakem čárka (,). V každém řádku tohoto souboru jsou následující položky:

- obličej - cesta k ořezu obličeje, vytvořeného během detekce
- fotografie - cesta k fotografii, z které byl daný ořez vytvořen
- identifikátor - 128 dimensionální vektor pro daný obličej navrácený neuronovou sítí, tento vektor byl byl převeden do textového formátu pomocí knihovny NumPy

8.10 Třídění fotografií podle obličejů

Při třídění fotografií je na serveru přijat požadavek, jehož parametr obsahuje souřadnice obličejů, podle kterých se mají fotografie filtrovat, název fotografií, na kterých jsou zvolené obličeje a také identifikátor, který rozlišuje zvolený způsob filtrování fotografií. V databázi je podle přijatých souřadnic vyhledána cesta k ořezu obličejů, podle nichž má být provedeno filtrování.

Filtrování fotografií může být provedeno dvěma způsoby, mezi kterými si uživatel sám vybírá.

Prvním způsobem se filtrují fotografie, na nichž jsou najednou všechny osoby zvolené uživatelem. V tomto případě se nejprve ze souboru *Identification.csv* načte 128 dimensionální vektor pro první z filtrovaných obličejů. Provede se první průchod souborem *Identification.csv*, z kterého jsou postupně načítány všechny vektory a porovnávány, pomocí euklidovy vzdálenosti, s vektorem pro první filrovaný obličej. Pokud je rozdíl menší, než stanovená hranice, lze konstatovat, že se jedná o stejný obličej. V tom případě se cesta k fotografii, na které je shodný obličej, uloží do proměnné **`images`** typu **`array`**. Po prvním průchodu jsou tedy v**`images`** uloženy cesty ke všem fotografiím, na kterých je detekován obličej shodný s prvním z filtrovaných. V případě, že uživatel zvolil více obličejů pro filtrování, prochází se soubor *Identification.csv* znovu. Počet průchodů je roven počtu obličejů, podle kterých se má galerie filtrovat. Na začátku každého průchodu se **`images`** zkopíruje do pomocné proměnné a původní proměnná **`images`** se vyprázdní. V opakovaných průchodech se však již nekontrolují všechny vektory uložené v souboru, ale pouze ty, u kterých je cesta k fotografii obsahující daný obličej, shodná s některou z cest v pomocné proměnné. Jestliže se porovnávané obličeje posoudí jako shodné, cesta k fotografii je opět uložena do **`images`**. Opakovanými průchody se tedy postupně třídí uložené cesty v **`images`** a po všech průchodech zůstanou v **`images`** pouze cesty k fotografiím, které obsahují všechny obličeje, podle nichž se filtrovalo.

Druhým způsobem jsou vyfiltrovány fotografie, na nichž je alespoň jeden ze zvolených obličejů. Tento způsob je velice podobný prvnímu způsobu filtrování. Jediný rozdíl je v tom, že každý vybraný obličej je porovnáván se všemi obličeji v souboru *Identification.csv* a ne pouze s těmi, jež jsou již uloženy v proměnné **`images`**. Po dokončení filtrování obsahuje proměnná **`images`** všechny fotografie, na nichž je alepoň jeden z vybraných obličejů.

Jakmile je filtrování dokončeno, proběhne renderování stránky stejným způsobem, jak již bylo popsáno v sekci 8.4.

Kapitola 9

Implementace klientské části aplikace

Tato kapitola vysvětluje implementaci stejných částí jako v předchozí kapitole, ale tentokrát je popsána implementace na klientské straně aplikace. Klientská část je tvořena pomocí HTML, JavaScriptu, Ajaxu, Bootstrapu a CSS.

9.1 Registrace, přihlášení a odhlášení uživatele

Pro registraci a přihlášení slouží formuláře, do kterých uživatel vyplňuje potřebné údaje. Tyto formuláře se uživateli zobrazí v případě, že zadal webovou adresu směrující k této aplikaci a nebyl v nejbližší době přihlášen nebo při odhlášení uživatele. K přihlášení je nutné zadat uživatelské jméno a heslo. U registrace je to uživatelské jméno, email a heslo, které je nutné zadat dvakrát pro potvrzení.

Na přihlašovacím formuláři je také umístěno tlačítko *Enter to gallery by code*, které slouží pro vstup do galerie pomocí kódu a pouze v režimu prohlížení. Blíže je režim prohlížení a vstup pomocí kódu popsán v sekci 9.4.

Formuláře jsou vytvořeny pomocí HTML, pro samotné vytvoření elementu, CSS a Bootstrapu pro stylizaci a pomocí JavaScriptu jsou vytvořeny animace při přechodu mezi formuláři.

Kontrola údajů, vyplněných při registraci je narozdíl od přihlašovacích údajů prováděna ještě před odesláním dat na server na straně klienta. Jediný údaj je nutný zkontrolovat i na serverové straně aplikace a tím je uživatelské jméno. U tohohto údaje se na klientské straně kontroluje pouze to, zda bylo zadané. Emailová adresa je testována za použití regulárního výrazu. Ten provede kontrolu zda zadaná adresa obsahuje pouze povolené znaky v *místní* části (část před znakem '@'), v *internetové doméně* (část za znakem '@') a zda obshuje znak '@'. Co se týče hesla, nejsou na něj kladený žádné požadavky, tudíž záleží na uživateli, jak silné heslo si zvolí. U hesla se kontroluje pouze zda bylo zadáno a zda zadané heslo a kontrolní heslo je totožné. Kontroly jsou prováděny postupně v pořádí, v jakém byly popsány v tomto odstavci. Tudíž uživatel je postupně upozorňován na chybné údaje a až v případě, že daný údaj je validní, provádí se kontrola dalšího údaje.

Ve fázi, kdy jsou všechny položky formuláře správně vyplněny proběhne odeslání vyplněných dat na server. Odesílání je provedeno, při přihlašovaní i registraci, za pomoci technologie Ajax na adresu serveru, na které jsou vyřizovány žádosti o přihlášení či registraci.

Jakmile proběhne příhlášení úspěšně, uživatel je přesměrován na stránku obsahující jeho galerii fotografií. V případě úspěšné registrace se provede přesměrování na přihlašovací formulář, kde se uživatel může poprvé přihlásit.

Odhlášení je provedeno po stisknutí tlačítka *LogOut* na ovládacím panelu galerie. Na server se odešle žadost o odhlášení uživatele. Po vyřízení žádosti o odhlášení je uživatel přesměrován na přihlašovací formulář.

9.2 Šablonovací systém

Šablonovací systém je využit zejména při renderování stránky obsahující galerii. Jelikož není předem známo kolik fotografií si uživatel nahraje na server, je nutné přidávat elementy, které budou obsahovat fotografie dynamicky podle reálného počtu fotografií.

Další části, kdy je využito šablonovacího systému je například při zobrazení uživatelského jména přihlašeného uživatele, kdy ho server vloží do stránky dynamicky jako proměnnou pomocí šablonovacího systému a to se při renderování stránky přeloží na skutečné přihlašovací jméno.

9.3 Galerie

Galerie fotografií tvoří hlavní část klientské části aplikace. Obsahuje ovládací panel, hlavní okno, ve kterém je zobrazena aktuální fotografie a také tzv. *filmový pás* s fotografiemi, což jsou v řadě uspořádané miniatury fotografií, které umožňují pohodlnější manipulaci s galerií.

Během prvního přihlášení uživatele nebo po přihlášení uživatele, který si zatím nenahrál žádné fotografie, je zobrazeno na stránce modální okno, v němž je tlačítko pro přeměrování na formulář pro nahrání fotografií.

Ovládací panel obshauje následující položky:

- Home - znovunačtení galerie, hlavní využití v případě, že galerie již byla filtrována a uživatel chce zobrazit galerii bez filtrování
- Upload images - přesměrování na formulář pro nahrávání fotografií
- Show detected faces - zobrazení obdelníků ohraničující obličeje na fotografii
- Delete image - smazání aktuální fotografie
- Share via code - zobrazení kódu, pomocí kterého lze galerii sdílet
- Uživatelské jméno - textový element zobrazující uživatelské jméno přihlášeného uživatele
- LogOut - odhlášení uživatele

Hlavní částí stránky je okno, v nemž se zobrazuje aktuální fotografie. Tento element je pomocí stylů nastaven tak, aby se jeho velikost adekvátně přizpůsobovala k velikosti okna webového prohlížeče. Každá fotografie tvoří samostatný element. Tyto elementy se vzájemně překrývají a mění se pouze jejich viditelnost podle toho, zda mají být momentálně zobrazeny. Po stranách tohoto okna jsou navigační šipky sloužící pro změnu zobrazené fotografie.

Změnit zobrazenou fotografii je možné také uchopením fotografie pomocí myši a přetažením buďto doleva (zobrazí se následující fotografie) nebo doprava (zobrazí se předchozí).

Pod hlavním oknem se nachází již zmíněný pás miniatur fotografií. Do tohoto pásu se fotografie vkládají stejným způsobem jako do hlavního okna. S pásem lze manipulovat stejně jako s hlavním oknem. Oproti hlavnímu oknu je zde přidaná pouze jedna funkce navíc a to ta, že po kliknutí na některou z miniatur fotografie se tato fotografie zobrazí v hlavním okně.

Animace hlavního okna i pásu miniatur jsou vytvořeny pomocí JavaScriptu.

9.4 Sdílení galerie

Tato aplikace umožňuje uživateli sdílení galerie mezi své známé. Jak již bylo popsáno v kapitole 8, děje se tak za použití unikátního kódu, který byl uživateli vygenerován během jeho registrace. Po stiskutí tlačítka *Share via code* se zobrazí modální okno s vygenerováným kódem. Uživatel tak má možnost tento kód zkopírovat a rozeslat lidem, s kterými chce svoji galerii sdílet.

Jak již bylo řečeno v sekci 9.1, na formuláři, který slouží pro přihlašovaní je také umístěno tlačítko *Enter code* sloužící pro vstup uživatele prostřednictvím kódu. Po stisknutí tohoto tlačítka se zobrazí formulář, do kterého má uživatel možnost vložit kód, který od někoho obdržel. Potvrzením kódu se odešle zadaný kód na server, kde proběhne ověření kódu. Jestliže je kód správný je uživatel přesměrován do galerie v režimu prohlížení. Tento režim dovoluje pouze prohlížet fotografie a třídit je podle obličejů. Ostatní prvky, které se nachází v ovládacím panelu uživatele, jež je registrován, jsou skryty.

9.5 Nahrávání fotografií do galerie

Exituje již mnoho šablon a pluginů[1], které jsou volně dostupné a slouží právě pro účely nahrávání souborů na server prostřednictvím technologie Ajax. Pro účely této práce byl použit právě jednen z volně dostupných pluginů. Jedná se o plugin *Krajee* vytvořený pomocí knihovny jazyka JavaScript, jQuery[2]. Tento plugin umožňuje nahrávat soubory libovolného formátu za použití technologie Ajax a to jak v synchronním, tak v asynchronním režimu. Synchronní režim nahrává soubory najednou jako sled bitů, zatímco asynchronní režim nahrává soubory odděleně.

Soubory je možné do formuláře vložit dvěma způsoby. Pomocí Drag and Drop[3], kdy je možné vložit soubory přetažením do příslušného místa. Druhá možnost je stisknutí tlačítka *Browse*, které otevře tzv. *open dialog*. Jsou povolené pouze formáty souborů typické pro obrázky (jpg, png).

Po vybrání fotografií se každá z nich zobrazí jako miniatura na stránce. U každé miniatury je stavový řádek informující uživatele o stavu nahrávání dané fotografie a ovládácí panel, umožňující nahrát pouze danou fotografii nebo ji odebrat z nahrávacího formuláře.

Nahrávání fotografií je spuštěno po stisknutí tlačítka *Upload*. Po jeho stisknutí se nahrají všechny fotografie vložené do formuláře. Ve spodní části formuláře se nachází stavový řádek, informující o celkovém stavu nahrávání fotografií.

Po dokončení nahrávání je uživatel přesměrován zpět do galerie.

[1] Plugin https://cs.wikipedia.org/wiki/Plugin
[2] jQuery https://cs.wikipedia.org/wiki/JQuery
[3] Drag and Drop https://cs.wikipedia.org/wiki/Drag_and_drop

9.6 Zobrazení detekovaných obličejů

K zobrazení detekovaných obličejů slouží checkbox[4] *Show detected faces*. Po jeho zaškrtnutí se odešle žádost o zaslání souřadnic detekovaných obličejů, pro aktuální fotografii, na server. Jakmile jsou tyto souřadnice obdrženy, vykreslí se podle nich obdélníky, které ohraničují obličeje na fotografii. Při každé změně aktuální fotografie se znovu odešle požadavek na server o zaslání souřadnic obličejů pro aktuální fotografii.

Vykreslení těchto obdélníků obstarává HTML element **`canvas`**[5]. Tento element je nastaven tak, aby měnil svoji velikost v závislosti na fotografii, kterou překrývá. **`Canvas`** umožňuje vykreslování různých geometrických útvarů, stačí zadat o jaký útvar se jedná, souřadnice kde má být jeho začátek a nakonec šířka a výška útvaru. Tyto parametry jsou v případě této aplikace vypočteny ze souřadnic obdržených ze serveru a poté je podle nich na **`canvas`** vykreslen obdélník ohraničující obličej.

Jakmile uživatel mění velikost okna prohlížeče, je nutné překreslit i obdélníky, aby byly správně umístěny v rámci fotografie.

9.7 Filtrování fotografií podle obličeje

Filtrování fotografií podle obličejů tvoří nejdůležitější prvek vytvářené aplikace. Uživatel si pouze zvolí obličeje, podle kterých má být galerie filtrována a následně se automaticky, za pomoci identifikace obličejů, galerie vyfiltruje.

Možnost filtrovat fotografie má uživatel v situaci, kdy si zobrazí detekované obličeje. Při zobrazení detekovaných obličejů totiž může na obličej kliknout a tím ho přidat do výběru pro filtrování. Po kliknutí na první obličej je zobrazeno modální okna skrz které, může uživatel zapnout filtrovaní. Počet zvolených obličejů je libovolný stejně jako množství fotografií, z kterých jsou obličeje vybírány. Opětovným kliknutím na již vybraný obličej ho odstraní z výběru pro filtrování.

Uživatel má také možnost si, skrz zobrazené modální okno, zvolit způsob filtrování fotografií. Jedna z možností je, že zvolené osoby musí být na stejné fotografii. Podle druhé možnosti jsou fotografie vyfiltrovány tak, že na fotografii musí být alespoň jedna ze zvolených osob.

Jakmile uživatel potvrdí filtraci některým ze dvou způsobů, odešle se na server seznam obličejů (jejich souřadnic společně s názvem fotografie, na které se nachází) a identifikátor rozlišující způsob filtrování. Na serveru proběhne samotné vyfiltrování fotek způsobem popsaným v sekci 8.10. Po dokončení filtrace na serveru se uživateli přenačte stránka a jsou zobrazeny pouze fotografie, které prošly filtrováním.

Zpět do galerie, jenž není filtrovaná se uživatel dostane stisknutím tlačítka *Home*, umístěným v ovládacím panelu galerie.

[4]Checkbox https://www.jakpsatweb.cz/enc/checkbox.html
[5]Canvas https://www.w3schools.com/html/html5_canvas.asp

Kapitola 10

Spuštění aplikace

V následující kapitole je popsán postup pro spuštění webové aplikace.

10.1 Spuštění webového serveru

Webová aplikace byla vytvořena pro operační systém Linux tudíž její spuštění je možné pouze v tomto operačním systému.

Hlavní část aplikace tvoří aplikační server. Ten je spouštěn za použití konzole. Aplikace pro svůj běh používá jazyk Python 2.7.13, na němž byla i testována, nicméně měla by být kompatibilní i s novějšími verzemi jazyka.

Kromě standardních knihoven, jež jsou součástí jazyka Python ihned po instalaci jsou nutné knihovny `NumPy`, `PIL`, `scikit-image`, `collections`, `sqlite3`, `Twisted`, `Zope`, `Dlib`, `OpenCV`, `OpenFace`, `Jinja2` a `Torch7`.

Script, který spouští chod celé aplikace, je spouštěn jako modul knihovny. Z kořenového adresáře aplikace je aplikace spuštěna příkazem: "`python2 -m imggallery.server &`". Aplikace běží standardně na portu 8888. Pro jeho změnu je nutné změnit číslo portu v jednotlivých modulech.

10.2 Spuštění klientské části

Pro běh aplikace na straně klienta jsou vyžadovány moderní webové prohlížeče v jejich novějších verzích a připojení k internetu, aby byla zajištěna podpora Bootstrapu a dalších pluginů a knihoven.

Do webového prohlížeče stačí zadat, jako webovou adresu, "localhost:číslo_portu (standardně 8888)"tedy "localhost:8888". Pro účel prezentace aplikace byl vytvořen ukázkový uživatelský účet s přihlšovacími údaji:

- login: user
- password: userpassword

Kapitola 11

Vyhodnocení detekce a identifikace

Tato kapitola se zabývá vyhodnocením úspěšnosti detekce obličejů a následně úspěšností její identifikace. Nejprve je však nutné popsat dataset fotografií, na kterých vyhodnocení probíhalo. Jsou zde popsány algoritmy použité pro vyhodnocení a následně samotné výsledky vyhodnocení.

11.1 Dataset

Dataset pro účely testování úspěšnosti detekce a identifikace je složen z fotografií, jenž se co nejvíce přibližují fotografiím, které budou v aplikaci používány. Bylo vycházeno z faktu, že aplikace je vyvíjena primárně pro klienty jimž fotografie vytváří sám autor práce. Velká většina fotografií v datasetu jsou svatební fotografie, na nichž jsou dvě a více osob. Na zbylých fotografiích je hudební kapela mající čtyři členy. Všechny fotografie byly použity se souhlasem osob, které se na nich nachází.

Dataset obsahuje fotografie ve vysokém rozlišení (přibližně *5000x3200* pixelů, případně *3200x5000* pixelů u fotografií, které jsou orientovány na výšku). Počet fotografií, obličejů a identit je shrnut v tabulce 11.1.

Tabulka 11.1: Dataset fotografií

Počet fotografií	**Počet obličejů**	**Počet jedinečných identit**
342	1618	425

11.2 Vyhodnocení detekce

Pro účely vyhodnocení detekce bylo nutné pro každý obličej z datasetu fotografií zjistit tzv. *ground-truth bounding box* (v textu bude dále označován zkratkou *GTBB*). GTBB jsou skutečné souřadnice objektu na fotografii, v tomto případě obličeje. GTBB pro každý obličej byl zjišťován manuálně za pomoci editoru fotografií, který dokáže zobrazit souřadnice místa, kde se nachází kurzor v rámci fotografie. Tyto souřadnice byly uloženy do souboru, včetně názvu fotografie, na niž byly zjištěny. Z tohoto souboru jsou načítány data v průběhu vyhodnocování detekce.

Jak již bylo popsáno v sekci 8.7 detektor obličejů vrací souřadnice obličejů (Bounding Boxes - zkáceně BB), které dokázal detekovat. Princip vyhodnocení detekce je tedy v tom porovnat GTBB s BB, které vrátil detektor a na základě toho rozhodnout, zda byl obličej detekován správně či nikoliv. Pro vyhodnocení, zda byl obličej detekován správně je použit algoritmus *Intersection over Union*.

Intersection over Union zkráceně IoU je algoritmus, který porovnává GTBB s BB na základě toho, jak se vzájemně překrývají. IoU lze zapsat pomocí vztahu, který je na obrázku 11.1. Na obrázku 11.2 je vidět již vypočítaná hodnota IoU v závislosti na míře překrytí GTBB s BB.

Před začátkem samotného vyhodnocení je nedříve načten soubor, obsahující GTBB. Pro každou fotografii je vytvořena proměnná `GTBB` datového typu `dictionary`. Jako klíč je použit název fotografie. Hodnoty poté tvoří jednotlivé GTBB pro danou fotografii.

Poté je spuštěna detekce. Detekované BB jsou ukládány do proměnných `BB`, které mají stejnou datovou strukturu jako v případě ukládání GTBB. Jakmile je dokončena detekce na celém datasetu, spustí se samotné vyhodnocení.

Během vyhodnocení se postupně prochází proměnné `GTBB` a `BB`. Postupně jsou, pomocí cyklu, procházeny všechny GTBB a BB pro danou fotografii a pro každou dvojici je vypočtena hodnota IoU. Z vypočtených hodnot se zaznamenávají pouze ty, které jsou z intervalu <0;1>, jelikož hodnoty mimo tento interval byly vypočteny pro dvojici GTBB, BB, jež se ani z části nepřekrývají. Počet správně detekovaných obličejů se ukládá do pomocné proměnné. Při výpočtu úspěšnosti detkce, pro jednu fotografii, v procentech, mohou nastat tyto dvě situace

- počet GTBB je větší nebo roven počtu BB
- počet GTBB je menší než počet BB

V prvním případě se procentuální úspěšnost vyjadří vztahem:

$$x = \frac{numGTBB}{numRD} \tag{11.1}$$

,kde:

- x je úspěšnost detekce v %
- $numGTBB$ je počet GTBB pro danou fotografii
- $numRD$ je počet správných detekcí pro danou fotografii

V druhém případě se procentuální úspěšnost vyjadří vztahem:

$$x = \frac{numBB}{numRD} \tag{11.2}$$

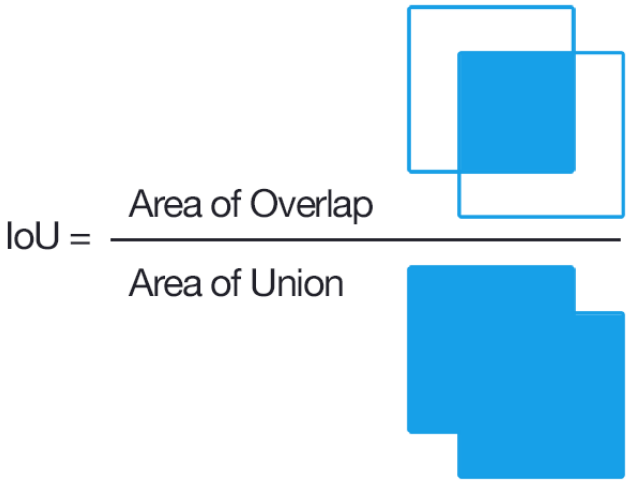
,kde:

- x je úspěšnost detekce v %
- $numGTBB$ je počet BB detekovaných pro danou fotografii
- $numRD$ je počet správných detekcí pro danou fotografii

11.2.1 Shrnutí

Z jednotlivých úspěšností je nakonec vypočten aritmetický průměr, aby byla vyjádřena úspěšnost detektoru na celém datasetu fotografií.

Výsledek vyhodnocení úspěšnosti detekce je shrnut v tabulce 11.2.



Obrázek 11.1: Intersection over Union, převzato z [22]

Obrázek 11.2: Příklady IoU, převzato z [22]

Tabulka 11.2: Úspěšnost detekce na celém datasetu

Počet obličejů	Počet správných detekcí	Úspěšnost v %
1618	1392	86,03

11.3 Vyhodnocení identifikace

K provedení vyhodnocení identifikace obličejů na vlastním datasetu fotografií, je nutné doplnit tento dataset identitami osob, které se nachází na daných fotografiích. Tyto identity

jsou uloženy ve stejném souboru, jako GTBB. Vždy k příslušnému *ground-truth bounding boxu* je doplněno jméno a příjmení osoby již patří daný obličej.

Pro vyhodocení identifikace byla použita metodika, jenž se používá pro vyhodnocování úspěšnosti klasifikátoru. Výsledek identifikace se zařadí do jedné z následujicích tříd:

- True Positive (TP) - Identifikátor vyhodnotí, že dva obličeje jsou *stejné* a ve skutečnosti tomu tak *je*
- False Positive (FP) - Identifikátor vyhodnotí, že dva obličeje jsou *stejné* a ve skutečnosti tomu tak *není*
- True Negative (TN) - Identifikátor vyhodnotí, že dva obličeje jsou *různé* a ve skutečnosti tomu tak *je*
- False Negative (FN) - Identifikátor vyhodnotí, že dva obličeje jsou *různé* a ve skutečnosti tomu tak *není*

Jakým způsobem je rozhodnuto, že dva obličeje jsou různé nebo shodné je popsáno v sekci 8.10. Po zařazení všech výsledků identifikátoru pro testované obličeje do některé z tříd je možné vypočítat *True Positive Rate - TPR* a *False Positive Rate - TPR*. Ty poté slouží k výpočtu úspěšnosti identifikace za použití metriky *Precision and Recall*[1]. *TPR* a *FPR* jsou vypočteny ze vztahů:

$$TPR = \frac{\sum TP}{\sum TP + \sum FN} \tag{11.3}$$

$$FPR = \frac{\sum FP}{\sum FP + \sum TN} \tag{11.4}$$

Precision and Recall je metrika úspěšnosti identifikace, z niž *Precision* značí, v kolika procentech je vyhodnocení identifikátoru, že jsou obličeje stejné, spravné. Výpočet *Precision* je následující:

$$Precision = \frac{\sum TP}{\sum TP + \sum FP} \tag{11.5}$$

Recall značí, kolik procent stejných obličejů bylo skutečně nalezeno a vyhodnoceno správně. Vypočte se pomocí vztahu:

$$Recall = \frac{\sum TP}{\sum TP + \sum FN} \tag{11.6}$$

Pro vyjádření celkové úspěšnosti identifikace byla vypočítána celková přesnot identifikátoru pomocí vztahu:

$$Accuracy = \frac{\sum TP + \sum TN}{\sum TP + \sum TN + \sum FP + \sum FN} \tag{11.7}$$

Výsledky těchto výpočtů jsou shrnuty v tabulce 11.3.

Automatický script pro vyhodnocení identifikace postupuje v těchto krocích:

1. Pro každý obličej z datasetu je za použití neuronové sítě vypočítán 128 dimensionální vektor příznaků

[1] Precision and Recall https://en.wikipedia.org/wiki/Precision_and_recall

2. Jednotlivé příznaky se uloží do souboru *Identification.csv*, který je popsán v sekci 8.9
3. Procházení souboru, pomocí dvou vnořených cyklů, paralelně. V prvním cyklu se postupně načítají vektory příznaků a ty jsou porovnávány se všemi vektory příznaků, následujícími za načteným vektorem. Načítání následujících vektorů zajišťuje druhý zanořený cyklus
4. Výsledek porovnání každé z dvojic obličejů, podle nastavené hranice, je zařazen do jedné z tříd *TP*, *FP*, *TN*, *FN*
5. Zvýší se počítadlo pro danou třídu
6. Vypočtení *TPR* a *FPR*
7. Uložení hodnot do proměnných pro pozdější použití
8. Zvýšení hodnoty prahu a opakování od kroku 3, dokud práh nedosáhl maximální hodnoty

Tabulka 11.3: Úspěšnost identifikace na celém datasetu

Precision v %	**Recall v %**	**Accuracy v %**
90,1	83,8	88,6

Kapitola 12

Testování uživatelského rozhraní

Testování aplikace a jejího rozhraní bylo prováděno na 10 klientech, s kterými jsem již dříve spolupracoval a vytvářel pro ně fotografie, které byly i součástí testovacího datasetu. Testování probíhalo za mého osobního dohledu, kdy testovaným osobám byly kladeny úkoly, jenž měly ověřit intuitivnost a složitost ovládání.

12.1 Úkoly pro testování

Úkoly byly uživatelům zadávány postupně a vždy ve stejném pořadí. Pořadí úkolů bylo stanoveno s ohledem na otestování všech funkcí aplikace. Úkoly byly následující:

1. Vytvořte si uživatelský účet a proveďte první přihlášení
2. Nahrajte 10 fotografií do galerie
3. Prohlédněte si všechny fotografie
4. 2 z fotografií odtraňte
5. Zobrazte detekované obličeje
6. Vyfiltrujte fotografie podle několika libovolných osob a to libovolným způsobem
7. Vraťte se zpět do nevyfiltrované galerie
8. Sdílejte svoji galerii
9. Odhlašte se
10. Pomocí předem připraveného kódu vstupte do sdílené galerie
11. Vyfiltrujte fotografie podle několika libovolných osob a to libovolným způsobem

12.2 Výsledky testování po splnění úkolů

Registrace a přihlášení nečinila nikomu z uživatelů, podle předpokladů, žádné potíže. Tento prvek je totožný s většinou aplikací a stránek jenž vyžadují registraci a přihlášení, tudíž s ním nikdo němel problémy.

Nahrávání fotografií pro nově registrované registrované uživatele je možné dvěma způsoby. Pomocí přímého odkazu v modálním okně, které je zobrazeno pouze v případě, že uživatel nemá nahrané žádné fotografie, nebo za použití ovládacího panelu. 8 uživatelů použilo odkaz v modálním okně. Zbylí 2 uživatelé použili ovládací panel. Pro nahrání fotografií použilo 9 uživatelů tlačítko *Browse* a pouze 1 využil funkce drag-and-drop. Všichni tento úkol zvládli opět bez potíží.

K prohlížení fotografií použili všichi z uživatelů pouze ovládací směrové šipky na stranách hlavního okna. Žádný z nich nepoužil fukcni přetažení fotografie pomocí myši, navigační šipky po stranách pásu miniatur ani kliknutí na některou z miniatur.

Během odstraňování fotografií si 2 uživatelé nebyly jisti, zda bude odstraněna aktuální fotografie, jelikož očekávali systém, kdy budou vybírat fotografie pro odstranění pomocí nějakého formuláře.

Zobrazení detekovaných obličejů a následně filtrování fotografií nečinilo nikomu z testovaných opět žádné potíže. K navrácení zpět do nevyfiltrované galerie použilo 5 uživatelů příslušné tlačítko v ovládacím panelu a 5 použilo tlačítko zpět v ovládacím panelu webového prohlížeče.

Zbylé úkoly zvládli všichni bez problémů.

12.3 Zhodnocení výsledků

Po testování vybranými uživateli se prokázalo, že uživatelské rozhraní ve vytvořené aplikaci má intuitivní ovládání, jelikož uživatelé při plnění úkolů, kdy měli využít prvky aplikace, neměli žádné problémy.

Během prohlížení fotografií byl využíván pouze jeden způsob procházení fotografiemi což vede k úvaze zda je potřeba mít implementovány i ostatní způsoby procházení galerie.

Při plnění úkolu týkajícího se odstranění fotografií se objevilo potencionální rozšíření aplikace. Jedná se o hromadné mazání fotografií pomocí výběru fotografií, které mají být odstraněny.

Filtrování a sdílení galerie se, podle reakce testovaných uživatelů, shledalo s velkým zaujetím. Všichni testovaní uživatelé se shodli, že vytvořená aplikace ulehčuje výběr fotografií pro vyvolání, což lze považovat jako úspěch, jelikož právě pro tento účel byla aplikace vyvíjena.

Kapitola 13

Závěr

Během řešení této práce jsem se seznámil s mnoha nástroji vhodnými pro tvorbu webových aplikací. Prostudoval jsem jejich přednosti a oblast využití, abych posléze mohl zvolit pro použití právě ty nástroje, které se pro tvorbu webové galerie hodí nejlépe. Bylo tedy nutné si také zvolit, pro koho bude výsledný produkt určen a jaké využití bude mít.

Nedílnou součástí práce tvořila manipulace s fotografiemi osob. Kokrétně detekce a identifikace obličejů. Při zkoumání této problematiky jsem vyhledal nejpoužívanější algoritmy pro řešení detekce obličejů obraze. Zjišťoval jsem na jakém principu fungují, jakou mají úspěšnost a zda jsou vhodné pro druh fotografií, s kterými vyvíjená aplikace bude pracovat. Následně jsem zkoumal nejpoužívanější nástroje, pomocí kterých je možné provádět detekci obličejů v obraze. Z těchto nástrojů byly vyselektovány pouze ty nástroje, které měli podporu jazyka **`Python`**, jelikož v něm byla vyvíjena celá aplikace. Výběr konkrétního nástroje ovlivňovala jeho úspěšnost.

Po procesu detekce obličejů jsem se seznamoval s jedním odvětvím strojového učení, neuronovými sítěmi. Pro pochopení této problematiky bylo nutné začít studijem jejich základních vlastností, principů, na kterých fungují, vyhledat nástroje, které s nimi dokáží pracovat a v neposlední řadě provádět různé experimenty.

Jakmile byly vybrány všechny nástroje, které jsou použity pro vývoj výsledné aplikace, přišla na řadu samotná implementace. Nejprve byly implementovány moduly, jenž zpracovávají fotografie. v první řadě proběhla implementace detektoru fotografií, který byl na části datasetu otestován, aby se potvrdilo, že funguje spravně a s dostatečnou přesností. Stejný postup byl aplikován i v případě identifikace osob.

Pro vyhodnocení, s jakou úspěšností pracuje detektor a identifikace obličejů, bylo nutné vytvořit testovací dataset fotografií, které se co nejvíce podobají fotografiím, pro něž je aplikace vyvíjena. Dataset byl vytvořen z vlastních fotografií, které jsem vytvořil již v minulosti pro svoje klienty. S jejich pomocí jsem získal, pro každý obličej na fotografii jeho identitu.

Detektor na vytvořeném datasetu pracuje s úspěšnosti *86,03 %*, což lze považovat za dostatečnou úspěšnost. Identifikace obličejů byla vyhodnocena za použití *Precision and Recall* metriky i vypočtením celkové úspěšnosti v procentech. Identifikace proběhne úspěšně v 88,6 % případů.

Samotný server a klientská část byly vyvíjeny současně, aby bylo možné průběžné testování funkčnosti aplikace.

Testování uživatelského rozhraní prokázalo, že ovládání aplikace je intuitivní a není nijak složité. Po některých testech je však nutné zvážit, zda je potřeba mít implementováno více způsobu procházení galerie. Jelikož testovaní uživatelé využívali pouze jeden z nich.

V budoucnu by bylo vhodné implementování možnosti hromadného mazaní fotografií, což vyplynulo z prováděných testů. Další z možných plánů v budoucím vývoji aplikace je možnost propojení aplikace s účtem Google, vytváření a třídění fotografií do jednotlivých alb, možnost přejmenování fotografie, napojení aplikace na externí úložiště kvůli snížení zatíženosti serveru a upravení rozhraní pro výběr obličejů pro filtrovaní způsobem, že v některé z částí stránky by byly miniatury zvolených obličejů.

Literatura

[1] How are filters and activation maps connected in Convolutional Neural Networks? Nov 9 '15 at 8:21.
URL https://stats.stackexchange.com/questions/180850/how-are-filters-and-activation-maps-connected-in-convolutional-neural-networks

[2] Amos, B.; Ludwiczuk, B.; Satyanarayanan, M.: OpenFace: A general-purpose face recognition library with mobile applications. Technická zpráva, CMU-CS-16-118, CMU School of Computer Science, 2016, [Online; navštíveno 16.04.2017].
URL https://cmusatyalab.github.io/openface/

[3] Berners-Lee, T.; Cailliau, R.: *HTML*. [Online; navštíveno 05.05.2017].
URL https://cs.wikipedia.org/wiki/HyperText_Markup_Language

[4] Blaha, M.: *Neuronové sítě - jednotlivý neuron*. [Online; navštíveno 14.04.2017].
URL http://portal.matematickabiologie.cz/res/f/neuronove-site-jednotlivy-neuron.pdf

[5] BogoToBogo: *Object detection : Face detection using Haar cascade classifiers*. [Online; navštíveno 16.04.2017].
URL http://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Object_Detection_Face_Detection_Haar_Cascade_Classifiers.php

[6] Bouvrie, J.: *Notes on Convolutional Neural Networks*. [Online; navštíveno 15.04.2017].
URL http://cogprints.org/5869/1/cnn_tutorial.pdf

[7] Collobert, R.; Kavukcuoglu, K.; Farabet, C.: *Torch*. [Online; navštíveno 16.04.2017].
URL http://torch.ch/

[8] Corporation, I.: *OpenCV*. [Online; navštíveno 01.05.2017].
URL http://opencv.org/

[9] Dalal, N.; Triggs, B.: *Histograms of Oriented Gradients for Human Detection*. [Online; navštíveno 20.04.2017].
URL http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

[10] Eich, B.: *JavaScript*. [Online; navštíveno 05.05.2017].
URL https://cs.wikipedia.org/wiki/JavaScript

[11] Garretta, J. J.: *Ajax*. [Online; navštíveno 06.05.2017].
URL https://cs.wikipedia.org/wiki/AJAX

[12] Geitgey, A.: *Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning*. [Online; navštíveno 01.05.2017].
URL https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d7

[13] Hipp, D. R.: *SQLite*. [Online; navštíveno 07.05.2017].
URL https://www.sqlite.org/about.html

[14] Höll, K.: *Aplikace metod detekce a rozpoznání obličeje*. Diplomová práce, Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií, Brno, 2013, [Online; navštíveno 16.04.2017].
URL
https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=82657

[15] Johnson, J.; Karpathy, A.: *Convolutional Neural Networks*. [Online; navštíveno 15.04.2017].
URL http://cs231n.github.io/convolutional-networks/

[16] Kačenka, P.: *Neuronové sítě*. [Online; navštíveno 14.04.2017].
URL http://mks.mff.cuni.cz/library/NeuronoveSitePK/NeuronoveSitePK.pdf

[17] King, D. E.: *Dlib*. [Online; navštíveno 01.05.2017].
URL http://dlib.net/

[18] Krajčovičová, M.: *Konvoluční neuronová síť pro zpracování obrazu*. Diplomová práce, Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií, Brno, 2015, [Online; navštíveno 15.04.2017].
URL https://dspace.vutbr.cz/xmlui/bitstream/handle/11012/40018/sablona.pdf?sequence=1&isAllowed=y

[19] Lefkowitz, G.: *Twisted*. [Online; navštíveno 02.05.2017].
URL http://twistedmatrix.com/trac/

[20] Lie, H. W.: *CSS*. [Online; navštíveno 05.05.2017].
URL https://cs.wikipedia.org/wiki/Kaskádové_styly

[21] Otto, M.; Thornton, J.: *Bootstrap*. [Online; navštíveno 05.05.2017].
URL https://cs.wikipedia.org/wiki/Bootstrap

[22] Rosebrock, A.: *Intersection over Union (IoU) for object detection*. [Online; navštíveno 06.05.2017].
URL https://cs.wikipedia.org/wiki/AJAX

[23] Rouse, M.: *Database management system (DBMS)*. [Online; navštíveno 03.05.2017].
URL http://searchsqlserver.techtarget.com/definition/database-management-system

[24] Solver.com: *Training an Artificial Neural Network - Intro*. [Online; navštíveno 14.04.2017].
URL http://www.solver.com/training-artificial-neural-network-intro

[25] Statistica, S.: *Úvod do neuronových sítí*. [Online; navštíveno 14.04.2017].
URL http://www.statsoft.cz/file1/PDF/newsletter/2013_02_05_StatSoft_Neuronove_site_linky.pdf

[26] Turner, A.: *Artificial Neural Networks*. [Online; navštíveno 14.04.2017].
URL http://andrewjamesturner.co.uk/ArtificialNeuralNetworks.php

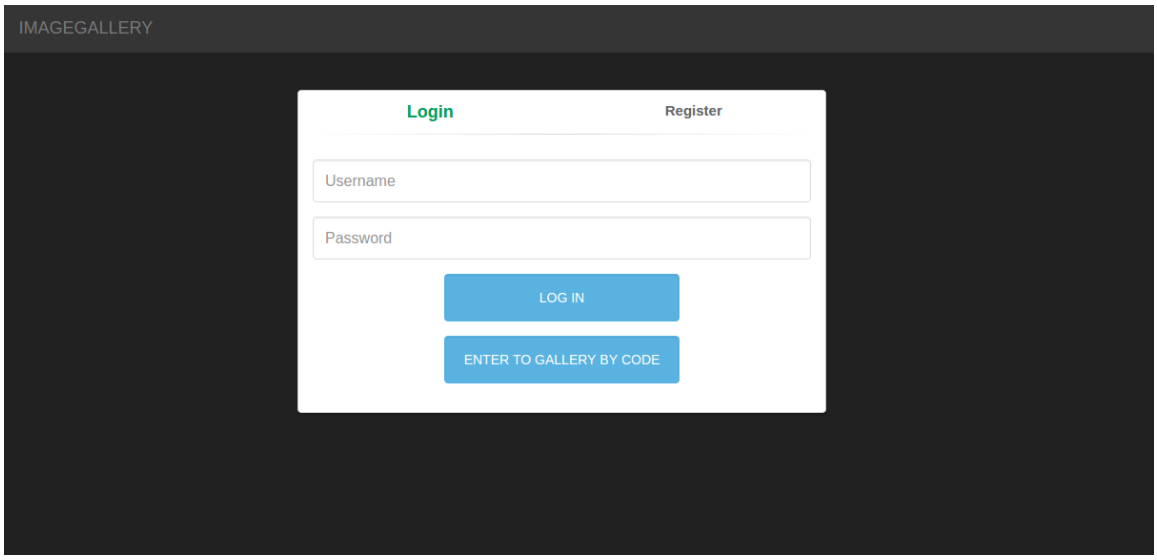
[27] Viola, P.; Jones, M. J.: *Robust Real-Time Face Detection*. [Online; navštíveno 20.04.2017].
URL http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf

Přílohy

Příloha A

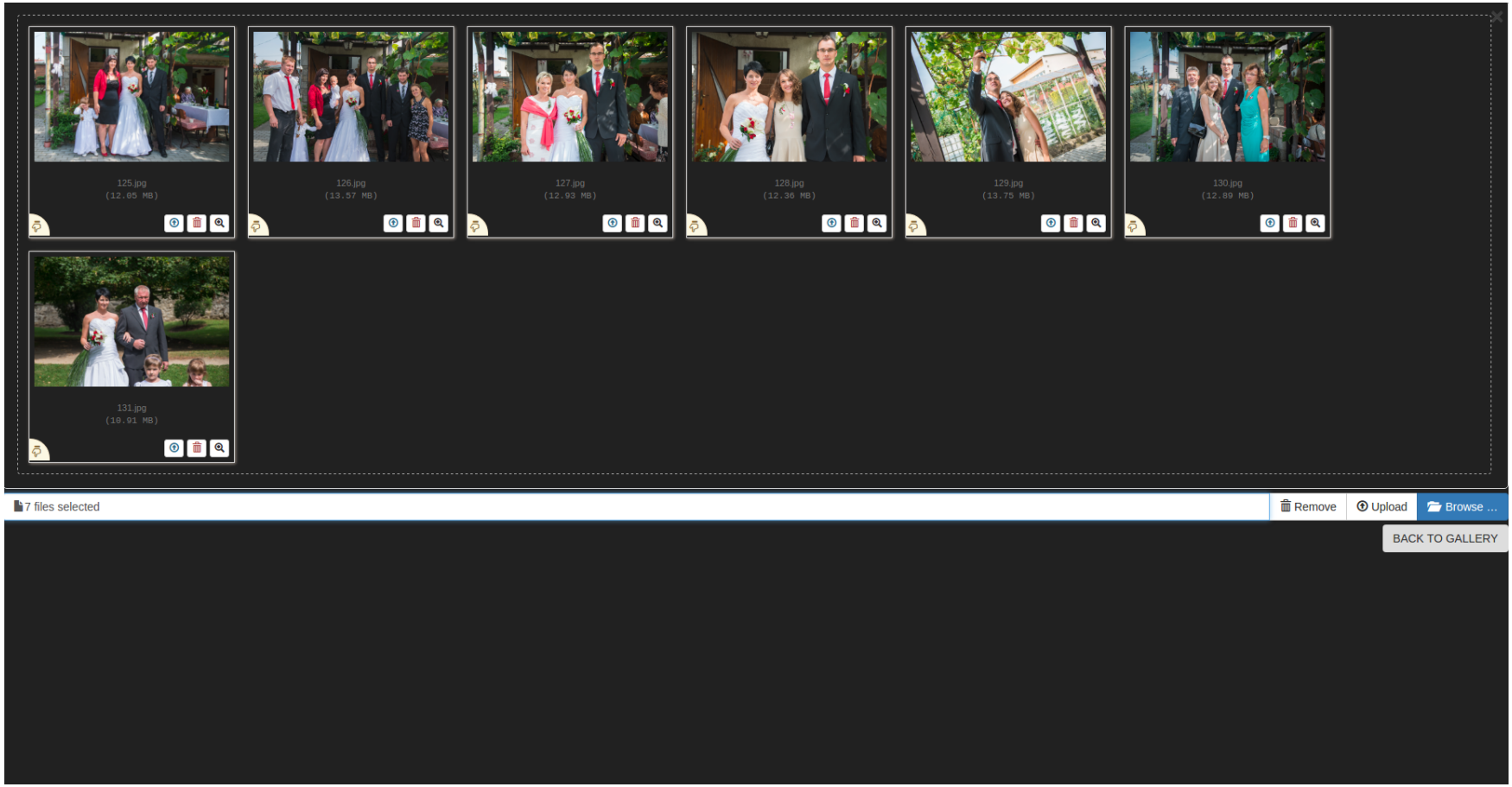
Uživatelské rozhraní

Obsah této přílohy je zaměřen na vzhled uživatelského rozhraní vytvořené aplikace.

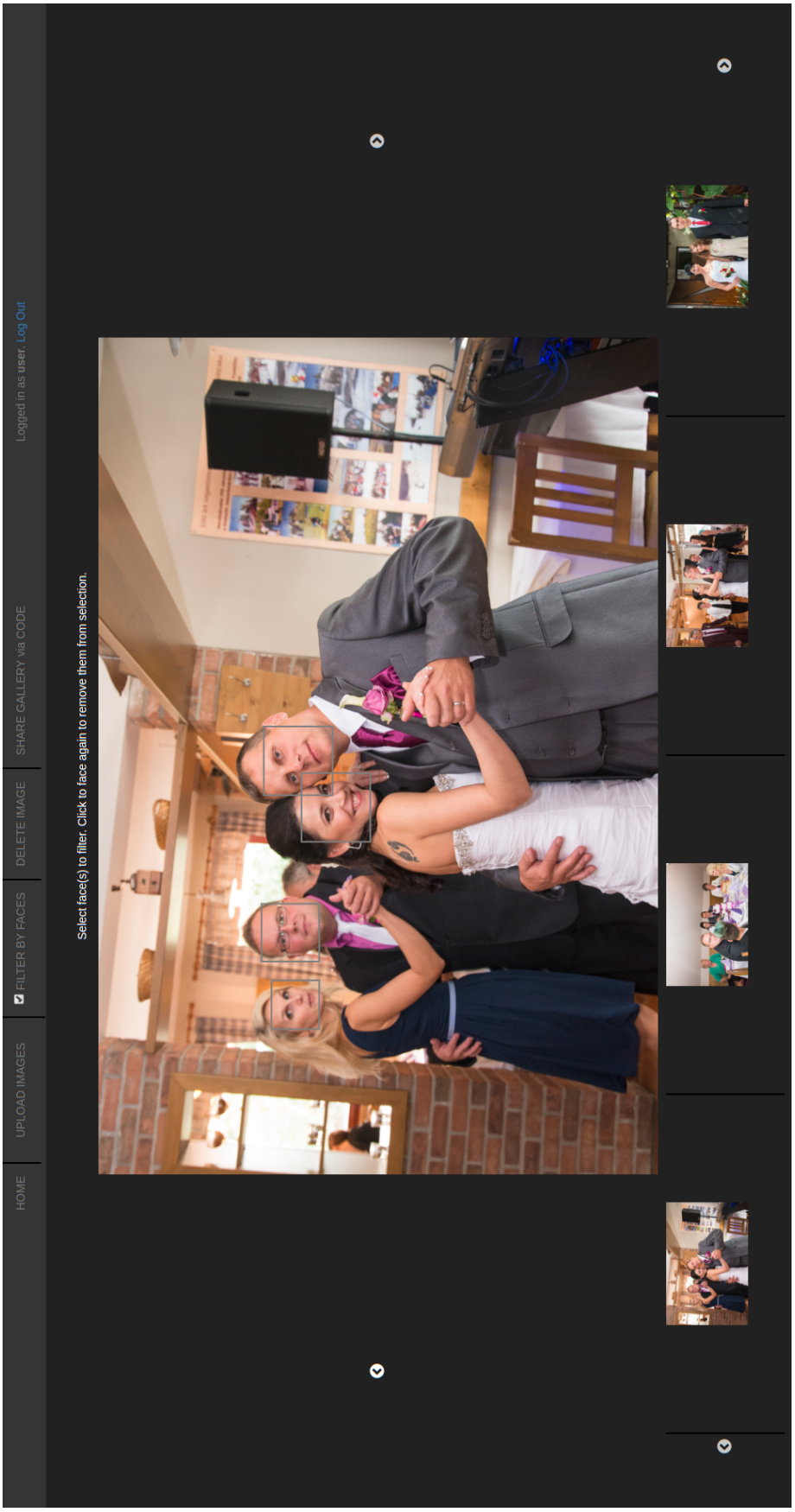


Obrázek A.1: Přihlašovací formulář

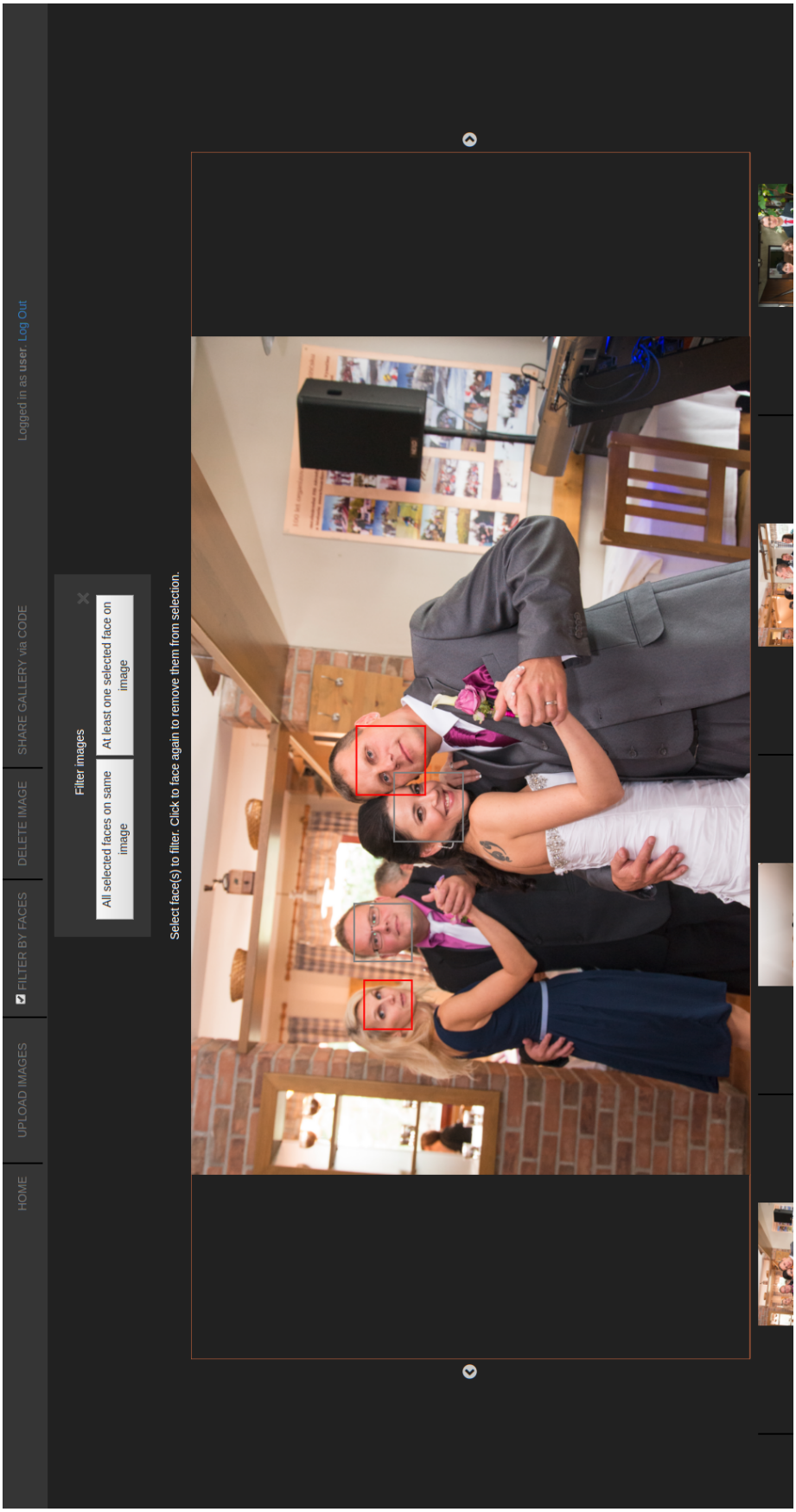
Obrázek A.2: Galerie s nahranými fotografiemi



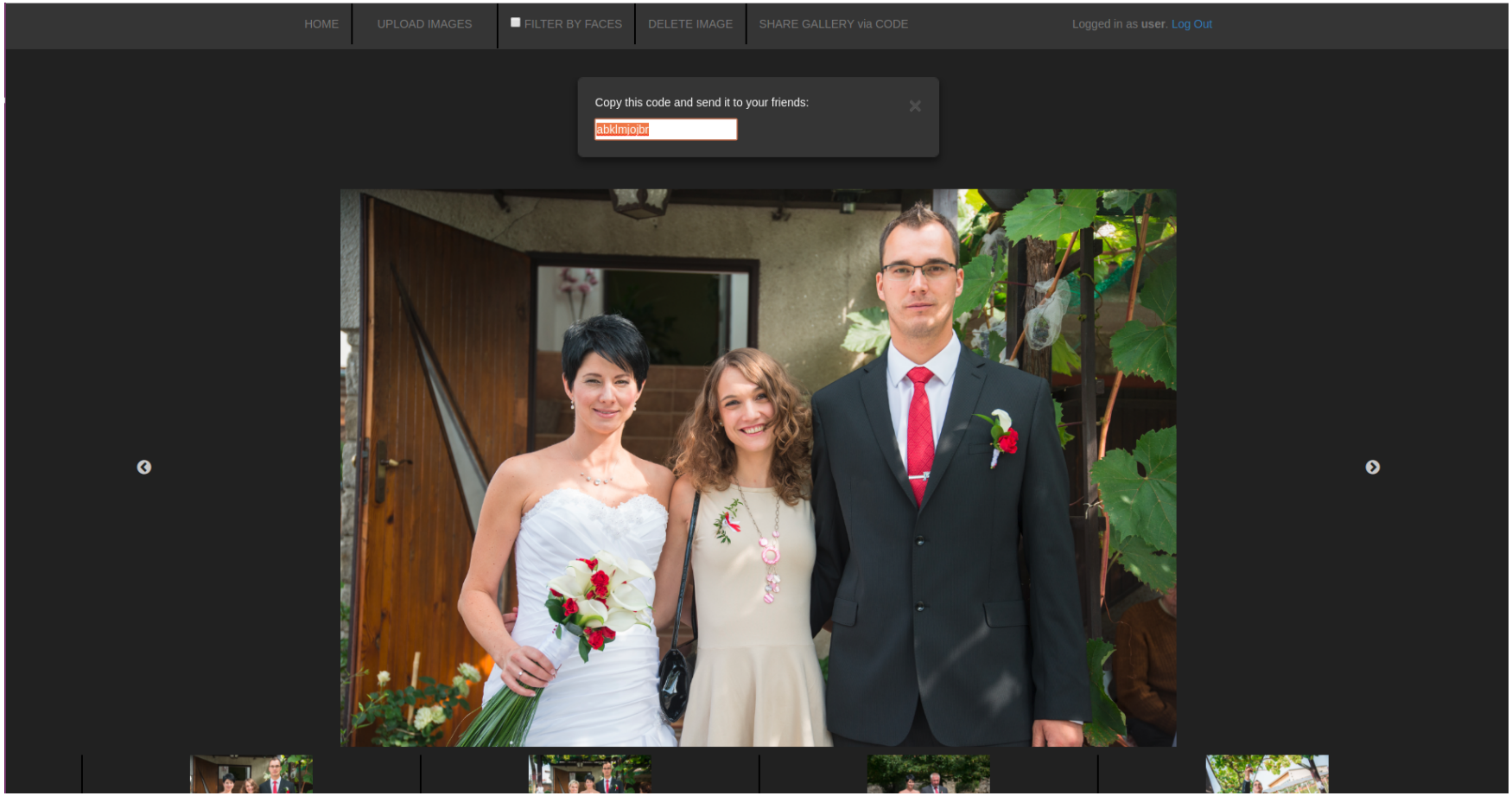
Obrázek A.3: Nahrávání fotografií do galerie



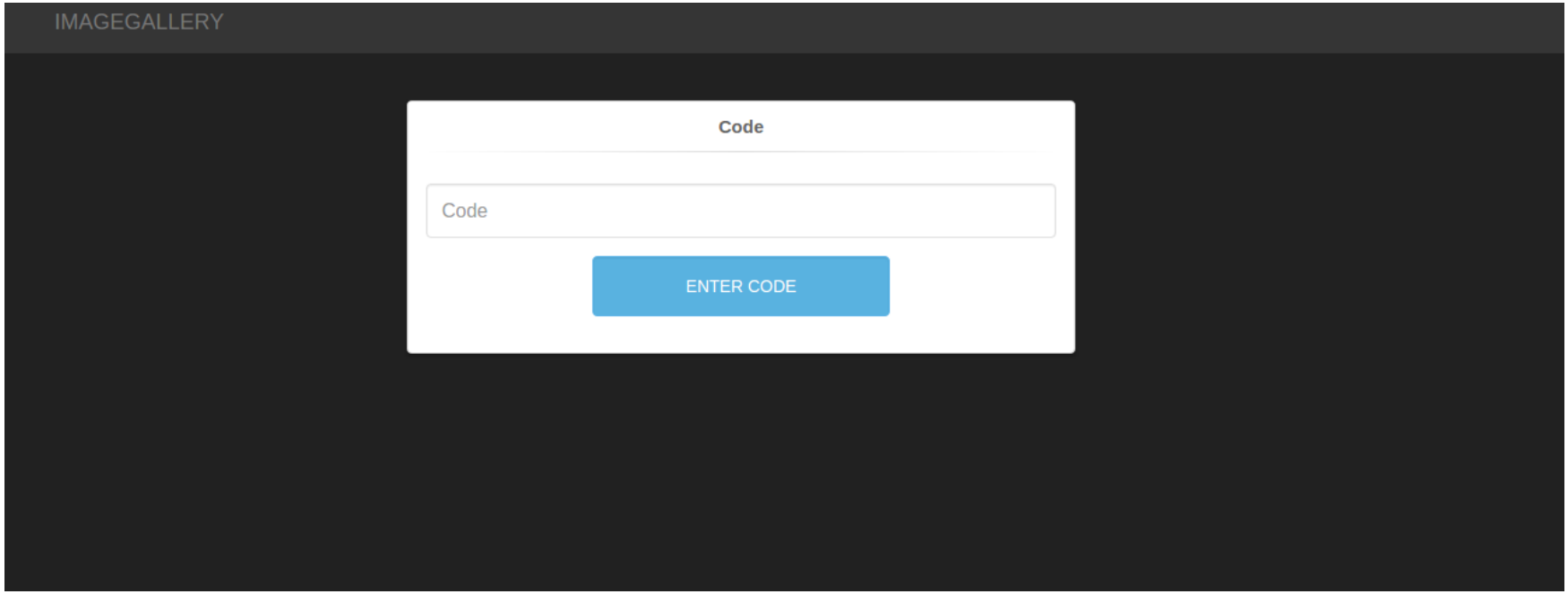
Obrázek A.4: Zobrazení detekovaných obličejů



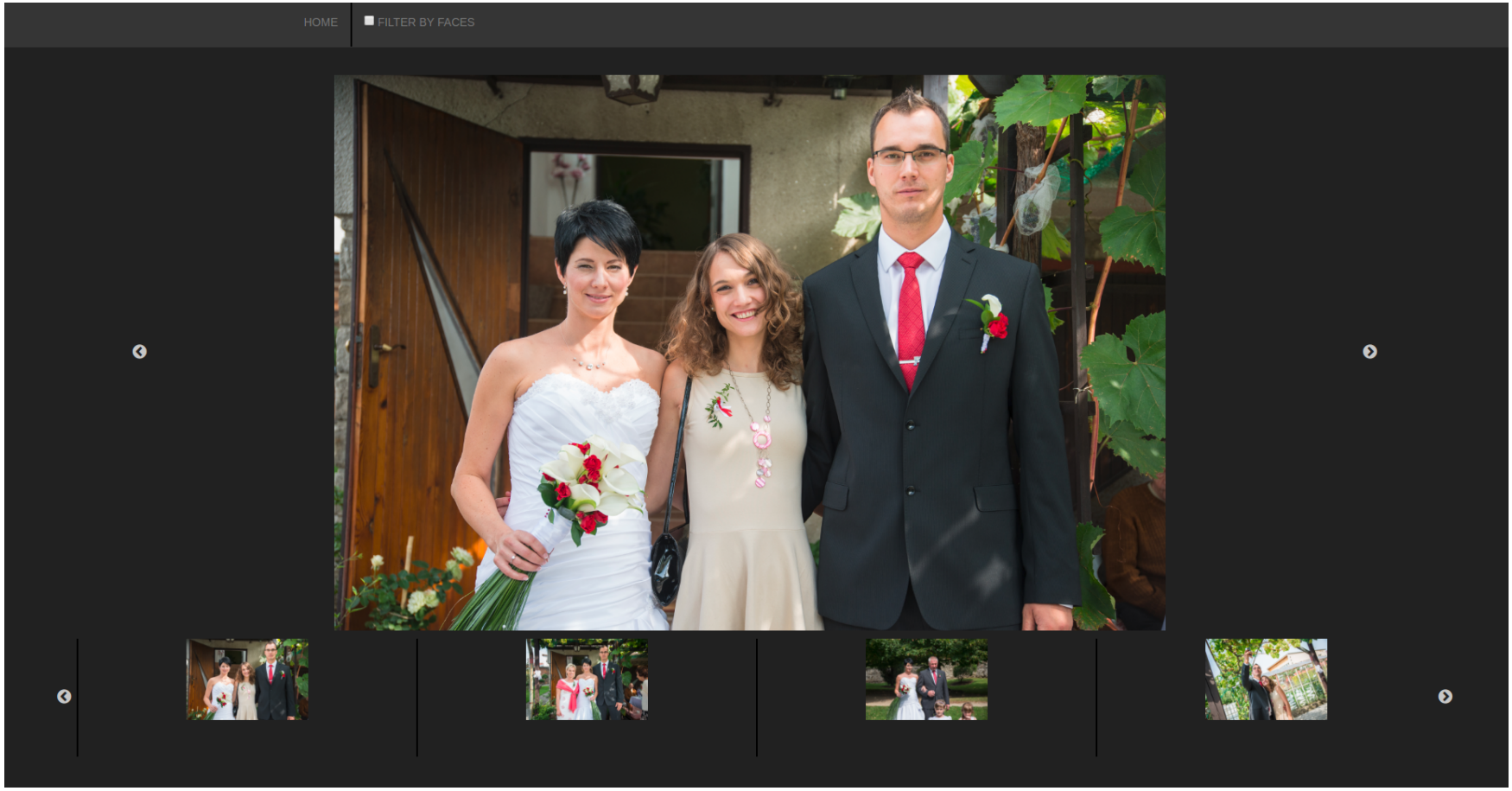
Obrázek A.5: Zvolení obličejů pro filtrování



Obrázek A.6: Sdílení prostřednictvím kódu



Obrázek A.7: Formulář sloužící k zadání kódu pro vstup do galerie bez přihlášení



Obrázek A.8: Galerie v režimu prohlížení po vstupu prostřednictvím kódu