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

ODHAD HUSTOTY DAVU OSOB Z FOTOGRAFIE

CROWD DENSITY ESTIMATION FROM A PHOTO

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ADAM FERENCZ**
AUTHOR

VEDOUCÍ PRÁCE — **VÍTĚZSLAV BERAN, Ing., Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Ferencz Adam**
Program: Informační technologie
Název: **Odhad hustoty davu osob z fotografie**
Crowd Density Estimation from a Photo
Kategorie: Zpracování obrazu

Zadání:

1. Seznamte se s metodami zpracování obrazu, předzpracování obrazových dat a konvolučních neuronových sítí. Zaměřte se na postupy odhadující hustotu davu v obraze.
2. Navrhněte celkový postup, dílčí metody a aplikaci, která bude z fotografie odhadovat hustotu davu v různých místech v obraze. Výsledkem nechť je mapa hustoty davu. Řešení navrhněte pro fotografie pořízené z horního pohledu (kamera na střeše budovy, drone apod.).
3. Připravte vhodnou datovou sadu a její anotaci tak, aby obsahovala různé relevantní situace.
4. Implementujte navržený systém pomocí dostupných knihoven.
5. Vyhodnoťte systém na připravené sadě dat.
6. Prezentujte klíčové vlastnosti řešení formou plakátu a krátkého videa.

Literatura:

- M. Sonka, V. Hlaváč, R. Boyle. *Image Processing, Analysis, and Machine Vision*, CL-Engineering, ISBN-13: 978-0495082521, 2007.
- Gary R. Bradski, Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*, ISBN 10: 0-596-51613-4, September 2008.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a částečně body 3 a 4.

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

Vedoucí práce: **Beran Vítězslav, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 1. listopadu 2019

Abstrakt

Cílem této práce je vytvořit aplikaci, která umožní získat odhad počtu lidí v davu na demonstraci či jiné hromadné akci. Vstupem je několik fotografií pořízených dronem, či jiných fotografií. Výsledkem jsou obarvené části mapy podle hustoty lidí v daném místě. Jednotlivé fotografie se umisťují do topologické mapy. Pro počítání lidí z fotky je použita metoda konvoluční neuronové sítě MCNN, která dokáže k fotografii vytvořit příslušnou mapu hustoty lidí. Pro zachování správného celkového odhadu v případě, že se obrázky v mapě překrývají je navrhnut algoritmus korekce překryvů. Aplikace je rozdělena na serverovou a klientskou část. Serverová část se stará o vytvoření map hustoty, ukládá data a dělá algoritmus korekce překryvů. Klient zpracovává vstupy uživatele a zobrazuje mu interaktivní mapu, která vše vizualizuje.

Abstract

The aim of this thesis is to develop an aplication estimating the total number of people at a demonstration or at other public events. Input is a serie of photos from a drone or some other photos. The output are couloured maps according to people density in the place. Photos are placed in a topological map. Convolutional neural network MCNN is used for the crowd counting, which can generate a density map from the photo. Special method was proposed to correct the total amount of counted people when photographs overlap. The application is divided into server and web client. The server part generates density maps, saves data and runs an overlap correction algorithm. Client handles user inputs and provides an interactiv map with visualization.

Klíčová slova

zpracování obrazu, konvoluční neuronové sítě, hustota davu, OpenCV, Keras, počítání lidí, počítačové vidění, Leaflet, OpenStreetMap, Flask, GUI

Keywords

image processing, convolutional neural networks, crowd density, OpenCV, Keras, crowd counting, computer vision, Leaflet, OpenStreetMap, Flask, GUI

Citace

FERENCZ, Adam. *Odhad hustoty davu osob z fotografie*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Vítězslav Beran, Ing., Ph.D.

Odhad hustoty davu osob z fotografie

Prohlášení

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

.....
Adam Ferencz
28. května 2020

Poděkování

Velmi děkuji panu Ing. Vítězslavu Beranovi Ph.D. za jeho rady, které mi pomohly při tvorbě této práce. Také bych chtěl poděkovat své rodině, která mě podporovala a umožnila mi v klidu pracovat.

Obsah

Kapitola 1

Úvod

Shlukování lidí do rozsáhlých davů při různých příležitostech je běžným fenoménem dnešní doby. S rostoucí hustotou davu roste také riziko konfliktu. Ten může vzniknou strkáním, masivním panikařením a způsobit celkovou ztrátu kontroly. Analýza davu je užitečná především pro bezpečnostní monitorování událostí s vyšším počtem lidí, jako jsou veřejné demonstrace a sportovní události. Z tohoto důvodu se o analýzu dynamiky a chování davu zajímá již mnoho oborů, jako je psychologie, sociologie, veřejné služby, bezpečnost a počítačové vidění [26].

O co nejpřesnější určení celkového počtu lidí na demonstraci či na veřejné události mají velký zájem jak novináři, tak i policie. Aktuálně ale neexistují žádná vhodná řešení, a tak se většinou musí přistoupit k velmi hrubým odhadům.

S rozvojem technologie dronů přichází možnost pozorovat tyto davy z výšky a následně získané záběry pak využít pro účely analýzy.

V historii bylo způsobeno mnoho úmrtí právě ztrátou kontroly nad davem. Například v roce 1996 v Guatemala City přišlo o život 84 lidí na fotbalovém hřišti, v roce 2001 v Ghaně to bylo 126 lidí při fotbalovém zápase, v roce 2003 bylo udupáno k smrti 21 evakuujících se při opouštění klubu a při tragédii Khmer Water Festival zahynulo 347 lidí při hromadném úprku [33].

Právě z tohoto důvodu je třeba dav analyzovat a snažit se těmto situacím předcházet. V tom může pomoct počítačové vidění tím, že se vytvoří systémy pro odhad hustoty a počtu lidí v davu. S rostoucí popularitou dronů bude získání vhodných záběrů davu snadné a zbývá tedy dav jen spočítat. To je pro člověka v krátkém časovém úseku nemožné a proto je zde zapotřebí vyvinout systém, který to dokáže rychle a s dostatečnou přesností. Na základě získaných informací pak mohou služby veřejné bezpečnosti jednat včas a na správném místě.

Na analýzu metod vhodných pro počítání davu je zaměřena kapitola 2. V kapitole 3 pak popíši návrh aplikace pro praktické využití neuronových sítí v úloze počítání davu a vlastní implementaci rozeberu v kapitole 4.

Při řešení tohoto problému je třeba čelit komplikacím jako je variabilita davu, různý úhel a vzdálenost pořizovaného obrazu. Ve vlastní aplikaci pak je důležité intuitivní ovládání a vhodná vizualizace.

Kapitola 2

Teorie počítání osob v davu z obrazu

Problém určení hustoty a počtu lidí v davu můžeme rozdělit do dvou základních kategorií. Základní dělení je na metody klasické a metody využívající neuronových sítí. Metody klasické se zaměřují na vyhledávání ručně vytvořených příznaků, detekci hran, předzpracování obrazu. Metody využívající neuronových sítí jsou obecně mladší záležitost. Mnohem lépe dokáží generalizovat a vyrovnat se se zašuměným obrazem. Jejich nevýhodou je potřeba rozsáhlého datasetu s anotacemi a většinou vyšší výpočetní náročnost při trénování. To v dnešní době výkonných grafických karet ale není problémem.

Dále můžeme rozdělit přístupy pro počítání davu podle toho, jak se k celkovému počtu dostanou. Buď se snaží určit počet osob přímo (počítání objektů) nebo nepřímo (hustota v dané oblasti). Přímý přístup, založený na počítání objektů, se snaží detekovat jedince v obraze a poté sečíst jejich celkový počet. Nepřímé způsoby se snaží detekovat hustotu lidí v místě obrazu a poté z celkové mapy hustoty vyvodit výsledný počet.

Poslední podstatnou informací pro volbu metody je, jaká jsou vstupní data. Zda se jedná o jednu statickou fotku, nebo o sekvenci z drona, či CCTV kamery. Vstupní data totiž určují, jaké možnosti se naskytnou při analýze. Například ve statickém obraze není možné detekovat lidi jako pohyblivé shluky částic, protože se nepohybují.

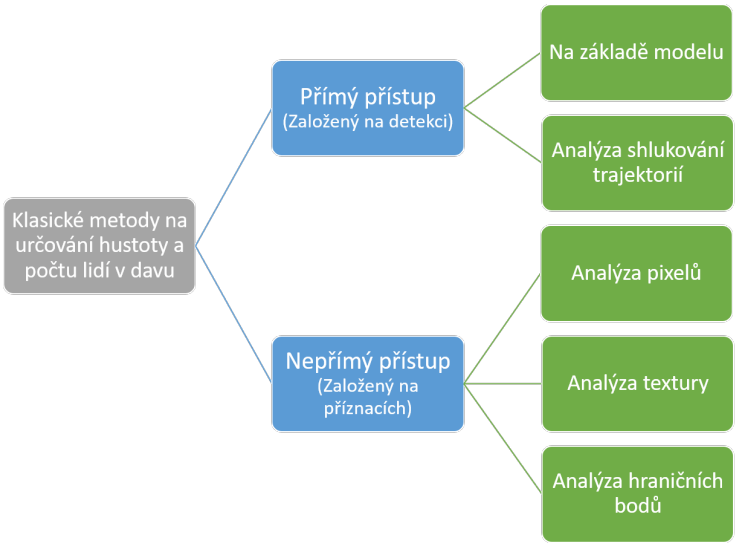
2.1 Klasické metody

Jejich výhodou je, že se konkrétně specifikuje, jak má algoritmus postupovat, a není tak třeba rozsáhlých datasetů. *V Recent survey on crowd density estimation and counting for visual surveillance* [26] se klasické metody klasifikují na přímé a nepřímé, ty se pak dělí na další podkategorie, viz schéma 2.1.

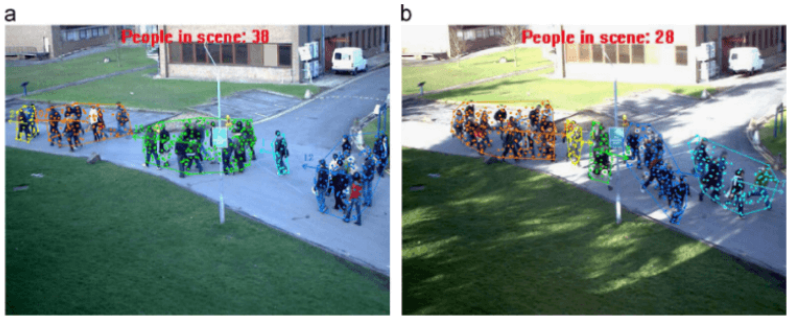
Analýza na základě modelu je prvním přímým způsobem. Principem je detekce jedinců v davu a jejich následné sečtení pomocí modelu nebo výskytu lidských příznaků. Příkladem může být *monoitic detection* a *head-like detection*.

Přístup na základě analýzy shlukování trajektorií využívá toho, že pozadí obrazu je oproti davu nehybné. Shlukuje tedy zajímavé pohybující se body a sleduje je v průběhu času (viz obrázek 2.2). Metody jsou velmi závislé na tom, že se dav pohybuje, pokud osoba zůstane stát bez pobytu, přestane být systém přesný.

Tyto metody fungují dobře na data s nízkou hustotou, jelikož v řídkém davu lze od sebe jednotlivé objekty (osoby) odlišit a poté spočítat. Pokud ale dav začne být hustý a na



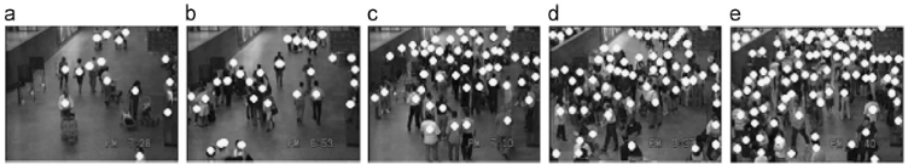
Obrázek 2.1: Rozdělení klasických metod pro počítání davu. Převzato z [26].



Obrázek 2.2: Shlukování trajektorií - Counting crowdflow based on feature points [21].

fotce z větší dálky, tyto metody přestávají fungovat. Tento problém se pokouší řešit nepřímé způsoby.

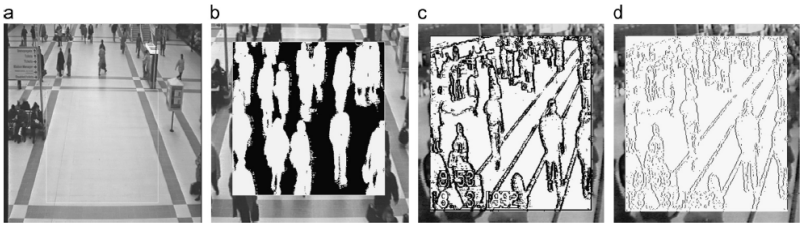
Analýza na základě pixelů se soustředí na velmi lokální příznaky, kvůli tomu spíše odhade pouze hustotu lidí než konkrétní umístění jednotlivců. Většina technik používá odstranění pozadí (viz obrázek 2.4) jako první krok buď na základě referenčního pozadí, nebo si generují pozadí uměle [26] (např. Hussain et al., 2011 [14]).



Obrázek 2.3: Pět různých tříd hustoty. Převzato z [26].

Analýza na textury: Marana [23] ve své práci popisuje, že husté davy mají jemnější textury a řidší davy mají hrubší. Poté se pomocí statistické metody Gray Level Dependency

Matrices (GLDM) [11] určí příznaky. Používají se: kontrast, homogenita, entropie[1] a energie. Potom se použije Kohogenova SOM[2] [19] neuronová síť na klasifikaci do pěti tříd hustoty [26]. Ty jsou zobrazené na obrázku 2.3.



Obrázek 2.4: Použití statického pozadí. (a) Referenční obrázek, (b) odstranění pozadí, (c) detekce hran, (d) výstup po ztenčení. Převzato z [26].

Sledování hran[3] nevykonává segmentaci nebo hledání jedinců. Jednoduše řečeno se snaží sledovat pohybující se hrany. Ty určuje pomocí konceptu Harrisova algoritmu [12]. I přes jednoduchost měl velký úspěch a na jeho práci později navázali další.

Přehled klasických metod je vhodný pro stanovení základních principů, kterými se úlohy počítání davu řešily. Tyto metody se využívaly do roku 2015, kdy byl přehled *Recent survey on crowd density estimation and countingfor visual surveillance* zpracován. Od té doby ale velmi zesílila výpočetní síla a došlo k popularizaci konvolučních neuronových sítí v oblasti počítačového vidění. Také v tuto dobu ještě neexistovaly velké datasety, jako je ShanghaiTech etc. Právě na tyto nové metody se zaměřím v podkapitole 2.3.

2.2 Neuronové sítě

Následující text je nepřímo převzat z [17]. Předtím, než popíši konkrétní přístupy založené na neuronových sítích, popíši principy a fungování neuronových sítí.

Lidský mozek se za mnoho let vyvinul a získal vlastnosti, které nejsou přítomné v dnešních počítačích. Jsou to například: masivní paralelizmus, schopnost se učit, aditivita, tolerance chyby a schopnost generalizace. Právě ve snaze získat tyto vlastnosti vznikly neuronové sítě, které se inspirují procesy v lidském mozku.

V biologickém smyslu je neuron buňka napojená na ostatní neurony. Ty mezi sebou komunikují. Skládá se z těla buňky (soma) a dvou typů napojení (dendrid a axon). Neuron na dendridech přijímá vzruchy a v buňce generuje signál, který přes axon posílá dál. Neuron bude na vzruch reagovat pouze pokud překročí určitý aktivační práh.

Základním stavebním prvkem neuronové sítě je umělý neuron, který je zjednodušením toho biologického. Má své vstupy, které podle vah sčítá a následně posílá dál výstup na základě aktivační funkce. V neuronové síti jsou neurony uspořádané do jednotlivých vrstev. První se nazývá vstupní a poslední výstupní. Vrstvy mezi nimi jsou skryté vrstvy. Nazývají se tak, protože při použití neuronové sítě se na jejich vnitřní hodnoty programátor nedívá, pouze deklaruje jejich strukturu.

Neuronové sítě dělíme na dva základní typy:

- **Feed forward** - v této architektuře neuronové sítě data proudí pouze jedním směrem. Výstupy neuronu tedy nejsou napojeny zpět na vstupy. Neznamená to ale, že se síť

[1] neuspořádanost
[2] Kohonens Self-Organizing Mapping
[3] Corner point based analysis

nedokáže při průchodu dat učit, jen se to děje pomocí algoritmu backpropagation, až když data dojdou na výstupní vrstvu [27].

- **Recurent** - Tyto neuronové sítě jsou dynamické a závisí vždy na předchozím stavu. Pro výpočet výstupu neuronu na základě vstupních dat je použit i jeho předchozí výstup [18]. Tato zpětná vazba se dá poté využít například při práci se vstupními daty, které jsou na sobě závislé v čase.

Učení neuronových sítí

Smyslem neuronových sítí je naučit se ve vstupních datech rozeznávat vzory a na základě nich podávat námi požadovaný výstup. Pojem vzory svádí ke zobecnění pouze na obrazová data, ale ve skutečnosti je možné mít na vstupu data různá, jako číselné informace, fyzikální veličiny a další. Neuronová síť se pak učí souvislosti mezi daty. Neuronová síť se učí rozpoznávat vzory aktualizováním vah spojů mezi jednotlivými neurony [24].

Učení s učitelem je způsob trénování, kde jsou neuronové síti poskytnuty dva sety vstupních dat - trénovací set a testovací set. Cílem je, aby se neuronová síť naučila rozeznat, klasifikovat data na trénovacím setu, u kterého vidí anotace (správný výstup). Poté by měla být schopna klasifikovat i testovací set [13].

Vždy po průchodu vstupních vah se může neuronová síť zhodnotit a aktualizovat váhy na základě své úspěšnosti. Pro trénování je potře mít všechna vstupní data anotovaná, což může být náročný úkol vzhledem k tomu, že čím více dat, tím lépe a obecněji se model natrénuje.

Učení bez učitele se používá v situacích, když nejsou dostupná anotovaná data, ale velké množství dat neanotovaných. Jako metriku zde ale nemůžeme použít chybu, jelikož není znám očekávaný výstup. Místo toho se používá například metoda shlukování. Neuronová síť ve vstupním prostoru hledá sobě podobná data. Na základě parametrů vstupních vzorků je třídí do skupin. Počet skupin může být předem dán [13].

Kombinované učení přichází v úvahu, když je k dispozici velká sada dat, ale jen část z nich je anotovaná. Potom se může trénovat nejdříve na datech anotovaných, poté podle modelu "pseudo-anotovat". Následné probíhá další trénování, nyní na celé datové sadě.

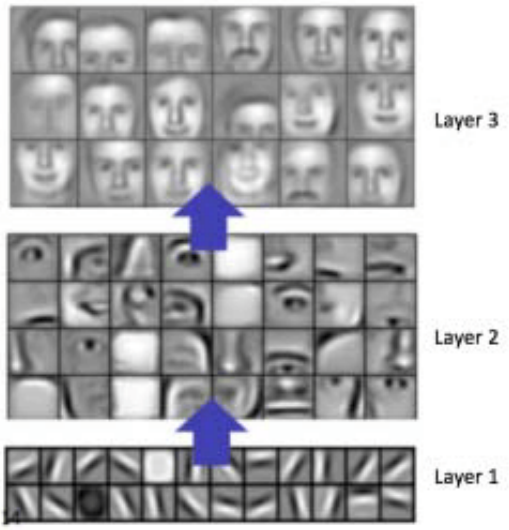
Konvoluční neuronové sítě

Pro zpracování obrazu jsou velmi vhodné konvoluční neuronové sítě, o kterých zde shrnu nejdůležitější informace podle článku [1].

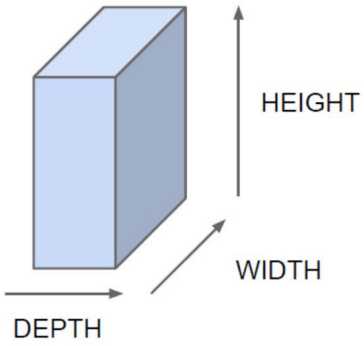
Hlubokou neuronovou sítí je myšlena jakákoli neuronová síť, která má více než jednu skrytou vrstvu. Speciálně v rozpoznávání vzorů jsou pak nejpopulárnější konvoluční neuronové sítě (Zkráceně CNN - Convolutional Neural Network). CNN zpravidla obsahují konvoluční vrstvu, nelineární vrstvu, pooling vrstvu a fully-connected vrstvu. Mají dva základní aspekty:

- Problém řešený CNN nemá příznaky závislé na umístění. Jinými slovy například při řešení detekce obličeje se může obličej vyskytovat v jakémkoli místě obrazu.

- Pro vstup vznikají abstraktní příznaky tím jak postupuje skrze více vrstev. Například nejdříve se detekují hrany, poté základní tvary a nakonec vyšší příznaky. Jak je vidět na obrázku 2.5 a 2.10.

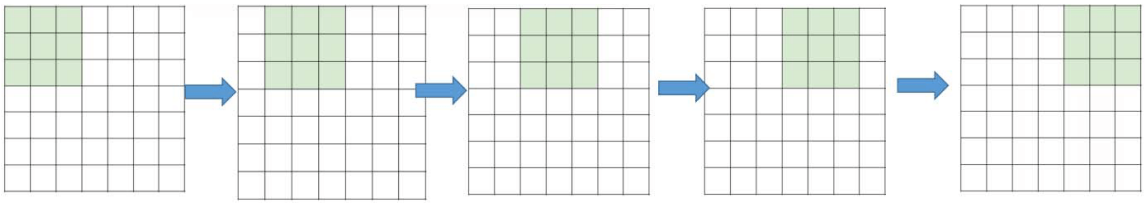


Obrázek 2.5: Ukázka různých příznaků v naučených na vrstvách CNN. Převzato z [1].



Obrázek 2.6: Dimenze vstupu CNN. Převzato z [1].

Vstup CNN má tři dimenze viz obrázek 2.6. Nejčastěji jsou dimenzemi výška a šířka obrázku se třetí dimenzí je RGB barevná hloubka. Vstupem ale může být například i černobílé video, kde hloubku nahrazují jednotlivé snímky.



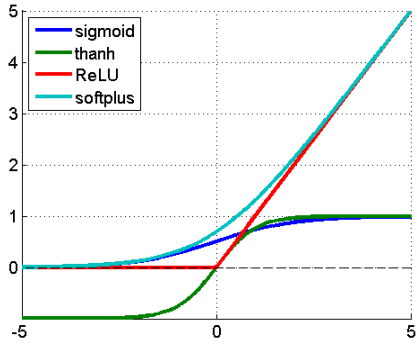
Obrázek 2.7: Konvoluce s filtrem o velikosti 3x3 a krokem 1. Převzato z [1].

Pro pochopení **konvoluční vrstvy** je dobré znát následující základní prvky:

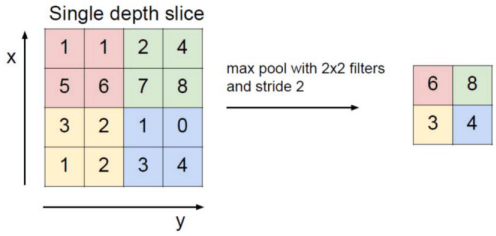
1. **Konvoluce** - Motivací pro zavedení konvoluce bylo nahrazení fully-connected vrstvy, která u obrázku vytvořila zbytečně velké množství vážených spojení (parametrů). Konvoluci jde přirovnat k posuvnému oknu, kde takové okno je konvoluční matice viz obrázek 2.7. Tímto způsobem je možné hledat příznaky bez závislosti na místě obrazu. Pokud bychom manuálně nastavili váhy na posuvném okně, bylo by možné dosáhnou výsledků jako je detekce hran, ostření či rozmazání. V CNN se ale právě tyto váhy mění v průběhu tréninku a tak vzniknou ideální filtry pro danou úlohu.

2. **Stride** (krok) je parametr, který se věnuje tomu, jak rychle se matice posouvá. Pokud se posouvá například okno 3x3 vždy o jedna na vstupním obraze 7x7, výstup bude ve formátu 5x5. Pokud ale zvětšíme krok na 2 bude výstup 3x3. To dává možnost dále snížit počet parametrů.

3. **Padding** - jedním z problémů konvoluce je možná ztráta informace na krajích obrazu. Tím totiž dochází ke zmíněnému smršťovaní výstupu a později to může být limitující



Obrázek 2.8: Známé typy nelineárních funkcí. Převzato z [1].



Obrázek 2.9: Demonstrace max-poolingu s filtrem 2x2 a krokem 2. Je vidět, že vstup 4x4 byl namapován na výstup 2x2. Převzato z [1].

při navyšovaní počtu vrstev CNN (nelze smršťovat donekonečna). Nejjednodušším řešením je tzv. zero-padding, který v praxi znamená, že se okolo vstupního obrazu přidají pomyslné nuly. V některých případech ale není zero-padding ideální, a proto existují i další možnosti ohraničení.

4. Vzorec pro konvoluci jednoho pixelu v následující vrstvě pak vypadá následovně:

$$net(t,j) = (x * w)[t,j] = \sum_m \sum_n x[m,n]w[t-m][t-n]$$

kde $net(t,j)$ je výstupní v další vrstvě, x je vstupní obrázek, w je matice filtru a $*$ je operace konvoluce.

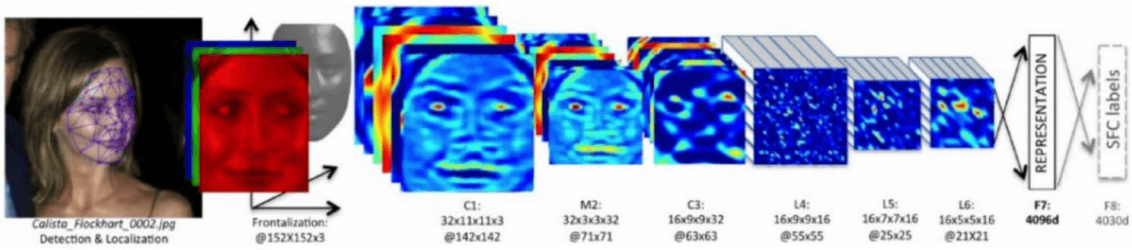
Nelineární vrstva následuje po vrstvě konvoluční a jejím účelem je zvýraznit výsledky konvoluce a limitovat tak dalším způsobem výstup. Pro její provedení se používají známé funkce, které jsou dříve hojně používané sigmoid a tahn, nicméně v dnešní době je nejpoužívanější Rectified Linear Unit - ReLU, který vyřešil problém slábnutí signálu u hlubších neuronových sítí. Ukázky průběhu těchto nelineárních funkcí jsou na obrázku 2.8.

Hlavní myšlenkou **pooling vrstvy** je snížení komplexnosti obrazu. Její princip je jednoduchý. Pole obrázku se rozdělí na pomyslné regiony (nejčastěji 2x2). V případě max-poolingu se z každého regionu vybere největší hodnota. Max-pooling hezky ukazuje obrázek 2.9.

Poslední vrstvou, která CNN uzavírá, je **fully-connected vrstva**. Ta je stejná jako u tradičních neuronových sítí. Vyskytuje se zde také nejvíce parametrů, což zvyšuje časovou náročnost tréninku. V této vrstvě se pro klasifikační CNN získávají z výstupu předchozích vrstev příznaky, které pak naznačí, co je na obrázku.

2.3 Přístupy založené na neuronových sítích

U tradičních metod se neuronové sítě používaly maximálně k regresi již předem získaných příznaků. Na základě úspěchu konvolučních neuronových sítí se přistoupilo k jejich aplikaci v počítání davu. Neuronové sítě mají výhodu, že se dokáží naučit rozeznávat vlastní příznaky na obrázcích, a tak vyřešit problém s počítáním velmi hustého davu. Pro trénovaní neuronových sítí je třeba zajistit velkou anotovanou sadu trénovacích dat, která odpovídají situaci, na kterou chceme náš model připravit (viz obrázek 2.11). Lze tedy říct, že s rostoucí



Obrázek 2.10: Vizualizace dat na jednotlivých vrstvách hluboké konvoluční neuronové sítě. Převzato z [1].

dostupností kvalitních datových sad bude růst i úspěšnost natrénovaných modelů. V článku *A survey of recent advances in CNN-based single image crowd counting and density estimation* [29] z roku 2017 se kategorizují přístupy využívající neuronových sítí do kategorií podle architektury CNN a jejich přístupu k překonání problému typu generalizace, různá perspektiva, velikost hlav a dalších.

Počítání na základě mapy hustoty

Při trénování neuronové sítě se vždy jedná o mapování vstupů na výstupy. V tomto případě jsou dva možné přístupy, jak vnímat výstup neuronové sítě. Tím prvním je, že pro vstupní obrázek bude mapa bodů, kde se vyskytují jednotlivé hlavy. Druhý postup na výstupu neřekne, kde jsou konkrétní jedinci, ale místo toho je výstupem mapa hustoty, která říká, jak je v daném místě vysoká hustota lidí (řekněme například kolik je zde lidí na jeden pixel). Poté se zjistí celkový počet výpočtem integrálu, tedy objemem pod celou touto 2D mapou hustoty. V praxi se to ale počítá jako suma všech hodnot mapy hustoty. Výhodou tohoto postupu je to, že je přesnější v komplexních scénách a hustých davech.

Pro trénink modelu je tedy potřeba připravit co nejlepší mapy hustoty z anotovaných dat (označené body polohy dat). Na získání mapy hustoty z anotovaných dat existuje více přístupů.

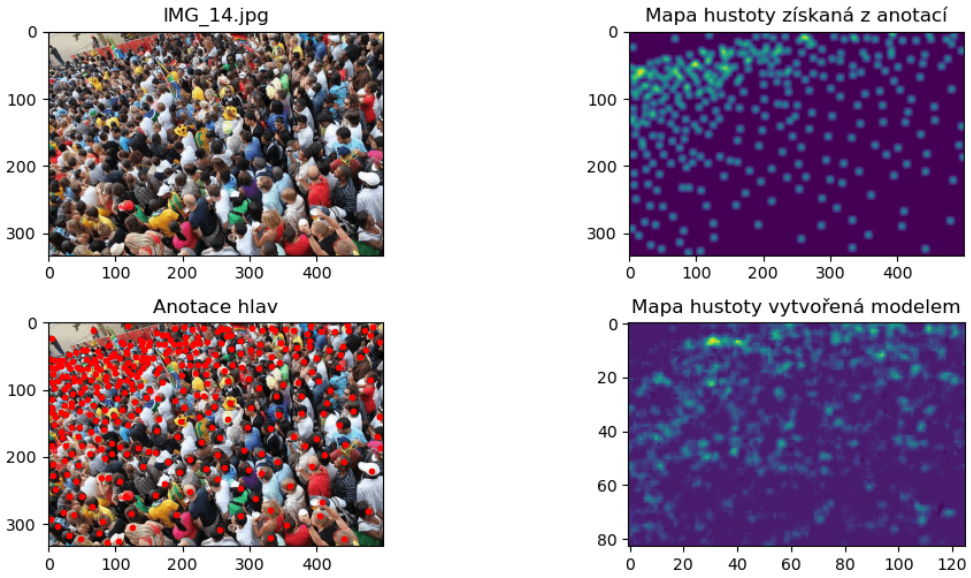
Nejjednodušším přístupem je použit konvoluci s gausovským filtrem (viz obrázek 2.12) a simulovat tak výskyt hlav. V principu jde o rozmazání každého bodu, čímž se simuluje, že jedna hlava koresponduje s více než jedním pixelem.

Tato metoda, ale nebere v úvahu perspektivní pohled a tedy různou velikost hlav. Fotka je většinou pořízena ze šikmého pohledu a proto je potřeba toto vzít v úvahu při generování. Existují dva způsoby jak toto implementovat. První odvozuje perspektivní geometrii z výšky lidí na obrázku. Druhý odvozuje perspektivu podle vzdálenosti okolních hlav. Kratší vzdálenost a velmi hodně okolních hlav znamená pozici někde dále v obraze, kde jsou hlavy menší. Právě tuto druhou metodu - *k-nearest neighbours* použili autoři pro generování perspektivních map hustoty. [4]

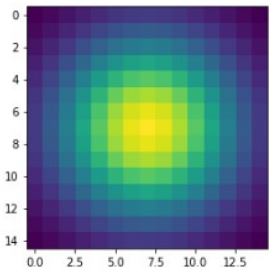
V článku [29] z roku 2018 se kategorizují přístupy využívající neuronových sítí do kategorií podle architektury CNN a podle jejich přístupu k překonání problému typu generalizace (různá perspektiva, velikost hlav a dalších). V této kapitole budou informace čerpány právě z tohoto shrnujícího článku. Kategorizace je na obrázku 2.13.

- **Základní CNN:** Přístupy, které určily počátek možnosti využití neuronových sítí pro analýzu davu. Jejich architektura není nijak kombinovaná a robustní.

[4] https://github.com/CommissarMa/Crowd_counting_from_scratch



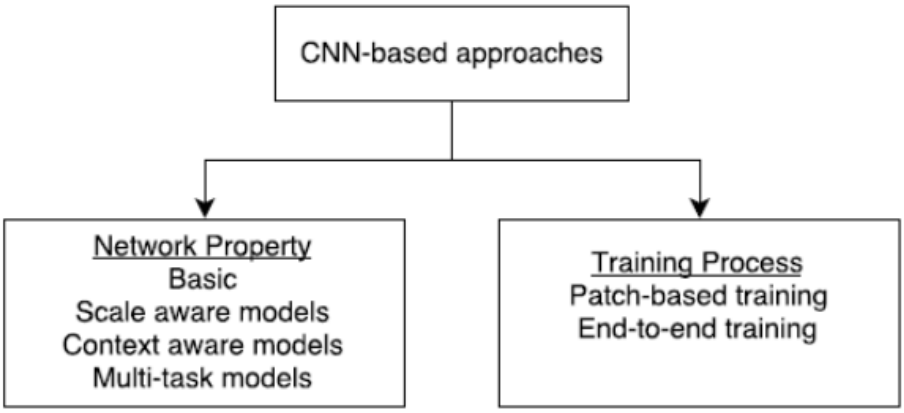
Obrázek 2.11: Ukázka fotografie z datasetu *ShanhaiTech A* [38], její anotace, vzniklé mapy hustoty a výstupu natrénovaného modelu.



Obrázek 2.12: Vizuální reprezentace Gausovského filtru.

- **Scale-aware models:** K základnímu modelu přidávají myšlenku možné změny přiblížení a oddálení, na které se snaží reagovat a zlepšit tak výsledek. Toho dosáhnou použitím více sloupcových architektur, nebo architentur pracujících s různým rozlišením.
- **Context-aware models:** Tyto přístupy přidávají extrakci příznaků (globálních a lokálních), na základě kterých se pak snaží snížit celkovou chybu odhadovaného počtu, či výsledné mapy hustoty.
- **Multi-task frameworks:** Jako reakce na úspěch multitask learningu se mnoho přístupů pokusilo zařadit tuto metodu do svého řešení. Jedná se obecně o sdružení více úloh dohromady. V tomto případě je počítání davu a odhad hustoty kombinován s oddělením pozadí a popředí, nebo s určením rychlosti davu.

Wang et al. [32] a Fu et al. [8] byli první, kdo použili CNN pro řešení problému počítání davu. Adaptovali AlexNet nahrazením poslední plně propojené vrstvy za jeden neuron odhadující počet. Aby zamezili započítávání nechtěných příznaků, například pozadí, do celkového počtu, přidali do trénovacího setu obrázky, jejichž ground truth byla nulová.

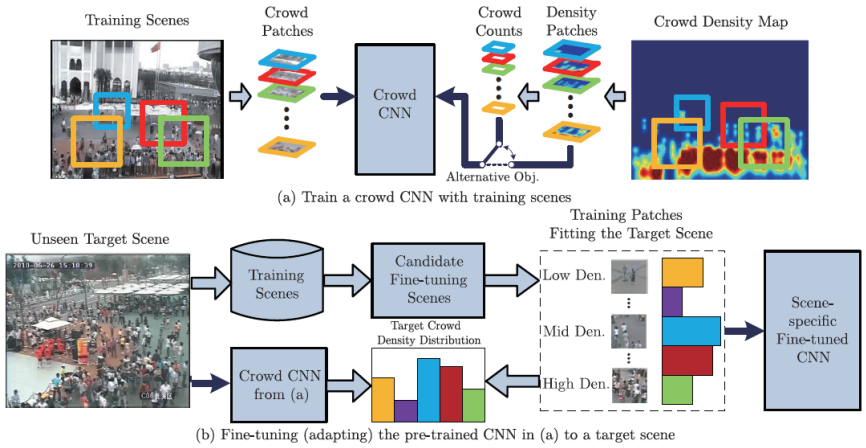


Obrázek 2.13: Kategorizace dosavadních typů konvolučních neuronových sítí pro počítání davu. Převzato z [29].

Výsledkem bylo, že dokázali klasifikovat obrázek do pěti kategorií od velmi vysoké hustoty po velmi nízkou.

Cross scene crowd couning

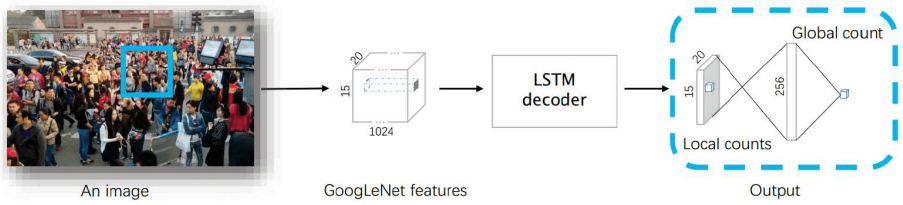
Úspěšnost u trénovaných modelů ale obecně klesala, pokud jim byla předložena jiná data, než ta, které obsahoval jejich trénovací dataset. Proto se Zhang et al. [37] pokusil tento problém překonat. Natrénoval dva různé modely (viz obrázek 2.14) - jeden na určení počtu a druhý na určení hustoty, aby kombinací obou těchto informací mohl dosáhnout lepšího výsledku. Dále pro vylepšení výkonu modelů použil data podobná cílové neviděné scéně a na základě nich bez anotací adaptoval model na nová data.



Obrázek 2.14: Scéma cross scene crowd couningu představené Zhang et al. [37].

Oproti dříve zmíněním metodám používajícím pouze části (patche) vstupního obrázku, Shang et al. [28] představil první end-to-end metodu na určení počtu. Navíc při přidání informace o kontextu byla schopná ignorovat pozadí a dosáhnout lepších výsledků. Její architektura je zobrazena na obrázku 2.15. Síť se skládá ze tří částí:

1. Přetrénovaný GoogLeNet
2. Long-short lime memory (LSTM) pro výpočet lokálního počtu
3. Plně propojená (fully connected) vrstva pro určení celkového počtu



Obrázek 2.15: Schéma end-to-end metody počítání davu, kterou představili Shang et al. [28].

Dalším krokem ve zlepšování architektur bylo použití více neuronových sítí paralelně za účelem zohlednění kontextu přiblížení. První takový pokus provedl Boominathan et al. [4], který zkombinoval jednu hlubokou a jednu mělkou plně propojenou neuronovou síť. Dosáhl tím způsobem možnost reagovat na různá přiblížení.

Zhang et al. [38] poté představil více sloupcovou architekturu (MCNN), kde díky třem různým filtrům ve sloupcích byla pokryta velká škála přiblížení. Dále popisuji MCNN v samostatné podkapitole.

Na podobném principu je založena také Hydra CNN představená Onoro and Sastre [25]. Základním kamenem této sítě je CCNN. Celá Hydra CNN má tři hlavy, kde na každou navazuje CCNN, poté tělo se skládá ze dvou plně propojených vrstev, na které navazuje jedno ReLu pro vytvoření mapy hustoty. Hydra CNN předvedla velmi dobré výsledky na datasetech s velmi variabilními scénáři.

Sam et al. [2] navrhl řešení, ve kterém nebude všechny sloupce trénovat na mixu všech trénovacích patchů, ale raději patche rozdělí podle jejich kontextu do skupin. Každou skupinu pak použije na vytrénování samostatné neuronové sítě. Nakonec byl natrénovan také blok "switch", který mezi těmito specializovanými sítěmi přepíná, podle toho jakou situaci rozezná. Trénování jednotlivých sloupců mělo více kroků. Přepínač byl založen na VGG-16.

Druhé dělení je podle způsobu, jakým metoda přistupuje ke vstupu při trénování:

Patch-based interface: Což je přístup, kde se vyřezávají části vstupního obrázku a ty se pak používají jako vstup pro CNN. S velikostí vyřezávaných obrázků lze manipulovat a ovlivnit tak výsledek. Poté co je model natrénovaný, tak predikce se vykoná na základě pohyblivého okna, které přejede celý obrázek. Celkový výsledek je spojení jednotlivých odhadnutých částí.

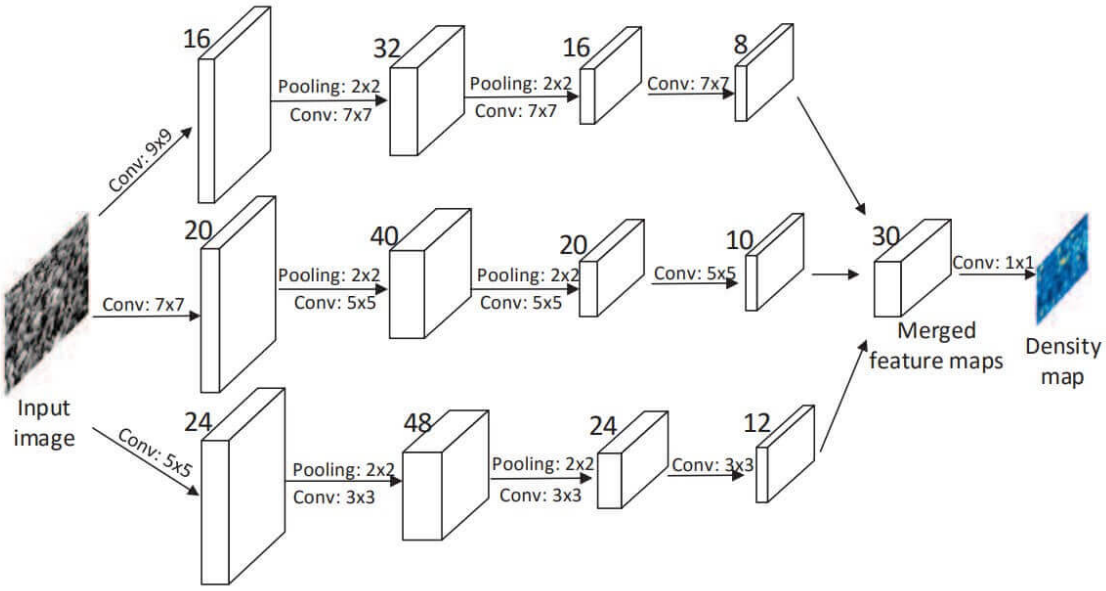
End-to-end training vyřazují využití sliding-window, které je náročné, místo toho vytváří rozhraní, kde se zpracovává přímo celý obrázek.

MCNN

V článku *Single-Image Crowd Counting via Multi-Column Convolutional Neural Network* se Zhang a další zabývají vytvořením architektury neuronové sítě, která by byla vhodná pro počítání davu z jednoho statického obrázku. V tomto úkolu bylo třeba překonat sérii komplikací.

První bylo, že na jednom obrázku lze velmi špatně provést oddělení na pozadí a popředí. Toho lze úspěšně dosáhnou pouze při znalosti pohybu, jak je tomu u analýzy videozáznamů a nebo se znalostí geometrie scény, která v tomto případě nebyla dostupna. Vzhledem k tomu, jak by to bylo obtížné, a tomu, že špatně provedená segmentace popředí by měla výrazný vliv na výsledek, se autoři rozhodli segnemtaci nedělat. Další velkou výzvou byla rozdílná hustota a rozmístění davu v jednotlivých snímcích. Především při velmi vysoké hustotě, kde metody založené na detekci začínaly být neúspěšné. Poslední komplikací, kterou je třeba vyřešit bylo, že při pohledu ze šikma, který je nejčastější, je potřebo počítat s tím, že velikosti hlav lidí se budou lišit. Proto je potřeba dokázat pracovat s příznaky rozdílných velikostí.

Právě k překonání těchto komplikací autoři navrhují nový přístup založený na více sloupcové neuronové síti (MCNN), kde každý sloupec bude reprezentovat filtr pro detekci příznaku rozdílné velikosti (viz obrázek 2.16). Vstupem je tedy obrázek a výstupem jsou mapy hustoty jednotlivých příznaků, které se poté spojí a vznikne jedna konečná mapa hustoty. Z té se poté spočítá počet pomocí integrálu. Sloupce v neuronové síti jsou celkem 3 pro malé/střední/velké příznaky lidí/hlav.



Obrázek 2.16: Schéma architektury MCNN. Převzato z [38].

Metriky přesnosti CNN pro počítání davu

Úspěch neuronových sítí pro počítání davu se hodnotí pomocí dvou metrik: MEA (*Mean Average Error*) a MSE (*Mean Absolut Error*),

$$MAE = \frac{1}{N}\sum_{1}^{N}|z_i - \hat{z}_i| \tag{2.1}$$

$$MSE = \sqrt{\frac{1}{N}\sum_{N}^{1}(z_i - \hat{z}_i)^2} \tag{2.2}$$

kde N je počet testovacích obrázků, z_i je opravdový počet lidí v $i$tém obrázku, a $\hat{z}_i$ je stanovený počet lidí v $i$tém obrázku. Obecně řečeno MAE vypovídá o přesnosti a MSE indikuje robustnost odhadů. K výpovědní síle této metriky je vždy potřeba dodat, na jakém datasetu podávala jaké výsledky. Je velmi pravděpodobné, že MCNN bude podávat nejlepší výsledky na Shanghaitech datasetu, na který byl navržen. Generalizovat pak bude nejlépe na datech podobných jeho tréninku. Podle článku [38] jsem výsledky shrnul do tabulky 2.1. V tabulce jsou vidět výsledky MCNN v souvislosti s průměrným počtem lidí na jednom obrázku daného datatsetu. Z toho lze odvodit, jaká chyba je pro jak hustý dav očekávatelná.

Dataset	**Prům. poč.**	**MAE**	**MSE**
ShanghaiTech P. A	501.4	110.2	173.2
ShanghaiTech P. B	123.6	26.4	41.3
WorldExpo	50.2	11.6	-
UCF CC 50	1279.5	377.6	509.1
UCSD	24.9	1.07	1.35

Tabulka 2.1: Tabulka shrnující výsledky MCNN [38] na různých datasetech.

V závislosti na datasetu NWPU [34] pak MCNN podává následující výsledky v porovnání s předními ostatními architekturami (viz tabulka 2.2). Je podstatné říct, že NWPU dataset je aktuálně nejkomplexnější dostupný dataset.

Metoda	MAE	MSE
MCNN [38]	232.5	714.6
C3F-VGG [9]	127.0	439.6
CSRNet [20]	121.3	387.8
CANNet [22]	106.3	386.5
SCAR [10]	110.0	495.3
SFCN+ [35]	105.7	424.1

Tabulka 2.2: Tabulka výsledků nejlepších metod na datasetu NWPU [34].

2.4 Existující aplikace v oblasti analýzy davu

V kapitolách 2.1 a 2.3 jsem se rozhodl věnovat tomu jak se při počítání davů v obraze dosahuje co nejlepších výsledků. V této kapitole se zaměřím na to, jakým způsobem lze vyhodnocované metody využít a jaká aktuálně existují řešení v praxi ve formě systému či aplikací.

Uživatelské úlohy v oblasti analýzy davu

Analýza davů protíná více různých vědeckých oborů jako je sociologie, psychologie, fyzika, počítačové vidění a veřejná bezpečnost. Možnost počítat a analyzovat davy má praktické uplatnění v mnoha oblastech. Zmíním zde ty nejpodstatnější, které popisuje Sindagi ve svém článku [29].

- **Bezpečnostní monitorovací systémy:** S velkým rozšířením bezpečnostních kamer je na místech velkého výskytu lidí, jako jsou stadiony, obchodní haly či letiště, možné provádět analýzu chování, analýzu přetížení, detekci anomálie a detekci události.

- **Správa katastrof:** Existuje mnoho scénářů, kdy se může čelit riziku způsobenému davem, například na politickém shromáždění či na hudebním koncertu nebo veřejné demonstraci. V takovém případě může pomoct analýza k včasné detekci přeplnění, případně k vhodnému řízení davu.
- **Návrh veřejných prostor:** V prostorách jako jsou vlaková nádraží, letiště, nebo obchodní domy může být použita analýza davu pro zkoumání pravidelného chování návštěvníků a pomocí čehož se může později vyvodit zlepšení návrhu objektu pro větší bezpečnost.
- **Sběr a analýza zpravodajských informací:** V tomto případě se jedná například o analýzu chování lidí v obchodních centrech a následné odvození například vhodného umístění produktu. Podobně se může například sledovat délka fronty, na kterou lze následně reagovat změnou počtu nasazených zaměstnanců.

iOmniscient

Společnost iOmniscient nabízí řešení pro komplexní analýzu chování davu v různých scénářích. Například analýzu návštěvníků v obchodních centrech, kde účelem je jak bezpečnostní riziko a předpověď potřeby evakuace, tak i touha získávat informace o chování nakupujících a jejich pohlaví atd. Především se soustředí na CCTV bezpečnostní kamery, prevenci přelidnění a realtime monitoring. Kromě početních výstupů nabízí možnost mapy zobrazení četnosti výskytu lidí a další analytická data.

Projekt LETSCROWD

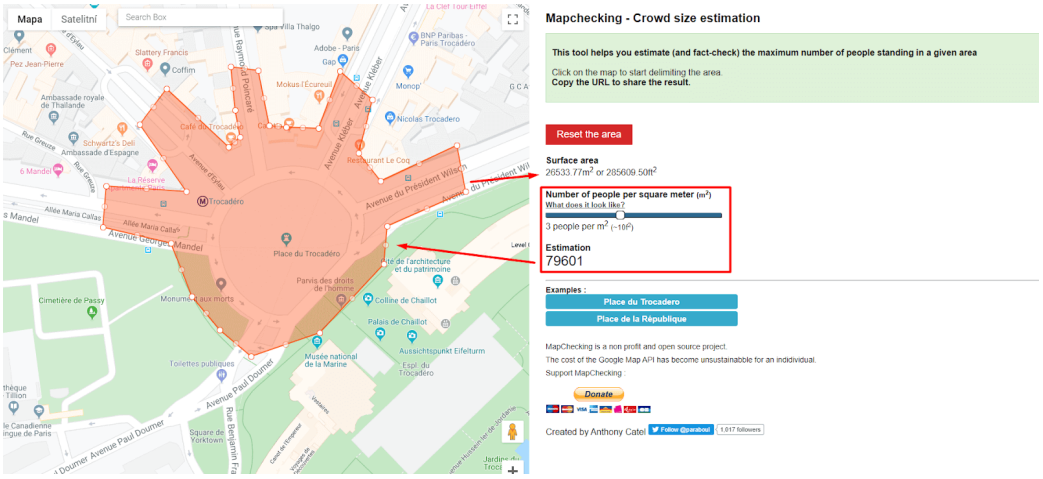
V rámci projektu LETSCROWD[5] vznikl komplexní nástroj pro dohled nad hromadnými akcemi. V rámci tohoto nástroje jsou sbírána data z bezpečnostních kamer pro předpovídání rizikových situací. Vše je konvertováno na serveru společně s dalšími informacemi jako je analýza příspěvků na sociálních sítích, hlášení policie a organizátorů. Používají také mapu, ale především pro výskyt kamer, vozidel či strážníků. Vše je zobrazováno v klientské aplikaci tak, aby vyšší autority měly dostatečný přehled o probíhající akci. Toto řešení je v současné době asi nejlepší nástroj užívající počítačové vidění v praxi.

Mapchecking

Aplikace Mapchecking [6] pro odhad celkového počtu lidí v davu používá pouze mapu bez dalších složitých metod. Uživatel zadá polygonem do mapy plochu, kde se dav vyskytoval. Poté musí správně odhadnout, jaká byla hustota daného davu. K tomu by podle autora měly pomoci modelové ukázky nasimulovaných hustot davu dostupné na webu Crowd Safety and Risk Analysis[7], který se zabývá porozuměním bezpečnosti davu. Výhodou aplikace je její jednoduchost a rychlost. Problém je ale v samotné nepřesnosti a nevěrohodnosti výsledků. I když má člověk dostupné fotografie daného davu, musí se na ně pouze podívat a odhadovat, jaká je hustota. Screenshot aplikace je na obrázku 2.17.

Právě na základě tohohle problému vznikala má myšlenka zapojit do řešení neuronové sítě, které počet a hustotu lidí dokáží odhadnou s dostatečně nízkou chybou.

[5] https://letscrowd.eu/
[6] https://www.mapchecking.com/
[7] http://www.gkstill.com/Support/crowd-density/625sm/Density6.html



Obrázek 2.17: Webová aplikace Mapchecking[9] pro pomoc odhadu počtu lidí na demonstraci. Na obrázku vidíte označené území, kde se dav pravděpodobně nacházel. Na slideru vpravo je nastavena odhadnutá hustota davu na metr čtverečný. Z těchto dvou údajů je odvozen celkový počet lidí na demonstraci.

2.5 Webové technologie

Pro tvorbu uživatelského rozhraní existuje široké spektrum nástrojů. V této podkapitole popíši, jaké možnosti se nabízejí. Zdůvodním, proč jsem si vybral pro tvorbu GUI právě webové technologie, a shrnu základní výhody a možnosti, které mi nabízejí.

Ve své aplikaci potřebuji propojit sílu modelu neuronové sítě s nějakým mapovým podkladem. Nyní je třeba vybrat, jaký nástroj poskytne možnost zobrazit mapu a pracovat s ní a zároveň vytvořit základní GUI.

GUI v jazyce Python

Vzhledem k tomu, že s modelem CNN se nejlépe pracuje v jazyce Python, první možností je vytvořit desktopovou aplikaci pomocí nějaké knihovny jazyka Python.

TKinter je standardní GUI knihovna pro Python. Lze díky ní vytvářet jednoduše a rychle aplikace. Nabízí možnost programovat objektově. Tkinter nabízí 15 různých ovládacích prvků, kterým se říká "Widgets". Právě z nich je výsledná aplikace složena. Je vhodný pro základní aplikace, u kterých je hlavní vstup, výstup a případně nějaké parametry. Byla by zde možnost využít statický obrázek mapy získané například z GoogleMaps API.

PyQt je nejpopulárnější binding pro Qt multiplatformní C++ framework. Qt nabízí komplexní možnosti pro vývoj desktopové aplikace. Aplikace využívající PyQt je třeba šířit pod licencí GNU GPL v3, tedy včetně zdrojového kódu, nebo lze využít méně stabilní ekvivalentní knihovnu PySide. Je zde také princip skládání GUI pomocí předdefinovaných widgetů, podobně jako v Tkinteru. PyQt je rozděleno do modulů (QtGui, QtWidgets, QtCore, atd.), kde další nadstavbové moduly mohou řešit například vykreslování SVG, SQL nebo síťovou komunikaci. Pomocí QtWebKit lze Qt integrovat s knihovnou Leaflet, která poskytuje interaktivní webovou mapu. O Leaflet knihovně píši podrobněji níže.

OpenCV, knihovna pro počítačové vidění, je designovaná pro full-scale aplikace. I zde je tedy možnost vytvořit uživatelské rozhraní. A to jak s využitím UI frameworků (Qt, WinForms, or Cocoa), tak i bez nich. V principu se ale samostatně používá spíše

pro rychlou vizualizaci výsledků, než pro vytvoření komplexní klientské aplikace. HighGUI modul v OpenCV nabízí vytváření oken, zobrazování obrázků a videa, trackbary, zpracování událostí myši a klávesnice a práci s kamerou. Zkrátka je zde to základní pro vizualizaci. Síla OpenCV je spíše v backendové části aplikace.

Nástroje pro práci s mapou

Dále je třeba prozkoumat, jakými způsoby je možné získat mapový podklad a pracovat s ním. To může být také určující pro výběr implementačních technologií.

Mapu lze získat z různých API a pomocí různých knihoven. Je třeba rozlišovat statickou a interaktivní mapu.

Statickou mapou se rozumí již neměnný obrázek. Takovou mapu poskytuje například **Google Maps Static API** [10]. V requestu se přesně specifikuje, jaké místo v mapě programátor požaduje, s jakým přiblížením a v jakém formátu. V odpovědi dostává statický obrázek mapy dle specifikací.

Interaktivní mapu pak představuje knihovna, která pro uživatele vytvoří mapu, se kterou přibližovat, oddalovat a hýbat s ní. Tato možnost pak přináší mnohem větší flexibilitu.

V Pythonu existují knihovny **ipyleaflet**[11], **gmap**[12] nebo **folium**[13], které umožňují nějakým způsobem vytvořit interaktivní mapu. V ukázkách jejich využití jsem spíše viděl využívání mapy jako vizualizační výstup aplikace a ne jako prostředníka mezi uživatelem a aplikací.

Knihovna Leaflet

Nejlepší vizulizační a zároveň interaktivní prostředky nabídla knihovna Leaflet[14]. Jedná ze o open-source JavaScriptovou knihovnu pro interaktivní práci s mapou. Při využívání této knihovny lze mapový podklad čerpat z různých API, kde některá jsou placená a jiná zdarma s povinností uvedení zdroje vpravo dole na mapě. Nejpoužívanějšími mapovými podklady jsou OpenStreetmaps[15], MapBox[16]. Leaflet je zdarma oproti **Google Maps JavaScript API**[17], u kterého se při počtu 25,000 načtení stránky za den platí. Přitom možnosti, které obě rozhranní nabízí jsou velmi podobné[18].

Knihovna umožňuje práci s interaktivními prvky v mapě jako jsou markery, polygony, přímky, obrázky a další. Komunitou je implementováno mnoho užitečných pluginů pro komplexnější problémy typu: transformace obrázku v mapě, vytváření teplotních map, drag and drop interakce a mnoho dalších. Všechny prvky jsou považovány za jednotlivé vrstvy. Lze je přidávat a odebírat.

Nad knihovnou leaflet pak existuje její High level nadstavba mappa.js[19], která umožnuje jednoduše vytvořit mapu a pomocí knihovny p5.js[20] využít vykreslovací funkce a vytvářet libovolné vizualizace do mapy.

[10] https://developers.google.com/maps/documentation/maps-static/intro
[11] https://ipyleaflet.readthedocs.io/en/latest/
[12] https://jupyter-gmaps.readthedocs.io/en/latest/
[13] https://python-visualization.github.io/folium/
[14] leafletjs.com
[15] www.openstreetmap.org
[16] https://www.mapbox.com/
[17] https://developers.google.com/maps/documentation/javascript/tutorial
[18] https://www.endpoint.com/blog/2019/03/23/switching-google-maps-leaflet
[19] https://mappa.js.org/
[20] https://p5js.org/

Tvorba GUI pomocí webových technologií

Vzhledem k tomu, jaké možnosti mi nabízí Leaflet a že práce s ním bude nejsnazší ve webovém prostředí, je třeba prozkoumat, jaké možnosti jsou pro tvorbu webového GUI. Na tuto otázku je odpověď v krátkosti jednoduchá - v principu neomezené. Web je nejpoužívanější nástroj pro tvorbu uživatelského obsahu a aplikací a velmi rychle se vyvíjí.

Na rozdíl od frameworků pro desktopové aplikace, kde se GUI většinou skládá z jednotlivých widgetu, ovládacích prvků, webová stránka je uspořádána na principu DOM, což je v podstatě XLM strom jednotlivých elementů. To, jaký význam mají jednotlivé tagy XML elementů, pak popisuje jazyk HTML.

Samotný HTML dokument ale vytváří pouze staticky nestylovaný obsah. To, co stránce vdechne život a udělá z ní aplikaci, je JavaScript. JavaScript je jazyk webových prohlížečů, který umožňuje manipulaci s objekty DOMu na straně klienta.

Pro dokončení vizuálně hezké aplikace je třeba upravit vzhled, velikosti a uspořádání jednotlivých prvků. V tomto případě lze použít atributy html tagů (nepoužívá se) a nebo CSS. CSS je jazyk popisující styl HTML dokumentu.

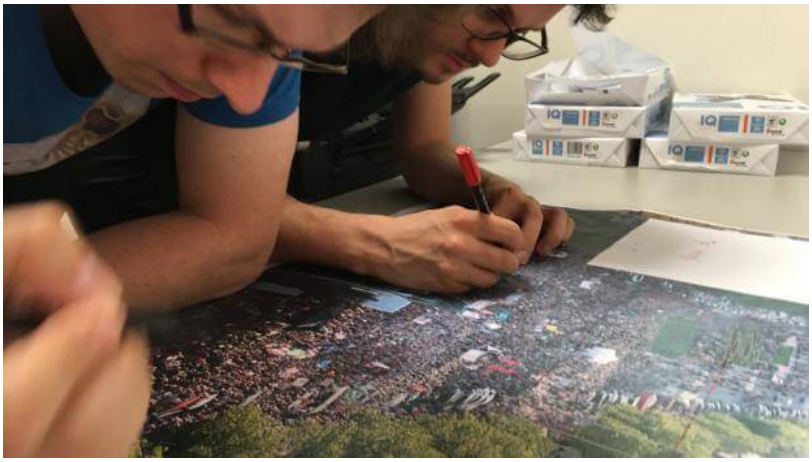
Vzhledem k tomu, že některé CSS, HTML a JS konstrukce byly používané velmi často, vznikla knihovna komponentů Bootstrap. Díky Boostrapu lze velmi elegantně vyřešit layout stránky, základní stylování i pokročilejší animace. Pokud je třeba nastylovat nějaké konkrétní elemety velmi specificky, použije se navíc i samostatné CSS.

Kapitola 3

Návrh systému využívající CNN pro odhad počtu osob z více fotografií

V této kapitole se zaměřím na konkretní uživatelský problém počítání davu a tím je odhad celkového počtu lidí na demonstraci po jejím proběhnutí z dostupných fotografií. Tento odhad provádí především novináři, ale také policie.

Popíši zde návrh řešení, které mohlo pomoci integrovat dostupné nástroje a řešit tak úlohu počítání davu ve scénáři již proběhlé demonstrace. Popíší jednotlivé podproblémy a způsoby, jakými je možné je řešit. Představená aplikace bude mít architekturu klient-server a propojí topologickou mapu a neuronovou síť.



Obrázek 3.1: Ruční metoda počítání davu. Redaktoři označují hlavy červeným fixem, aby spočítali celkový počet lidí na demonstraci. Převzato z iROZHLAS.cz[2].

3.1 Popis problému

Uživatel chce spočítat celkový počet lidí na demonstraci či jiné hromadné akci. V případě novinářů pak použije toto číslo v článku o dané akci a v případě policie v oznámení o proběhlé akci. Zpravidla má k dispozici sérii fotografií z dané akce. Pokud je ale akce

rozsáhlá, není možné v dostatečné kvalitně pořídit její fotografii pouze na jeden snímek a proto uživatel potřebuje pracovat s více souvisejícími snímky. Ty se pravděpodobně budou překrývat.

Ruční metoda počítání davu

O tom, jakým způsobem se to řeší například v České Republice, je napsán hezký článek na stránkách iROZHLAS.cz[3].

Pro představu popíši, jak by uživatel postupoval tzv. "ruční metodou"(viz obrázek 3.1). Pravděpodobně by si vytiskl všechny fotografie ve velkém formátu a poté je vůči sobě začal skládat, jako puzzle. Když by se mu podařilo vyskládání, vzal by do ruky červenou fixu a začal tečkovat hlavy jednotlivých účastníků a počítat. Tato metoda by vedla pravděpodobně k docela přesnému výsledku, ale za cenu velkého úsilí.

Potřeby uživatele

Kvůli komplikacím popsaným výše navrhuji následující inovační kroky:

- Ušetřit uživatele od počítání hlav.
- Ulehčit umisťování fotografií, tzv. puzzle - poskytnutím mapového podkladu.
- Poskytnout více než jenom číslo celkového počtu, ale i další data.

Představená aplikace mu umožní nahrát a rozmístit fotografie v mapě a vypočítat hustotu davu v jednotlivých fotografiích. Pro rozmístěné fotografie je provedena korekce počtu u překrývajících se. Výsledkem je vizualizace rozmístění a počet lidí na demonstraci.

Řešení celé aplikace lze rozdělit na následující podproblémy:

- Zpracování samotných fotografií
- Využití mapového podkladu jako místo pro fotografie
- Jaké úlohy bude uživatel provádět, aby dosáhl výsledku
- Jak vyřešit překrytí fotografií, aby byl odhad korektní

K řešení mě inspirovala aplikace Mapchecking[4], která využívá mapu pro odhad maximálního možného počtu lidí, kterého je možné dosáhnou při dané hustotě. Aplikaci jsem více popsal v předchozí kapitole 2.4.

Uživatel tedy bude potřebovat nahrát fotografie, umístit fotografie do mapového podkladu, interagovat jak s mapovým podkladem, tak i s umístěnými fotografiemi, ukládat si rozdělanou práci a vidět výstupní informace o počtu lidí.

[3] https://www.irozhlas.cz/zpravy-domov/ani-30-ani-20-tisic-spocitali-jsme-demonstranty-proti-babisovi_1705111741_cib

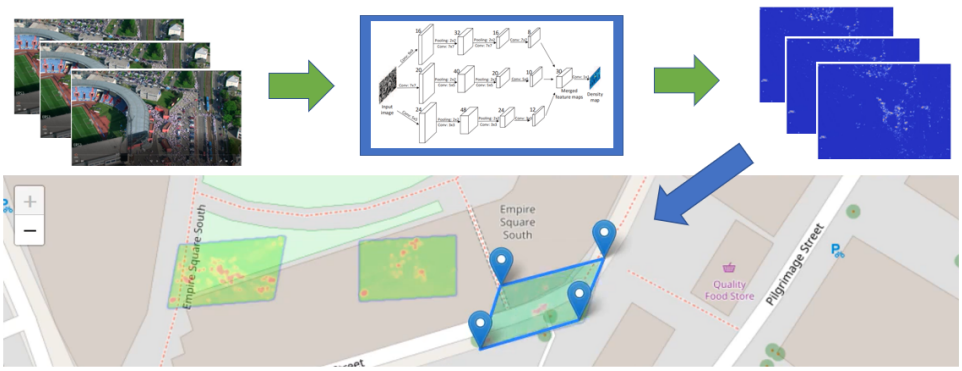
[4] https://www.mapchecking.com/

3.2 Architektura aplikace

Co se týče zpracování samotných fotografií, z předchozí kapitoly 2.3 je jasné, že v současné době je nejvýhodnější využití neuronové sítě. Tu lze pro potřeby návrhu vnímat jako modul, který na vstupu dostane fotografii a na výstupu vrátí mapu hustoty lidí, která v sobě zároveň nese počet lidí ve fotce.

Tento modul bude výpočetně nejnáročnější součástí celého systému, a proto je vhodné ho vložit na samostatný server. Architekturu aplikace tedy navrhuji jako klient-server s využitím webových technologií. Využití webového klienta je navíc výhodné, neboť nabízí využití kvalitních knihoven pro tvorbu interaktivní mapy a práci s ní. Interakce s uživatelem bude realizována na klientovi a vlastní výpočet pomocí CNN na serveru. Další výhodou zvolení architektury klient-server je to, že pokud je model dílo, které je třeba chránit před odcizením, na serveru se toho lehce dosáhne.

Klientská část umožní nahrání fotografií a jejich umístění do mapy. Tato data je třeba synchronizovat se serverem. Ten bude počítat hustotu davu, převádět na počet osob, transformovat fotografie a provádět detekci překryvů fotografií, díky které poté vypočte redukovaný výsledný počet osob. Výsledky pak budou opět synchronizovány s klientskou částí, kde budou uživateli vizualizovány. Schéma architektury je zobrazeno na obrázku 3.2.



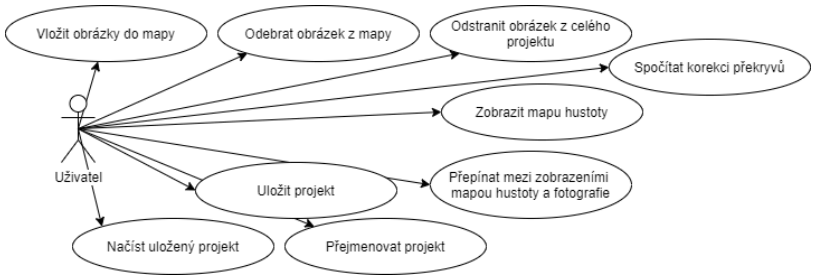
Obrázek 3.2: Ilustrační schéma architektury aplikace.

3.3 GUI pomocí mockupu

V této podkapitole se budu věnovat návrhu GUI, rozepíšu, proč a jaké prvky v něm musí být, jaké jsou důvody jejich konkrétního uspořádání.

Užívání aplikace z pohledu uživatele lze znázornit usecase diagramem (viz obrázek 3.3). Data, která poskytuje uživatel, jsou fotografie, které z místa akce pořídil. Dále uživatel musí mít možnost dostat se v mapě na místo, kde se akce konala. Uživatel potřebuje pracovat s obrázky, konkrétně: nahrát je, odstranit je, vložit je do mapy, ale i vyjmout. Pro daný obrázek si potřebuje zobrazit k němu vytvořenou mapu hustoty. V prostoru mapy pak potřebuje přepínat mezi zobrazením vloženého obrázku jako mapy hustoty, nebo jako fotografie.

Uživatelská aktivita v rámci jedné akce (např. demonstrace na náměstí) je spravována jako projekt.



Obrázek 3.3: Usecase diagram uživatele aplikace.

Aplikace umožňuje vybrat fotografie, ty umístit přesně do mapy a vidět výslednou hustotu davu a odhad počtu lidí.

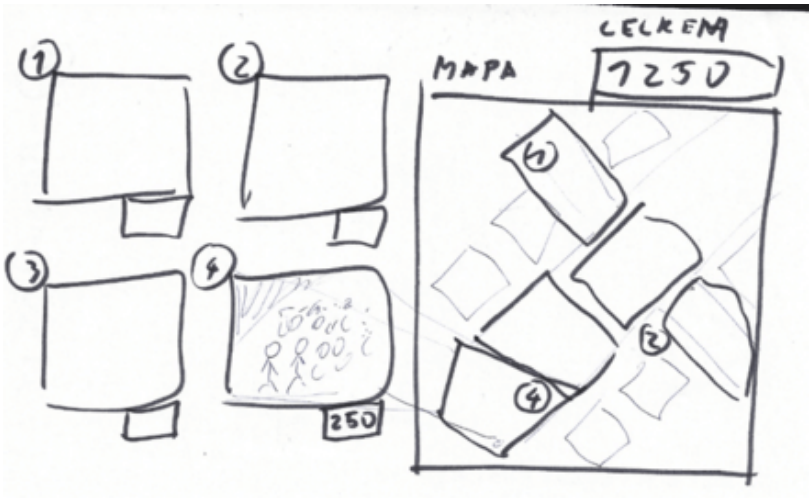
V GUI budou tedy figurovat tyto základní prvky:

- Tlačítko vložení fotografií
- Seznam fotografií
- Vybraná fotografie
- Mapa
- Informace o vybrané fotografii
- Informace o celkovém počtu lidí v mapě

Původní ilustraci myšlenky zakresluje náčrt na obrázku 3.4, z tohoto obrázku je odvozený wireframe, tvořený již v editoru viz obrázek 3.5.

Uživatel potřebuje mít možnost vložit do aplikace obrázky, proto zde musí být prvek pro výběr obrázků pro vložení z počítače. V návaznosti na to pak místo, ve kterém se nahrané obrázky objeví, aby uživatel vizuálně poznal, že je úspěšně nahrál. Toto místo bude reprezentováno malou galerií náhledových obrázků.

Z galerie obrázků je potřeba umožnit vybírat na přeskáčku obrázky a v případě potřeby dále s daným vybraným obrázkem pracovat. Proto navrhuji zobrazovat aktuálně vybraný obrázek ve zvětšené podobě. Ke zvolenému obrázku je také třeba zobrazovat korespondující mapu hustoty ve chvíli, kdy je dostupná.



Obrázek 3.4: Ruční náčrt návrhu aplikace.

Uživatel potřebuje umístit obrázek do mapy. V mapě bude obrázek určený čtyřmi krajními body. Uživatel ale tyto body nepotřebuje postupně naklikávat. Proto navrhuji urychlení - pomocí tlačítka "přidat do mapy"se obrázek objeví uprostřed mapy a uživatel tam s ním bude moci dále manipulovat.

Aby uživatel mohl pohodlně manipulovat s obrázky v mapě, může je posouvat jako celek, nebo upravovat pouze pozici rohových bodů, čímž je mu umožněno obrázek transformovat. Je potřeba co nejlépe umožnit správně umístit obrázek. Proto navrhuji, že uživatel bude mít možnost přepínat mezi zobrazením obrázku v mapě jako mapu hustoty a jako průhlednou fotografii.

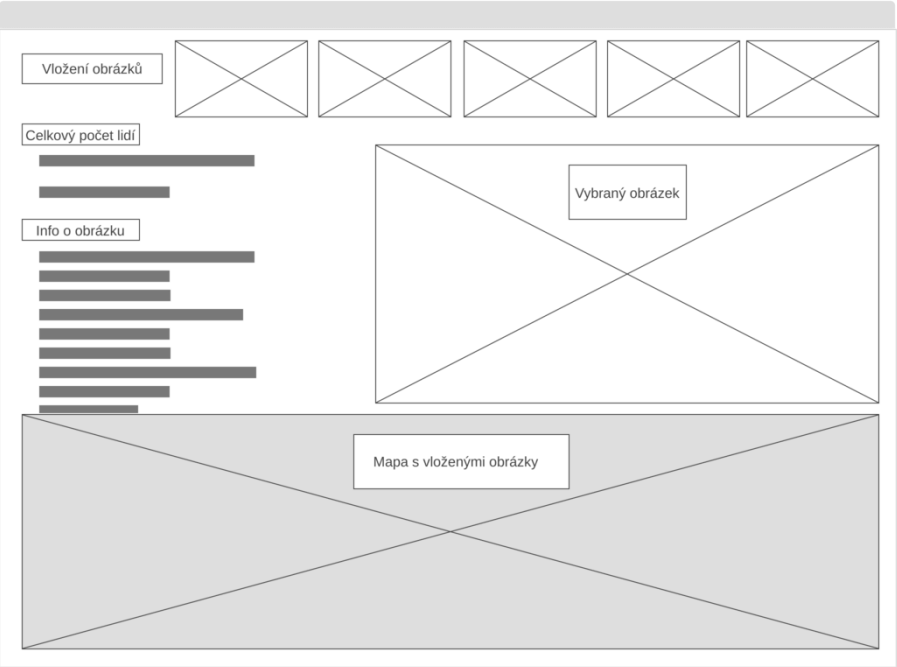
Je potřeba také umožnit uživateli vyjmout zpět obrázek z mapy a nebo ho dokonce odstranit z celého projektu. Jinak by při špatném vložení musel začínat znovu od začátku.

V případě velkého množství překryvů by odhadovaný celkový počet lidí na dané akci byl moc velký. Proto jsem navrhl metodu pro korekci překryvů, kterou může uživatel spustit a dostat tak přesnější odhad.

Jednou z vlastností aplikace, kterou je třeba zařídit, je persistence. Uživatel musí být schopný si projekt pojmenovat, uložit, načíst a případně i smazat. Tyto akce bude zajišťovat sada ovládacích tlačítek. Data se pak budou ukládat na serveru. Je potřeba, aby při smazání projektu aplikace vypadala jako při prvním spuštění, tedy byla prázdná. Akce přejmenování projektu a načtení uloženého projektu budou potřebovat prostor pro zobrazení. To jsem se rozhodl místo modálního okna vyřešit vysouvacím panelem. Při načítání uloženého projektu se může uživatel nacházet na mapě v jiném místě než projekt. Proto je třeba při uložení uchovat bod, kde se akce odehrává, aby se tam pohled při načtení mohl přesunout.

Projekt si uživatel může uložit a později se k němu vrátit a znovu s ním pracovat. Uživatel nejdříve pracuje na projektu, který se neukládá. Pokud si ho chce zachovat, pojmenuje ho a uloží. Díky tomu ho bude moct později načíst.

Tím, že se umístila mapa hustoty do topografické mapy, je možné vypočítat plochu, na které se dav vyskytuje. To může být zajímavý údaj, ze kterého lze vyvodit hustotu vztaženou k metrům čtverečným.



Obrázek 3.5: Wireframe původního GUI.

3.4 Datové struktury

Díky tomu, že architektura bude rozdělená na klient a server, každá z těchto dvou jednotek bude uchovávat jiná data. Tato data budou mít ale společné jádro a vzájemně se budou ovlivňovat. Na obou stranách bude totiž datová struktura obsahovat kolekci objektů fotografií, rozdílné bude jen to, co objekt fotografií musí obsahovat za data na klientské straně a co na serverové.Data se vždy udržuji na místě, kde se mění a druhá strana je o změně informována.

Datová struktura klienta

Klient bude zodpovídat za co nejlepší vizualizaci, data budou tedy vizualizačního charakteru a obsáhnou jen to potřebné pro vhodné zobrazení, tedy prvky související s mapou, odkazy na obrazová data, mapy hustoty upravené do formy vizuálně hezkého obrázku, získané hodnoty od serveru, které je třeba zobrazit.

V klientu je vždy otevřený jen jeden projekt současně, ten se skládá z objektu obrázků a vizuálních informací o nich. Konkrétně je o každém obrázku potřeba uchovávat na straně klienta jméno, odkaz na obrázek fotografie, odkaz na obrázek mapy hustoty, dále data o prvcích mapy tedy markery, čtyřúhelník obrysu, objekt transformovaného obrázku v mapě a nakonec celkový počet lidí na obrázku. Pak už se na klientu uchovávají jen obecná data jako je celkový počet lidí v celém projektu. Datová struktura je vizualizována na obrázku 3.6.

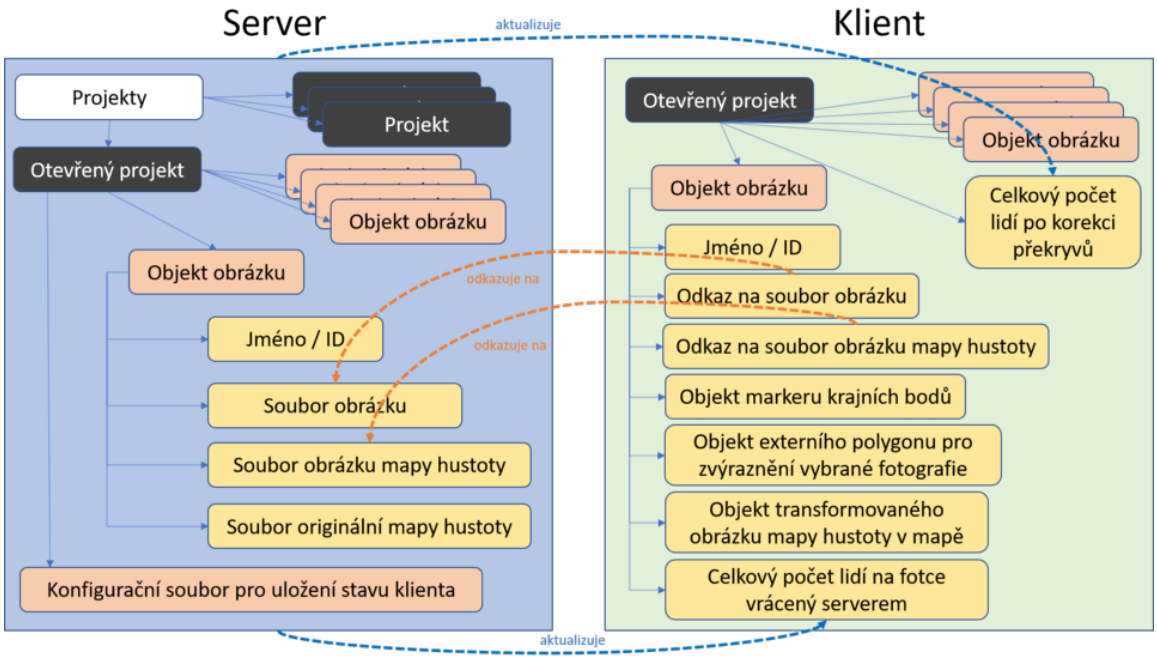
Datová struktura serveru

Server na druhou stranu bude udržovat data potřebná pro výpočet překryvů - originální mapy hustoty vrácené modelem, fotografie, model samotný atd.

Ukládání dat na serveru je řešeno přes souborový systém, bez využití databáze. Vždy když server získá, nebo vypočítá data, uloží si je do složky odpovídajícího projektu. Takto ukládaná data jsou originál obrázku, mapa hustoty vrácená modelem, aby už jí znovu nemusel počítat, a obrázek mapy hustoty, který je pak poskytován klientovi.

V paměti si pak server udržuje jen to, s čím právě počítá. Server vždy dokáže říct, jaké projekty jsou u něj uložené. Otevřený je v jednu chvíli právě jeden projekt. Projekt se skládá ze seznamu fotografií a má své jméno, které je podstatné, jelikož určuje cestu k souborům. Pro každý projekt je možné provést korekci počtu lidí z důvodu překryvů a tu odeslat klientovi.

Na serveru je ale také třeba vyřešit, jak uchovat stav klienta, pokud uživatel uloží projekt. Uživatel musí být schopný načíst znovu uložený projekt a vidět ho v takovém stavu, v jakém ho viděl naposledy. Pro uchování tohoto stavu se použije konfigurační soubor, který bude vystihovat stav klienta a bude obsahovat všechny potřebné informace k obnovení stavu, v jakém se nacházel při uložení. Data tohoto souboru budou odvozena od datové struktury, kterou si udržuje klient, ale s drobnými úpravami. Je potřeba uchovat seznam všech obrázků a o nich následující informace: jméno, URL obrázku, URL obrázku mapy hustoty, počet lidí na obrázku. Dále je ještě třeba o obrázcích uchovat to, jak byly umístěny v mapě. V klientu to jsou markery, čtyřúhelník ohraničující obrázek v mapě a objekt transformovaného obrázku v mapě. Všechno toto stačí ale uložit jako čtyři GPS souřadnice rohových bodů, zbytek se totiž dá zpětně vygenerovat. (Navíc v praxi to budou velmi zanořené mapové objekty včetně funkcí.) Z celého projektu se pak do konfiguračního souboru zapíše ještě celkový počet lidí, jméno projektu a GPS souřadnice centra zobrazené části mapy, aby se uživatel objevil na stejném místě jako předtím.



Obrázek 3.6: Schéma datových struktur udržovaných na klientu a na serveru.

3.5 Komunikace mezi serverem a klientem

Mezi serverem a klientem je třeba aktualizovat informace. V této kapitole popíši, jakým způsobem bude jejich komunikace probíhat, jaká data posílají a jaká by měla být odpověď druhé strany. Jednotlivé scénáře komunikace jsou vždy vztažené k akci uživatele, kterou právě provedl.

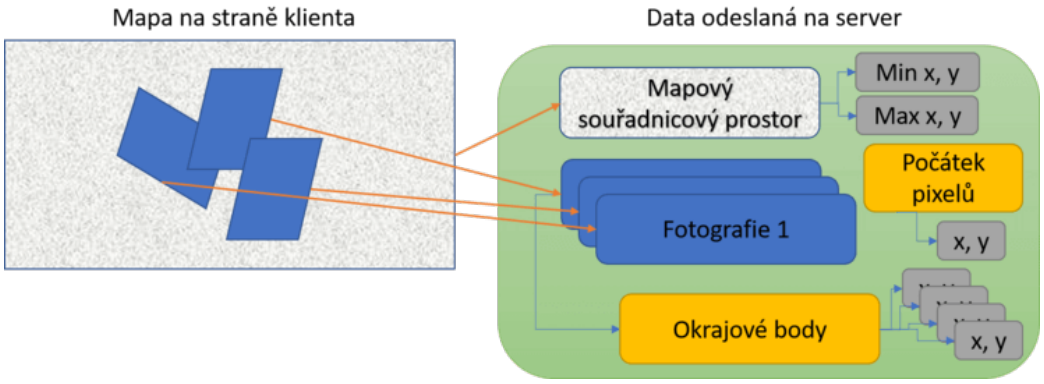
Komunikace při vložení obrázku do mapy

Uživatel nahrál jednotlivé fotografie, se kterými bude pracovat, teď první z nich chce umístit do mapy. Proto je třeba odeslat požadavek na server pro vygenerování mapy hustoty a počtu lidí na fotografii. Serveru se odešle fotografie a jméno projektu. Na serveru je fotografie uložena a pomocí CNN vypočítána mapa hustoty, tato mapa hustoty se také uloží na serveru. Dále se zde vygeneruje obrázek mapy hustoty vhodný pro vizuální zobrazení. Ten se uloží na serveru. V odpovědi pak server odesílá počet lidí na fotografií a odkaz na uložený obrázek mapy hustoty.

Komunikace při korekci překryvu pro výpočet celkového počtu lidí na akci

Pokud uživatel již umístil do mapy více fotografií, je potřeba spočítat celkový počet lidí na akci ze všech fotografií dohromady. Tento počet ale nebude pouhým součtem, musí se vzít v potaz překryvy a číslo redukovat. Metoda korekce je podrobně rozepsána v kapitole 3.6, zde popíši pouze to, jaká data kam a kdy putují.

Při každé aktualizaci vzájemné polohy fotografií v mapě se serveru odesílají rohové body všech obrázků a informace o velikosti souřadnicového systému mapy, na které se nachází. Souřadnice rohových bodů obrázků jsou ale vztažené k počátku, ve kterém se mapa objevuje při prvním načtení. Proto je potřeba tento počátek také odeslat serveru a následně upravit hodnoty bodů tak, aby odpovídaly a trefily se do mapového souřadnicového prostoru. Díky tomu pak získává server všechny potřebné informace pro to, aby mohl vypočítat korekci celkového počtu. Server vrací celkový vypočítaný počet v odpovědi. Zasílané informace shrnuje obrázek 3.7.



Obrázek 3.7: Schéma dat, která je třeba odeslat na server pro výpočet korekce překryvu.

Komunikace pro správu projektu

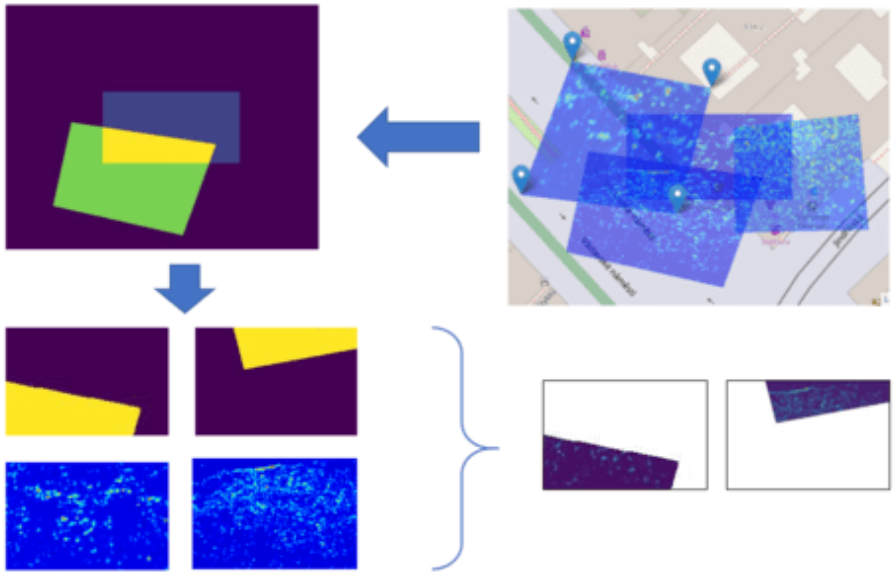
Změna jména projektu se může zdát jako nepodstatná akce, ale je zde třeba provést pár důležitých akcí. Klient posílá serveru request s novým jménem, na serveru se aktualizuje jméno projektu a také název složky projektu v souborovém systému. Pokud se akce provede úspěšně, aktualizuje se jméno i u klienta a aktualizují se URL cesty k souborům obrázků a obrázkům map hustoty.

Při uložení projektu se na klientské straně připraví potřebná data pro konfigurační soubor. Ta se odesílají na server, kde se konfigurační soubor buď vytvoří, nebo aktualizuje. To, jaká data jsou potřeba pro konfigurační ukládací soubor, jsem popsal v předchozí kapitole 3.4.

Pokud klient chce načíst již uložený projekt, jedná se o dvoufázovou komunikaci. Nejdříve klientská strana požádá o seznam uložených projektů. V odpovědi se vrátí seznam projektů, které je možné načíst. Ten se zobrazí uživateli, uživatel vybere, který projekt chce načíst, a tím odesílá klientská strana request o konfigurační soubor daného projektu. Podle tohoto souboru se zobrazí požadovaný stav UI (obrázky, jejich umístění, mapa, info ...).

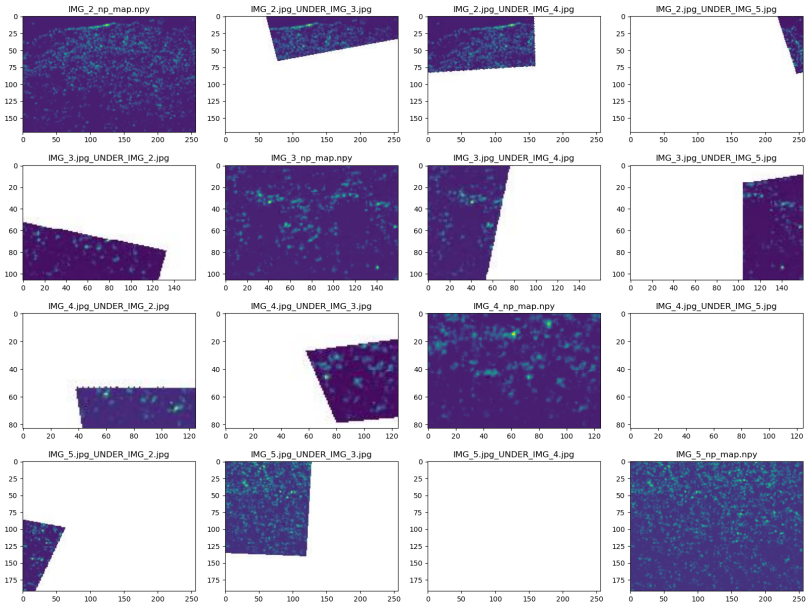
3.6 Metoda pro korekci překrývání

Model na serveru odhaduje počet lidí pro celou fotografii vytvořením mapy hustoty. V řadě případů se ale fotografie překrývají a výsledný počet lidí na akci by byl tedy zkreslený, proto je třeba provést korekci počtu osob v místech překryvů. Celkový počet by neměl být jen součet všech hodnot samostatných obrázků, ale musí brát z překryvů poměrnou část.



Obrázek 3.8: Schéma získání a použití masky pro korekci překryvu.

Je potřeba nalézt překryvy pro jednotlivé kombinace fotografií. Pro jednotlivé fotografie existují korespondující mapy hustoty, ze kterých se odvozuje počet lidí na fotografii. Je potřeba vypočítat počet lidí i pouze v místech překryvu, aby se vědělo, jakou část je třeba redukovat. To se zjistí tím, že se získá pouze ta část mapy hustoty, která je v překryvu. Dále je vhodné vytvořit si matici těchto překryvů, pro lepší přehlednost všech kombinací.



Obrázek 3.9: Vizualizace matice jednotlivých překryvů.

Řádky a sloupce matice reprezentují fotografie a buňky matice odpovídají počtu osob v překryvu těchto fotografií. Vizualizační forma toho, jak matice vypadá, je na obrázku 3.9, obrázek 3.11 pak zobrazuje matici s číselnými hodnotami.

Redukci pak lze realizovat odečtením překrývajících se oblastí (součet hodnot pravé horní části matice od součtu všech počtů osob na fotografiích (diagonála)). V rovnici 3.1 se tedy od diagonály D (žlutá ve zjednodušeném příkladě 3.11) odečítá horní trojúhelník matice. Ten se získá tak, že od celé matice se odečte diagonála a následně se tento počet vydělí dvěma. Toto dělení vyřeší to, že horní a spodní trojúhelník mají rozdílné hodnoty, jelikož dojde k zprůměrování.

Výsledný počet se pak získá pomocí vzorce:

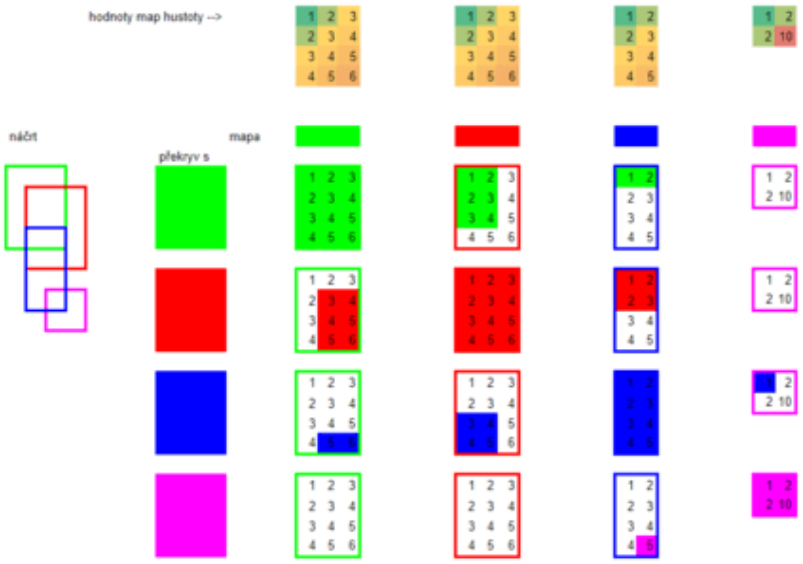
$$SUM = D - \frac{M - D}{2} = \frac{1}{2} \cdot (3 \cdot D - M) \tag{3.1}$$

kde SUM je odhad s korekcí, D je součet prvků diagonály (zároveň také odhad před korekcí), M je součet celé matice.

Aby server věděl, jak jsou fotografie umístěné, je třeba zaslat polohu všech obrázků v mapě a s nimi i hranice mapového souřadného systému, na kterém se vyskytují. Nejdříve se vytváří masky pro překryvy transformovaných obrázků. Transformovaná mapa hustoty by ale dávala zkreslený počet. Je tedy třeba provést inverzní transformaci masky tak, aby byla použitelná na originální tvar mapy hustoty. Tento postup je znázorněn na schématu 3.8.

Příklad výpočtu korekce překrytí

Na následujícím příkladu bych chtěl zjednodušeně ukázat, jakým způsobem probíhá výpočet korekce překrytí. Na obrázku 3.10 v části s náčrtem jsou znázorněné mapy, které jsem pro názornost zjednodušil. Překrývají se zde zelená, červená, modrá a růžová mapa. Rozměry každé mapy byly zjednodušeny na pár čtverečků oproti reálným příkladům. Každý



Obrázek 3.10: Vizualizace zjednodušeného příkladu překrytí. Vlevo je náčrt jakým způsobem se zjednodušené mapu překrývají. Nahoře jsou hodnoty v jednotlivých pixelech map. Největší část obrázku zabírá matice znázorňující jednotlivé překryvy.

čtvereček má svou hodnotu, kterou jsem stanovil v tomto zjednodušeném příkladu sám. Prvním krokem ve spočítání příkladu je získání všech kombinací překryvů. Tyto kombinace je vhodné pro názornost umístit do matice, která je také znázorněna na obrázku 3.10. Pro každou kombinaci se dále vypočítá celková hodnota všech pixelu, v tomto případě čtverečků. Tato hodnota se umístí do matice viz obrázek 3.11, se kterou se počítá dále. V této matici žlutá diagonála znázorňuje hodnoty samotných obrázků, její součet by tedy znamenal počet bez korekce překryvů. Horní a spodní oranžový trojúhelník pak reprezentuje překryvy. Pak se podle vzorce 3.1 vypočítá redukovaný počet.

	zelená	červená	modrá	růžová
zelená	42	15	3	0
červená	27	42	8	0
modrá	11	16	24	1
růžová	0	0	5	15

D	123
M	209
SUM = 0,5*(3*D-M)	80

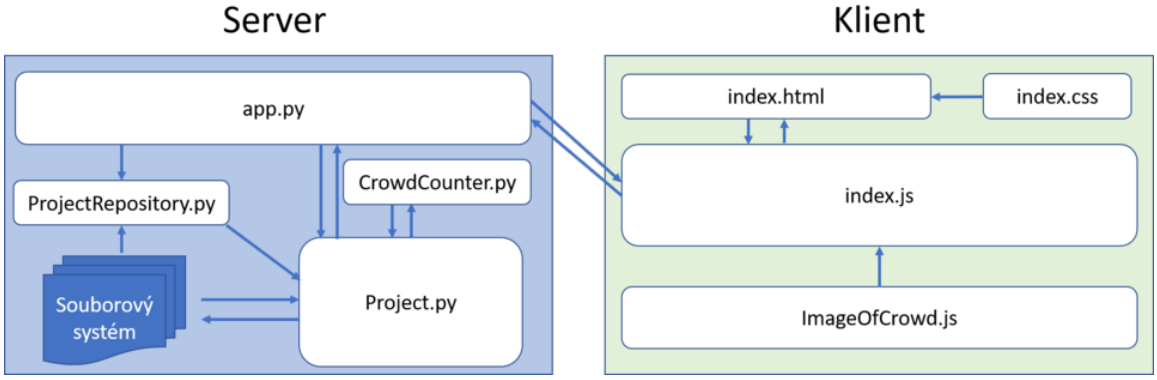
Obrázek 3.11: Matice a celkový výpočet pro ukázkový příklad. Sloupce a řádky znázorňují zelený, červený, modrý a růžový obrázek, žlutě je diagonála (počet lidí v obrázcích) a oranžově je horní a dolní trojúhelník (počet lidí v místech překryvu).

Kapitola 4

Aplikace pro odhad počtu osob na akcích

Navržený systém implementuji do uživatelské aplikace. V této kapitole popíši nejdůležitější algoritmy či funkce serverové a klientské části. Dále zde zpracuji přehled a porovnání dostupných datasetů a vytvořím svůj vlastní dataset. Datasety následně využiji na otestování MCNN modelu a i celé aplikace. Nakonec zde popíši, jakým způsobem probíhalo uživatelské testování pro zlepšení GUI.

Aplikaci jsem rozdělil na moduly pro lepší orientaci v kódu (viz obrázek 4.1). Na obrázku je také naznačeno, jakým způsobem spolu moduly komunikují.



Obrázek 4.1: Schéma znázorňující moduly, jejich rozsáhlost a komunikaci.

4.1 Serverová část

Server je hlavním výpočetním a úložným modulem aplikace. Klient tedy řeší už jen vizualizaci a interakci. Jednou z hlavních součástí serveru je i natrénovaný model MCNN.

Server je implementován v jazyce Python především z důvodu jeho flexibility v oblasti počítačového vidění (OpenCV, NumPy) a také v oblasti neuronových sítí (TensorFlow, Keras). Komunikaci s klientem zajišťuje lehký WSGI[1] webový aplikační framework Flask[2], který je jednoduchý a splňuje všechny potřeby pro nasazení natrénovaného modelu. Na

[1] WebServerGatewayInterface
[2] https://pypi.org/project/Flask/

server přichází požadavky pro přepočítání obrazových vstupů na mapy hustoty. Ty se pak odesílají k zobrazení zpět na klientskou stranu. Výhodou serveru je, že model se nedává na web veřejně přístupný, pokud by ho bylo třeba udržet v soukromí.

Server je rozdělen na čtyři části (viz obrázek 4.1):

- **app.py** - Jedná se o jediný spustitelný skript. Tato část vytváří serverové rozhraní a řeší zpracování požadavků klienta. Více o tomto modulu píši dále.
- **CrowdCounter.py** - Je objekt obsahující model a další pomocné funkce pro vytvoření mapy hustoty z obrázku.
- **ProjectRepository.py** - Tato třída spravuje projekty uložené na serveru v souborovém systému.
- **Project.py** - Třída, která reprezentuje projekt. Nacházejí se zde funkce algoritmu pro výpočet korekce překryvu, funkce pro ukládání, načítání a přejmenování projektu.
- **utils.py** - Pomocné funkce.

Natrénovaný model pak na serveru běží pomocí knihovny Keras. Při spuštění serveru je model načten z json souboru a nahrají se do něj váhy z *weights.h5*. Poté je spuštěn server na localhostu. Zobrazení klienta tedy proběhne po zadání lokální ip adresy a Request-URI indext.html[3].

Komunikace se serverem

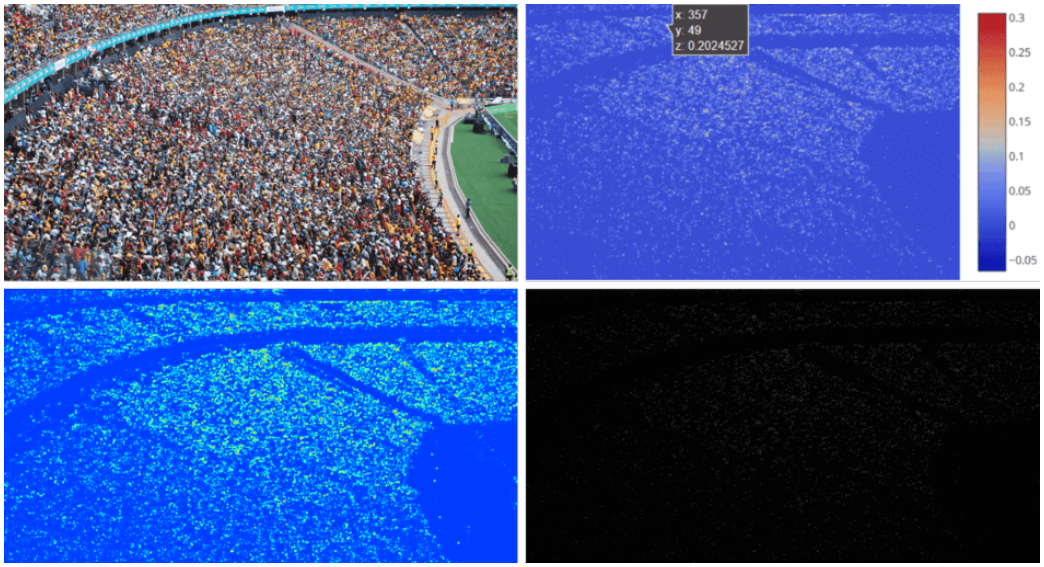
Nejběžnější komunikace se děje při přidání obrázku do mapy (viz obrázek 4.2). Klient odesílá HTTP POST request, který obsahuje jméno obrázku a zakódovaný obrázek. Ve funkci *sendmap()* se zpracuje request, vytvoří se mapa hustoty za použití modelu v **CrowdCounter.py**. Mapa hustoty se ukládá na serveru a v odpovědi se odesílá její URL a počet lidí na fotografii. Po této akci obrázek svým počtem lidí přispívá do celkového počtu lidí v projektu.

Obrázek 4.2: Ukázka komunikace mezi klientem a serverem při vložení obrázku do mapy.

V **CrowdCounter.py** je ke zpravování obrázků využita knihovna OpenCV. Ta se využije především k preprocesingu obrázku před tím, než je vstupem natrénovaného modelu a

[3] http://127.0.0.1:5000/static/index.html

pro korekci překryvů. Samotný model vrací mapu hustoty v nevhodném formátu pro zobrazení, proto je nejdříve zpracována na pole polí. Dále je zde funkce *generate_density_map_image*, která z této mapy hustoty vytvoří obrázek lépe pochpitelný pro člověka. To se udělá převodem a přeškálováním na stupně šedé a poté obarvením pomocí *cv2.applyColorMap*. Výstup, který jde přímo z modelu, má totiž velmi nízké hodnoty pixelů od 0 do 0.3 (počtu lidí na pixel) jak je vidět na ukázce 4.3.



Obrázek 4.3: Ukázka mapy hustoty v různých modifikacích. Zleva dolů: Fotografie, Mapa hustoty zobrazená v grafu včetně opravdových hodnot, Mapa hustoty po barevných modifikacích, Mapa hustoty zobrazená bez barevných modifikací jako černobílý obrázek.

Další komunikace probíhá při požadavku na korekci počtu osob při překryvech. Na server přichází informace potřebné k tomu, aby věděl, jak jsou rozmístěné obrázky v klientské aplikaci. Poté server provede výpočet korekce, o kterém budu psát níže, a odesílá redukovaný počet zpět klientovi.

Poslední komunikace je při akcích souvisejících s projektem samotným, tedy přejmenování, uložení, zobrazení seznamu uložených projektů, načtení a smazání projektu. Při těchto akcích je na straně serveru vždy manipulováno se souborovým systémem. Nejkomplikovanější akce je uložení a poté i načtení. Od klienta přichází konfigurační informace o jeho stavu ve formě *dictionary*. Ty se ukládají do konfiguračního JSON souboru. Při načtení je pak proces opačný. Struktura konfiguračního dokumentu je znázorněna na obrázku 4.4.

K uchování dat na straně serveru se používá souborový systém. Ve složce *ROOT/static/projects* se nachází složky jednotlivých projektů. V každé z nich jsou pak složky:

- */density_maps* - pro vygenerované numpy mapy hustoty
- */density_map_images* - mapy hustoty pro zobrazení u klienta
- */images* - fotografie

```
{
  "projectName": "ostrava",
  "center": {
    "lat": 49.80382228767067,
    "lng": 18.255946314432546
  },
  "images": [
    {
      "name": "1.jpg",
      "imageURL": "projects/ostrava/images/1.jpg",
      "densityMapURL": "projects/ostrava/density_map_images/1_den_map.png",
      "anchors": [
        {
          "lat": 49.80420659384306,
          "lng": 18.25462102890015
        },
```


Obrázek 4.4: Ukázka konfiguračního JSON dokumentu pro uložení stavu klienta.

4.2 Klientská aplikace

Uživatel vybírá obrázky, umisťuje je do mapy, posouvá je, odstraňuje, ukládá a načítá projekt atd. - vše to, co uživatel dělá, musí klient implementovat. V této kapitole popíši hlavní funkce potřebné pro funkčnost programu.

Logika webového klienta je implementována v JavaScriptu. Klientskou aplikaci jsem rozdělil na dvě části - index.js, což je hlavní skript spravující reakce na vstupy uživatele a ImageOfCrowd.js, který obsahuje třídu reprezentující objekt jednoho obrázku. Slovník těchto objektů pro jeden projekt je uchováván v paměti a tvoří to, co uživatel vidí a co modifikuje.

Objekt obrázku na klientské straně

Při vložení obrázků na klientské straně vznikají objekty jednotlivých obrázků. Většina prvků v objektu je podstatná především z vizualizačních důvodů.

Objekt má následující proměnné:

name	Jméno obrázku, který byl vložen.
imageURL	URL obrázku na serveru.
densityMapURL	URL mapy hustoty uložené na serveru.
base64Image	Načtený vstupní obrázek zakódovaný v base64.
anchorMarkers	Pole objektů 4 markerů krajních bodů obrázku v mapě.
anchors	Pole 4 gps souřadnic krajních bodů obrázku v mapě.
externalPolygon	Objekt externího polygonu ohraničující obrázek v mapě.
count	Odhad počtu lidí v obrázku vrácený mapou hustoty.
isPlaced	Flag určující, zda byl obrázek již umístěn do mapy.

Při načtení obrázku zůstávají všechny položky objektu inicializované na null nebo, v případě polí, na prázdné pole. Inicializuje se jen *name* a *base64Image*, aby byl vidět načtený obrázek. Poté při vložení obrázku do mapy se vyplní i proměnné určující hraniční body - *anchorMarkers*, *anchors*, *externalPolygon*. Zároveň se odesílá před umístěním do mapy požadavek na zpracování obrázku serverem. Proto se tedy vyplní i proměnné *densityMapURL*, *count*, *isPlaced*.

Objekt obrázku na straně klienta dále obsahuje funkce pro dynamické zvýraznění a umožnění modifikace aktuálně vybraného snímku. Mapové prvky lze do mapy vložit a při jejich vyjmutí je možné je zachovat pro pozdější vložení. Tyto funkce jsou:

- *deleteMarkers* - odstranění markerů z mapy
- *deletePolygon* - odstranění polygonu z mapy
- *drawMarkers* - vykreslení markerů zpět do mapy
- *drawPolygon* - vykreslení polygonu zpět do mapy
- *drawPolygon* - vykreslení polygonu zpět do mapy
- *updateAnchors, updateMarkers* - aktualizace markerů při manipulaci s nimi

Poté je zde funkce *makeAnchors*, která se používá při prvním vložení obrázku do mapy. Vytvoří inicializační umístění čtyř okrajových bodů obrázku v závislosti na centru mapy.

Poslední funkcí objektu obrázku je *getArea*. Tato funkce vrací velikost plochy, na které je obrázek v mapě zobrazený. Plocha se vypočítává díky Leaflet funkci *distanceTo*, která vrací vzdálenost dvou bodů, a Heronovu vzorci pro výpočet obsahu trojúhelníka. Správnost vypočítané plochy byla prakticky ověřena přeměřením známých ploch v mapě.

Komunikace se serverem

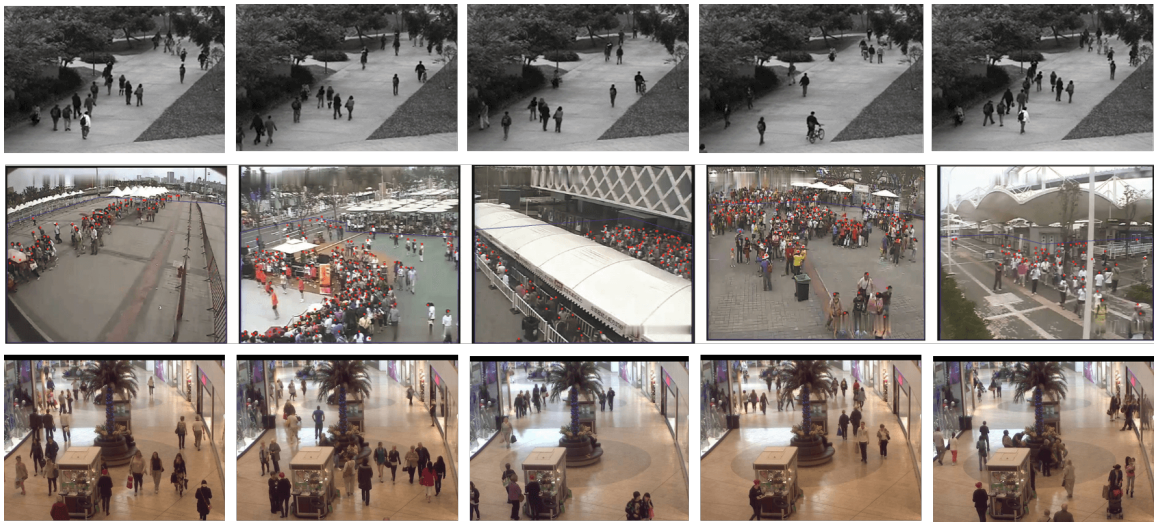
Odesílání požadavků je řešené pomocí jQuery requestů (POST, GET). Celá aplikace je implementována asynchronně. To výrazně zpříjemní uživatelův zážitek, jelikož načítání uvidí pouze jednou. Data jsou serializovaná do podoby JSON, při odesílání fotografie se používá kódování base64.

Funkce, které reagují na akce požadující komunikaci se z pravidla skládají ze tří částí: připravení dat a jejich zabalení do zprávy, odeslání dat a nakonec vykonání akce případe přijmutí dat. Z klienta odchází následující požadavky:

- /sendmap - požadavek na spočítání mapy hustoty
- /compute-overlapping - vypočítání korekce překryvů
- /rename_project - přejmenování projektu
- /open_project - načtení uloženého projektu
- /save_project - uložení projektu
- /delete_project - odstranení projektu
- /list_projects - vypsání seznnamu projeků možných k načtení
- /delete_img - vymazání obrázku z projektu

4.3 Datasety pro počítání davu

Kvalitní datasety tvoří základ trénování neuronových sítí, právě to, co obsahují společně s architekturou neuronové sítě, rozhoduje o kvalitě výsledného modelu. Lze je mezi sebou porovnávat podle mnoha kritérií, a to jak kvantitativních, tak kvalitativních. Datasety mohou být složeny ze samostatně pořízených fotografií, nebo extrahováním snímků z video sekvencí. Dataset vytvořený ze snímků video sekvencí se pyšní velkým počtem obrázků, zato samostatné fotografie variabilitou scény.



Obrázek 4.5: Ukázkové snímky z prvních významných datasetů. Od shora **USDC**, **Mall** a **WordExpo'10**. Převzato z [5, 7, 6, 37].

Kromě trénování se pak datasety používají při vyhodnocování a porovnávaní jednotlivých architektur neuronových sítí.

Kvantitativními parametry datasetů pak jsou:

- Počet snímků
- Počet anotovaných osob

Kvalitativními jsou následující:

- Počet různých scén
- Rozlišení fotografií (velikost a i to, zda je rozlišení u všech stejné či ne)
- Průměrný počet lidí na fotografii (pak i rozložení počtů lidí v celém datasetu)
- Vzdálenost, z jaké je fotografie pořizována
- Úhel, pod kterým je vyfocena
- Typ anotace osob

Standardní datasety

Datasety začaly vznikat pouze během poslední několika let. Dřívější datasety obsahují spíše řídké davy, ty novější už obsahují davy husté, s větší variabilitou, kvalitou obrazu a úhlem pohledu. To umožňuje větší generalizaci při tvorbě modelů. Nyní popíši nejznámější dosud vytvořené datasety.

První významné datasety, které obsahují anotované videosnímky z CCTV kamer, jsou **USDC** [5], **Mall** [7, 6] a **WordExpo'10 dataset** [37] zobrazené na obrázku 4.5 . **USDC Dataset** je první dataset vytvořený pro počítání lidí. Obsahuje 2000 snímků o velikosti 238x158 z video sekvence cesty, kde prochází chodci. Poskytuje také informaci o tzv. "area of interest"k redukci vlivu okolních pohybujících se objektů. Na obrazech je dav s poměrně

nízkou hustotou okolo 15 osob. Nejsou zde žádné variace scény. **Mall dataset** obsahuje pouze jednu scénu. Anotace jsou zde typu dot. Obsahuje 2000 video snímků, přes 60 000 anotovaných osob s průměrem 31 osob na snímek. **WordExpo'10** pak obsahuje 108 scén, 3980 anotovaných snímků s průměrem 50 osob na snímek. Byl vyvinut v rámci Cross-scene Crowd Counting via Deep Convolutional Neural Networks [37].

Další datasety zobrazené na obrázku 4.6 cílí na davy s větší hustotou a variabilitou.

Obrázek 4.6: Ukázka nových datasetů zaměřujících se na hustější davy. Zleva **UCF_CC_50** [15], **UCF-QNRF** [16], **ShanghaiTech** [38].

UCF_CC_50 dataset jako první přichází s variabilními daty. Obsahuje 50 obrázků velmi hustých a rozsáhlých davů z koncertů, protestů a maratonů. Jeden obrázek průměrně obsahuje 1280 osob. Nevýhodou je zde malý počet snímků.

UCF_QRNF dataset je dataset, který se snaží doplnit hlavní nedostatky **UCF_CC_50 datasetu**, a to především v počtu snímků a variabilitou scény. Přichází již v barevném provedení. Obsahuje 1535 obrázků s 1 251 642 anotacemi a průměrným počtem lidí na fotku 815.

Shanghai Tech dataset je novým rozsáhlým datasetem, který představil Zhang. Obsahuje celkem 1198 snímků. Je rozdělený na dvě části: **Part A** a **Part B**. Část **A** obsahuje 482 různých obrázků z internetu, které mají vyšší hustotu. Část **B** obsahuje snímky z ulice Shanghaie, které mají značně nižší hustotu. Všechny obrázky jsou o velikosti 576 x 720 pixelů.

Na přelomu roku 2019 a 2020 byly publikovány dva velmi dobře zpracované a rozsáhlé datasety, benchmarky, a to **JHU-CROWD++** [30, 31] a **NWPU Crowd** [34]. Tyto datasety nastavují novou úroveň jak v počtu snímků tak ve variabilitě.

JHU-CROWD++ je dataset obsahující 4 372 snímků a 1,51 miliónů anotací. Dataset obsahuje celou škálu hustot davu, možných zhoršených světelných podmínek a také fotografie, které neobsahují dav, ale rušivé scény (tzv. distractor). Ukázka datasetu je na obrázku 4.7. Dataset je tvořen anotacemi hlav, ale také poskytuje bounding boxy a navíc samostatně bounding boxy pro rozmazané obličeje.

NWPU Crowd dataset je v aktuální chvíli (2020) nejrozsáhlejší dataset obsahující 5 109 obrázků s 2 133 238 anotacemi. Velký důraz je zde kladen na přítomnost negativních a někdy i matoucích vzorků, jejichž ukázka je zobrazena v prvním řádku obrázku 4.8. Tím, že tento dataset má zároveň funkci benchmarku, tyto vzorky sehrají při testování velkou roli.

Obrázek 4.7: Ukázkové snímky z datasetu **JHU** s popisky toho, čím jsou charakteristické. Převzato z [30, 31].

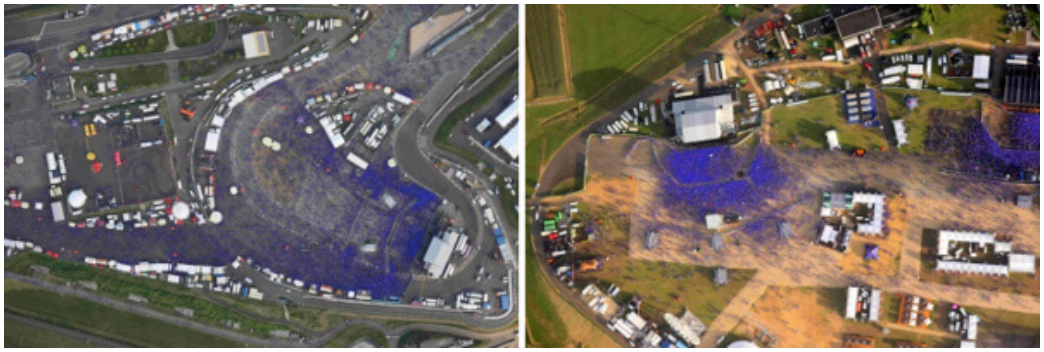
Datasety pořízené dronem

Vzhledem k vzestupu technologie v oblasti dronů jsou tato zařízení čím dál tím víc dostupná. Jejich přínos v oblasti počítání davu spočívá v tom, že dron je schopný rychle a efektivně zaznamenat celý dav z horního pohledu . Fotografie z dronů jsou ale odlišné od těch, které se vyskytují v doposud zmíněných datasetech. Proto pro lepší výsledky v této oblasti je vhodné použít datasety vytvořené dronem. V současné době jsou zde dostupné dva takové: **DroneCrowd** a **DLR-ACD**.

DLR-ACD [3] je dataset pořízený ze vzduchu, který obsahuje 33 velkých snímků ze 16 různých letů. Jsou zde zachycené sportovní události, městská centra i open-air festivaly. Snímky byly pořízené DSLR kamerou umístěnou na helikoptéře. Jejich měřítko oproti realitě se pohybuje od 4.5 do 15 cm/pixel. V rozlišení se pak jedná například o fotografii velkou 34565184 pixelů. Ukázka datasetu je na obrázku 4.9. Dataset byl anotován manuálně pomocí bodů. Je zde celkem 226 291 anotací s rozsahem 285 až 24 368 osob na jeden obrázek. Dataset byl publikován v souvislosti s architekturou MRCNet [3].

DroneCrowd dataset, který byl vytvořen v projektu VisDrone [39] [40], obsahuje 288 video záznamů s 261 908 snímky a 10 209 statických fotografií. Pořízen byl pomocí různých dronů v situacích s variabilním počasím, světelnými podmínkami a různými scénami. Obsahuje scény z města i venkova oblastí v Číně. Některé davy jsou husté a jiné řídké. Je zde přes 2,6 milionu anotovaných objektů. Mezi objekty jsou ale nejen lidé, ale i vozidla a kola. Anotace jsou provedené pomocí bounding boxů. Aplikaci datasetu pro crowd counting provedl Wen et. al [36]. Ukázka z datasetu je zobrazená na obrázku 4.10.

Obrázek 4.8: Ukázkové snímky z datasetu NWPU Crowd [34], kde na prvním řádku jsou zobrazeny různé negativní vzorky a na druhém již fotky s davy.



Obrázek 4.9: Ukázka z datasetu DLR-ACD [3] pro určování hustoty davu a počtu lidí z leteckých fotografií.

Dataset se zaměřuje na 5 různých výzev:

1. detekce a kategorizace objektů v obrázku - VisDrone2019-DET dataset
2. detekce objektů ve videu - VisDrone2019-VID dataset
3. tracking jednoho objektu ve videu - VisDrone2019-SOT dataset
4. tracking více objektů ve videu - VisDrone2019-MOT dataset
5. počítání davu - VisDrone2020-CC dataset

Obrázek 4.10: Ukázka datasetu DroneCrowd (VisDrone2020-CC) v projektu VisDrone [39, 40].

Vlastní dataset

Téma je řešeno v rámci fakultního výzkumného projektu, který spolupracuje i s bezpečnostními složkami státu. Shromáždil jsem záběry relevantních situací a vybral z nich a anotoval celkem 32 snímků. Dataset jsem rozdělil na dvě části DRONE_1 a DRONE_2.

Záběry v datasetu DRONE_1 jsou pořízeny z výšky pomocí dronu nebo kamery na helikoptéře. Scénáře obsahují hloučky až davy lidí. na části záběrů jsou lidé přicházející na městský stadion, záběry jsou z velké dálky bez přibližování. Prostředí navíc komplikují stromy. Dále je zobrazena celá cesta sportovních fanoušků od nádraží po již zmíněný stadion. V těchto záběrech navíc figurují i zásahové jednotky Policie ČR, které na akci dohlíží. Záběry jsou z dálky, ale občas je použit optický zoom. Poslední záběr zobrazuje větší skupinu lidí na prostranství. Záběr je z menší výšky a často i z úhlu. Dataset DRONE_2 pak obsahuje konstantní scénu fotbalového hřiště s pohybujícím se hloučkem sportovců. Dron natáčí pod mírným úhlem z výšky cca 60 metrů.

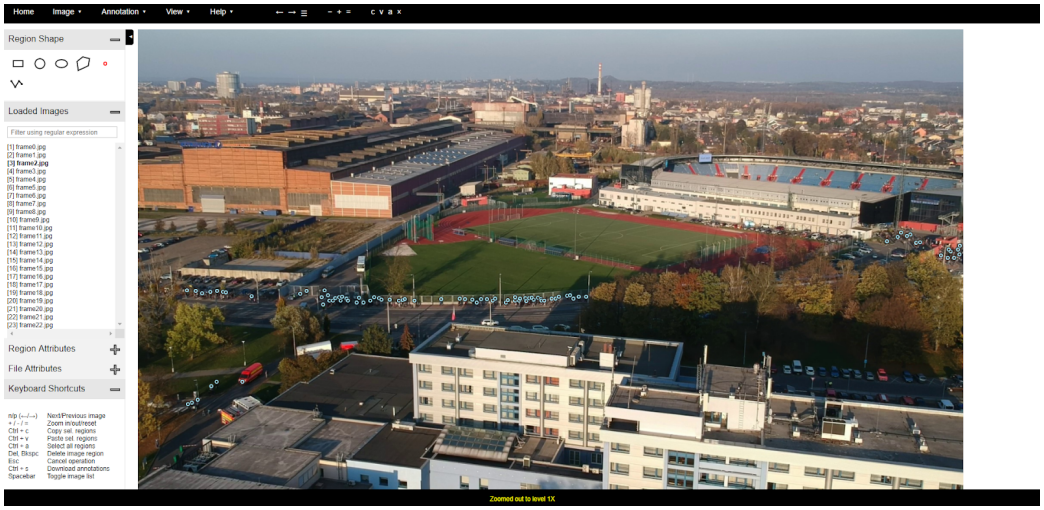
Pro anotování snímků jsem použil OpenSource anotační nástroj VGG Image Annotator (VIA)[4] viz obrázek 4.11. Tento nástroj ze založený čistě na HTML, CSS a JS. Je možné si ho stáhnout a používat i offline. Nabízí velké množství anotací od boundary boxu po polygon. Ta, kterou jsem využil já, byly prosté body. Anotace je možné si stáhnou ve formátu CSV nebo JSON. Stažené anotace mají specifickou strukturu, která odpovídá možnostem, které VIA nabízí. V mém případě šlo ale jen o prosté anotace hlav, a tak jsem JSON soubor s anotacemi dále zpracoval a extrahoval z něj pouze body umístění hlav. Ve vzniklém datasetu pak jsou dvě složky. Jedna s obrázky a druhá s anotacemi ve formátu (x, y \n[5]).

4.4 Testování modelu MCNN na datových sadách

V této kapitole popíši výsledky testování použitého modelu CNN na standartních datasetech, vyhodnotím různé konkrétní situace, kdy model selhával, a poté analyzuji model na vlastním datasetu.

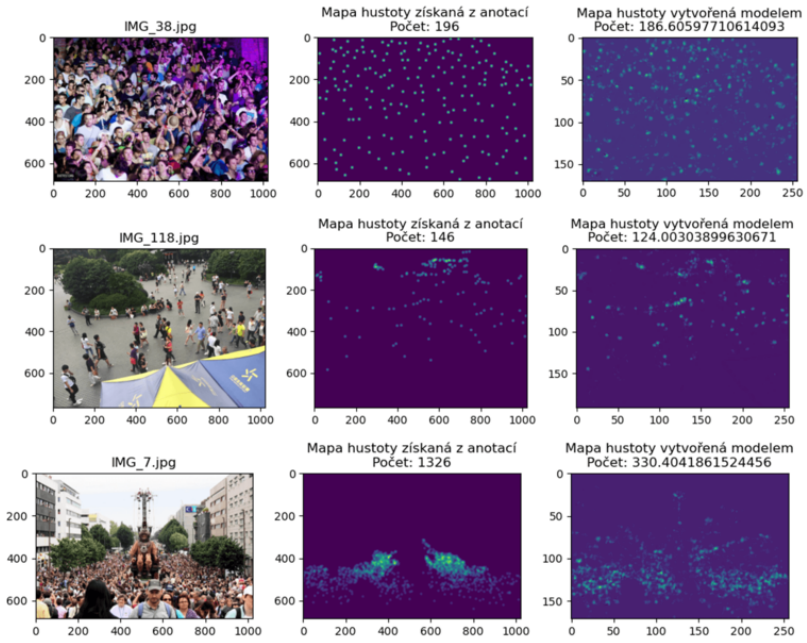
[4] http://www.robots.ox.ac.uk/~vgg/software/via/via.html

[5] new line - nový řádek



Obrázek 4.11: Anotační nástroj VGG Image Annotator.

Do své aplikace jsem použil již natrénovaný model MCNN. Konkrétně používám natrénovaný model z neoficiální implementace MCNN priyanka-kasture[6] v knihovně Keras. Tento model je dobře použitelný s malým počtem závislostí na externích knihovnách.



Obrázek 4.12: Ukázky z výsledků datsetu ShanghaiTech.

Na vyhodnocení úspěšnosti natrénovaného modelu se používají standardní datasety. V praxi to pak znamená, že architektura je na daném datasetu nejdříve natrénována a poté na testovací části testována.

[6] https://github.com/priyanka-kasture/Crowd-Counting-with-MCNNs

V případě nasazení modelu do provozu ale nastává situace jiná. Pro nové situace se již model nedotrénovává a zůstává stejný. Vyhodnocovat budu tedy pouze model MCNN natrénovaný na datasetu ShanghaiTech B.

Pro testování jsem vytvořil skript, který s drobnými modifikacemi lze využít na více různých datasetů. Datasety se vyhodnotí na základě metrik MEA a MSE. Navíc skript vytvoří přehled výsledků (obrázky 4.12, 4.14, 4.15, 4.16) i pro jednotlivé snímky, aby se mohly hodnotit podmínky úspěchu či neúspěchu.

	AVG	MAE	MSE	APE
DRONE_2	28.21	1.99	2.35	6.91%
ShanghaiTechB	123.80	29.40	62.64	30.72%
DRONE_1	133.17	74.84	110.38	67.92%
VisDrone	146.12	73.74	93.01	56.36%
ShanghaiTechA	433.31	199.23	331.46	38.63%

Vyhodnocení jsem provedl pro standardní datasety Shanghaitech A, Shanghaitech B, VisDrone. Poté jsem vyhodnotil výsledky na malých mnou vytvořených datasetech, Konkrétně DRONE_1 a DRONE_2. Výsledky shrnuje tabulka 4.1. MAE a MSE jsem objasnil již v teoretické části. Zde jsem navíc ještě přidal hodnotu AVG, která ukazuje, jaký je průměrný počet lidí v datasetu a procentuální hodnotu chyby, díky těmto dvěma údajům je snadnější si představit, jak je chyba veliká ve vztahu k velikosti datasetu. Procentuální chybu jsem spočítal podle vzorců 4.1, 4.2.

$$PE = \frac{|ES - GT|}{GT} * 100 \tag{4.1}$$

$$APE = \frac{\sum_1^N PE}{N} \tag{4.2}$$

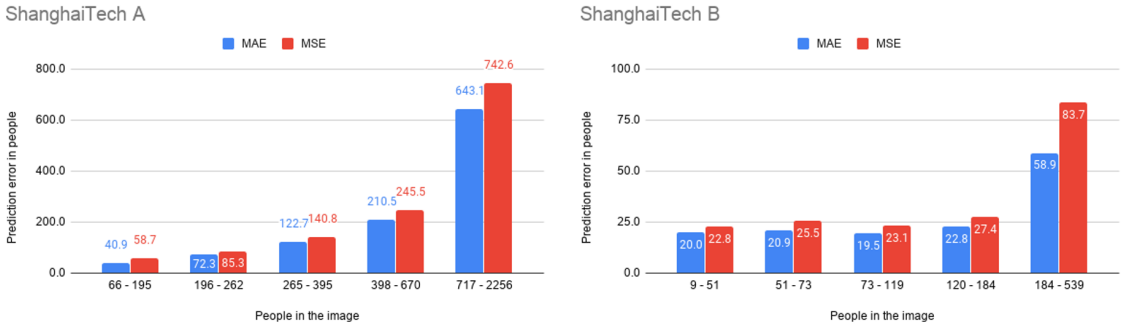
kde PE je procentuální chyba, APE je průměrná procentuální chyba, GT je počet lidí podle anotace a ES je odhadovaný počet lidí.

Vyhodnocení výsledků na vybraných datasetech

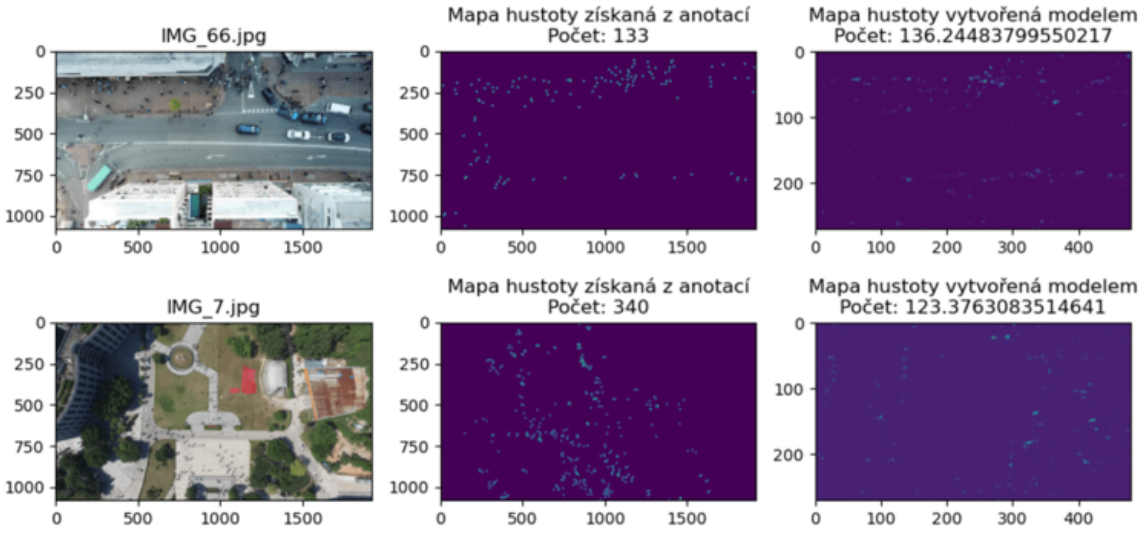
Na datasetu ShanghaiTech podává model dobré výsledky. Generované mapy jsou přesné jak svou hodnotou, tak rozmístěním. Ukázku výsledků lze vidět na obrázku 4.12. Značná část chyb je způsobena vyšší hustotou davu ve vzdálenějších oblastech scény. Pokud se ve scéně vyskytnou stromy, je možné, že je to bude registrovat také a zkreslovat výsledek. V grafu 4.13 jsem pro vyhodnocení rozdělil testované snímky vždy do pěti stejně velkých skupin podle počtu. Lze zde pozorovat, že na části B podává model stabilní výsledky. U části A se výsledky rapidně zhorší ve skupině s počtem 717 - 2256 lidí.

Velikost chyby je roste při následujících scénářích:

1. při zvýšení hustoty davu (tedy překrývání)
2. při zhoršení rozlišení obrazu
3. úhlu pohledu (pohled kolmo z vrchu je složitější úloha než např. pohled pod úhlem 30 stupňů)



Obrázek 4.13: Výsledky testování na datasetu ShanghaiTech A a B rozdělené do pěti stejně velkých skupin podle počtu lidí.

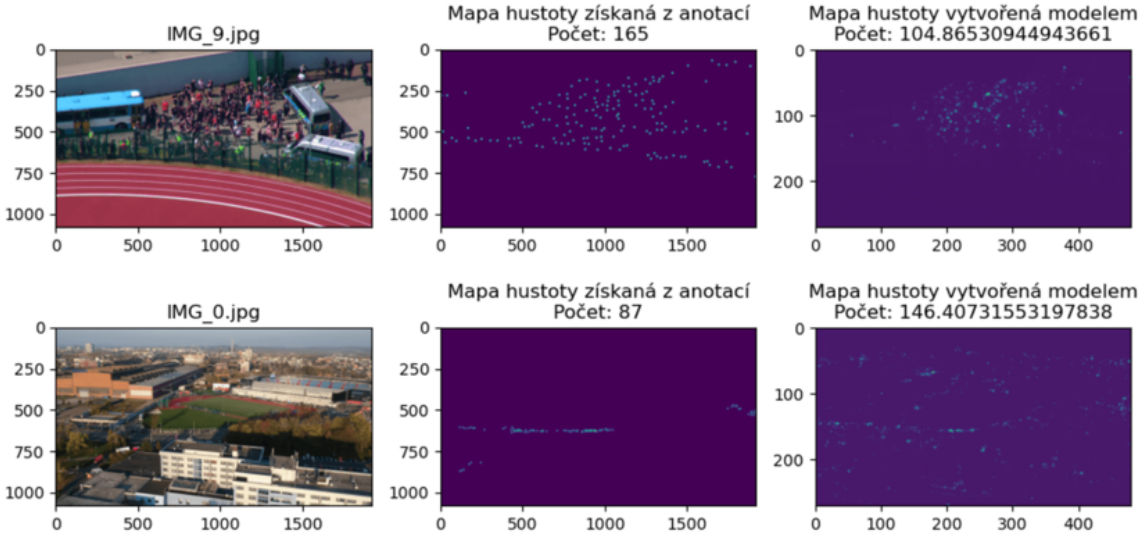


Obrázek 4.14: Dva ukázkové snímky z datasetu VisDrone2020-CC.

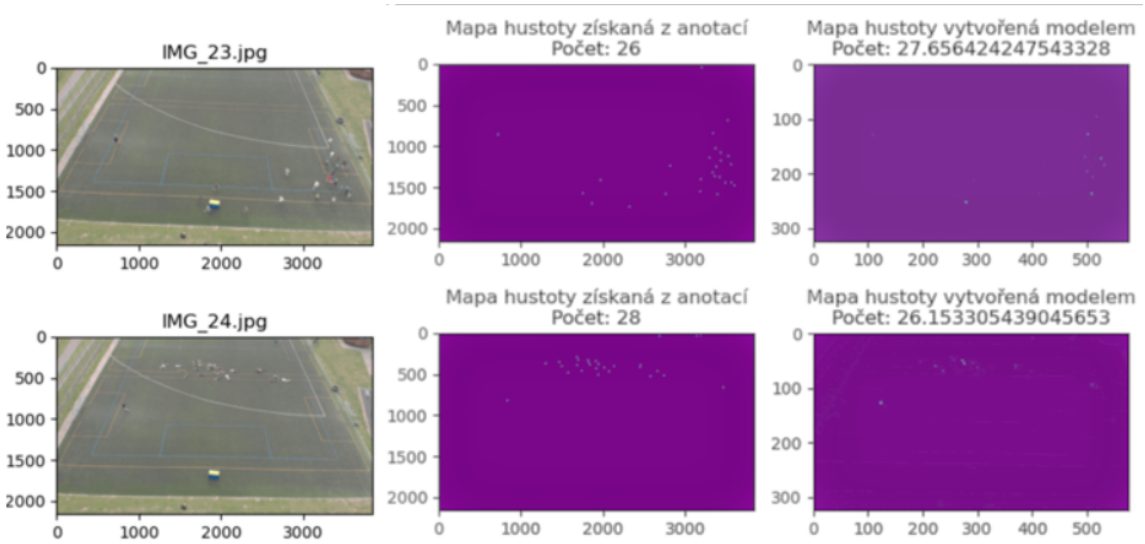
Dále jsem vyhodnocoval model na datasetu VisDrone2020-CC. Tento dataset jsem vybral především proto, že se scény blíží využití aplikace - snímky z dronu. Bohužel ale fotografie jsou foceny až moc z dálky, a proto model, natrénovaný na datech focených maximálně z výšky budovy, nepodal uspokojivý výkon. Při průměrné hodnotě 146.16 lidí na fotografii je chyba 73.74 moc velká a navíc podle vzhledu generovaných map je vidět, že model není na tak velkou vzdálenost vhodný (viz. obrázek 4.14). Snímek IMG_66.jpg se podařil správně zpracovat, na snímku IMG_t.jpg už model selhal z důvodu velké vzdálenosti.

Dataset DRONE_1 není rozhodně svým počtem reprezentativní. Je u něj podstatné dívat se na jednotlivé snímky samostatně. Reprezentují totiž snímky z případu užití aplikace. Na obrázku 4.15 jsou dva ukázkové snímky z tohoto datasetu.

Nakonec jsem testoval na snímcích ze záběru sportovní rozcvičky na fotbalovém hřišti pořízené dronem (Dataset DRONE_2). Z tři a půl minuty dlouhého videa jsem extrahoval 13 ukázkových snímků. Průměr počtu osob na jednom snímku zde byl 28.21. Snímky byly v rozlišení 4K. Při zpracovaní snímků modelem predikce vždy vycházela nad 70. To bylo neuspokojující, jelikož vzhled výstupní mapy hustoty odpovídal. Problém byl ve vysokém



Obrázek 4.15: Dva ukázkové snímky z datasetu DRONE_1.



Obrázek 4.16: Dva ukázkové snímky z datasetu DRONE_2.

rozlišení. Po tom, co jsem snímku snížil rozlišení o 60 procent, výsledky začaly odpovídat velmi přesně s MEA 1,99.

4.5 Uživatelské testování

V rámci řešení jsem provedl pilotní test. Cílem testu bylo získat zpětnou vazbu na první verzi GUI a na základě ní ho aktualizovat.

Pro uživatele je zásadní, aby získání celkového počtu lidí bylo co nejsnazší a intuitivní. V této kapitole popíši, jaké akce jsou od uživatele očekávané, a kroky, které jsem provedl pro zjednodušení uživatelské činnosti.

Prvním krokem je výběr a nahrání fotografií davu, se kterými se dále pracuje. Poté má uživatel za úkol umístit do mapy jednotlivé obrázky podle 4 bodů, které korespondují se čtyřmi rohovými body obrázku. Obrázky posouvá jako celek nebo je vhodně posunem krajních bodů perspektivně transformuje. Obrázkem je myšlena fotografie nebo k ní korespondující mapa hustoty davu. Mezi těmito pohledy uživatel může přepínat.

Uživatel vidí, kolik je lidí na jednotlivých obrázcích a kolik je tedy celkově lidí na celé akci (suma jednotlivých obrázků s korekcí překryvů).

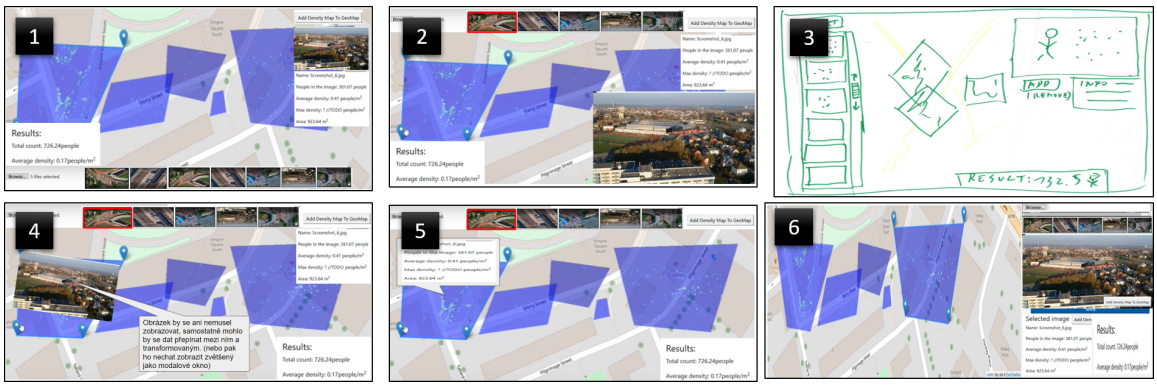
Tím, že se umístila mapa hustoty do topografické mapy, je možné vypočítat plochu, na které se dav vyskytuje. To může být zajímavý údaj, ze kterého lze vyvodit hustotu vztaženou k metrům čtverečným.

Pilotní test pro úpravu GUI

První verze GUI, zobrazená na obrázku 4.18, reflektovala základní funkcionalitu aplikace. Vznikl tak MVP, se kterým byl proveden pilotní test (muž, 43 let, znalý práce s počítačem). Výstupem experimentu jsou následující požadavky na revizi GUI:

- Je potřeba, aby uživatel poznal, v jakém pořadí a proč má aplikaci ovládat. GUI musí reflektovat uživatelský proces.
- Mapa je klíčový interaktivní prvek (manipulace s fotkami), musí jí být dáno výrazně více prostoru.
- Vhodné zobrazovat vybraný obrázek a jeho mapu hustoty ve větším formátu, než jen náhledový.

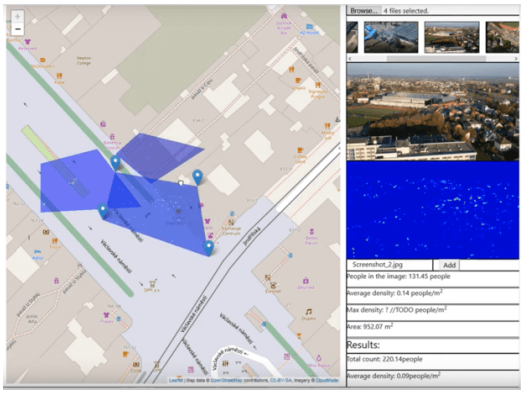
Dosavadní GUI jsem rozebral na jednotlivé komponenty a poté skládal zpět do jednoho celku. Inspiraci jsem čerpal z fungujících aplikací, které vizualizují pomocí mapy. Existovaly dvě základní cesty, mezi kterými jsem se rozhodoval: první možností bylo ovládací prvky mapy nechat mapu překrývat, druhou možností bylo umístit je vhodně vedle mapy. Na základě této zpětné vazby jsem pomocí wireframů a foto koláží v grafických editorech vytvořil 6 různých nových layoutů ve formě makety, ty jsou znázorněné na obrázku 4.17.



Obrázek 4.17: Návrhy na úpravu GUI na základě zpětné vazby formou koláží a wireframu.

Návrhy jsem s účastníkem pilotního testu znovu konzultoval a výsledné finální rozvržení implementoval (viz obr. 4.19). Za nejvhodnější návrh byl vybrán layout číslo 6. GUI je rozděleno na dva sloupce, což jasně vymezuje prostor vizualizační a prostor dalšího ovládání.

Obrázek 4.18: Prototyp aplikace před pilotním testem.



Obrázek 4.19: Výsledné GUI po pilotním testu.

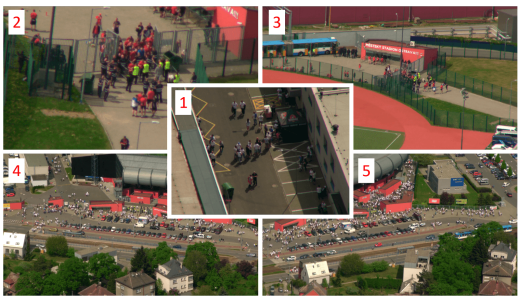
V levém, který zabírá 60 procent celého prostředí, je zobrazena mapa. V pravém sloupci se nachází ovládací prvky. Od shora je zde umístěn výběr pro načtení fotografií, procházení náhledů načtených fotografií, velký náhled aktuální vybrané fotografie, velký náhled její mapy hustoty a na závěr informace o fotografii a celém projektu. Prvky v ovládání jsou seřazeny od shora dolů tak, jak se s nimi bude popořadě pracovat.

Test kvality odhadu počtu osob v reálné situaci

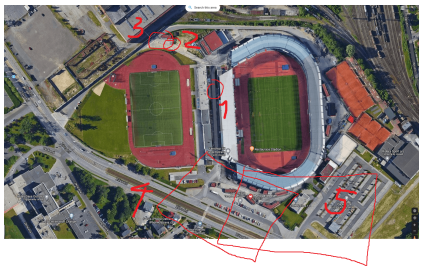
Pro otestování použití jsem využil jeden ze záznamů v rámci fakultního výzkumného projektu. Na záznamu je více různých davů lidí přicházejících na akci k ostravské aréně. Záznam je pořízen z helikoptéry pomocí kamery, která měla možnost zoomu. V záběru obletí helikoptéra celou arénu a zabere všechny hloučky lidí včetně toho největšího před arénou.

Vybral jsem pět nejvhodnějších snímků z videa, při vybírání jsem si dělal poznámky do screenshotu mapy, kam je mám umístit. Vybrané snímky jsou znázorněné na obrázku 4.20 a mapka s poznámkami je na obrázku 4.21.

Výsledek po umístění obrázků do aplikace je znázorněn na screenshotu 4.22. Je vidět, že snímky 1 - 3 jsou o poznání menší, jelikož byly pořízené při přiblížení, oproti nim snímky 4 a 5 pokrývají velký prostor z větší dálky. Tím, že jsou snímky 1-3 zobrazeny malé, není z dálky na mapě hustoty dostatečně vidět zachycená skupinka, na snímcích 4 a 5, je dav vidět zřetelně. Přesto by se dalo uvažovat o vhodnější barevné úpravě.



Obrázek 4.20: Vybrané snímky pro ukázku reálné situace.



Obrázek 4.21: Mapa s poznámkami kam mají být snímky umístěné.

fotografie	predikce	anotace	chyba	procenta
1	41.61	50	8.39	16.79%
2	35.40	74	38.60	52.16%
3	32.66	84	51.34	61.12%
4	112.58	365	252.42	69.16%
5	140.97	449	308.03	68.60%
celkem bez korekce překryvu (2x se dvě fotografie se překrývají)				
celkem	363.22	1022	658.78	64.46%
po korekci překryvů				
celkem	270.7	761.7	491.0	64.46%

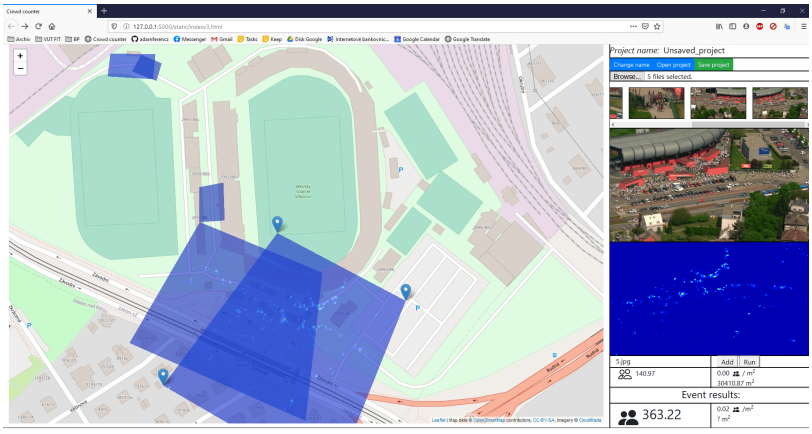
Tabulka 4.1: Tabulka vyhodnocující odhady počtu lidí k fotografiím použitých v aplikaci.

Co se týká počtů, jednotlivé fotografie jsem pro vyhodnocení anotoval, abych mohl objektivně vyhodnotit správnost výsledku. U obrázku 1 je odhad správný, u snímků 2 a 3 byla chyba způsobená tím, že některé osoby jsou skryté za plotem, nebo jsou to těžkooděnci v černém. U fotografií sice mapa hustoty kopíruje správně rozmístění davu, odhadovaný počet ale neodpovídá. Je to způsobeno tím, že je dav natáčen až moc z dálky. Bylo by vhodnější záběr přiblížit a oblast pokrýt více snímky.

V tabulce 4.1 je vidět, že celkový odhad bez korekce překryvů je 363 lidí oproti součtu anotací 1022. Po korekci počtu z důvodu překryvů se celkový odhadovaný počet snížil z 363 na 271, což odpovídá očekávání podle toho, jak jsou obrázky překryté. Abych určil i hodnotu pro anotované snímky po korekci překryvů, použil jsem přímou úměru.

Ještě je třeba zdůraznit, že při ručním anotovaní dat také dochází k chybám. Navíc jsem byl velmi přísný a snažil jsem se anotovat i velmi rozmazané husté části davu i částečně skryté osoby (za plotem, za sklem). Pokud bych anotoval pouze to, u čeho bych věřil, že to model dokáže rozpoznat, chyba by byla mnohem menší.

Zvýšení přesnosti by se dalo řešit dvěma způsoby - vhodnější záběry, nebo lépe natrénovaný model na data více z dálky.



Obrázek 4.22: Zobrazení ukázkové akce po rozmístění snímků do mapy.

Kapitola 5

Závěr

Cílem práce bylo prozkoumat a vhodně využít metody počítání davu. Vytvořil jsem aplikaci, která umožní spočítat lidi na hromadné akci s využitím topologické mapy, fotografií a neuronové sítě.

Hlavní myšlenkou mého řešení je propojení interaktivní mapy s možnostmi metod pro analýzu davu, konkrétně počítání davu pomocí neuronových sítí. Využití mapového podkladu dopřeje uživateli nadhled a možnost lépe si vizualizovat danou akci (demonstraci atd.). V rámci této práce jsem nastudoval řadu metod pro odhad počtu lidí z fotografie. Současně s tím jsem prozkoumal a shromáždil tradiční i nejnovější datasety. Navíc jsem anotoval i vlastní dataset obsahující reálné situace pořízené dronem. Své řešení jsem navrhl jako klient-server aplikaci. Server umožňuje nasazení natrénovaného modelu neuronové sítě s architekturou MCNN a komunikuje s webovým klientem pomocí HTTP protokolu. Klient zajišťuje vstup dat, interakci s mapou a vizualizaci výsledku. Vyvinul jsem efektivní GUI, které jsem testoval na uživateli a dále optimalizoval. Navrhl a implementoval jsem metodu, která bere v potaz, že když se jednotlivé fotografie překrývají, měl by být výsledný počet lidí vhodně redukován. To umožní poskládat celkovou vizualizaci akce z více překrývajících se fotografií a dosáhnout tak přesnějšího výsledku. Na vlastním datasetu i na některých standardních jsem vyhodnotil MCNN. Soustředil jsem se na to, jaké parametry jsou rozhodující při počítání davu modelem neuronové sítě.

Výsledky mé práce budou dále využity např. ve fakultním výzkumném projektu, který spolupracuje i s bezpečnostními složkami státu. Dalšími kroky může být práce s video záběry v reálném čase. Aplikaci bude možné modifikovat a upravit pro připojení dalších modulů např. na detekci anomálie v davu. Dlouhodobým cílem je pak aplikace, která by pomáhala policii v dohledu nad davem při veřejné akci. Měla by poskytnout mnoho dalších informací, jako je nebezpečné chování, rychlost davu, detekce nebezpečných osob a jejich následná lokalizace v mapě, informace o pohybu jednotek, umístění dronů a toho, kam se dívají. Bezpečnostní složky by pak měly možnost analyzovat situaci v reálném čase a lépe předcházet nepříjemným situacím.

Literatura

[1] Albawi, S., Abed Mohammed, T. a ALZAWI, S. Understanding of a Convolutional Neural Network. In:. Srpen 2017.

[2] Babu Sam, D., Surya, S. a Venkatesh Babu, R. Switching Convolutional Neural Network for Crowd Counting. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[3] Bahmanyar, R., Vig, E. a Reinartz, P. MRCNet: Crowd Counting and Density Map Estimation in Aerial and Ground Imagery. *ArXiv preprint arXiv:1909.12743*. 2019.

[4] Boominathan, L., Kruthiventi, S. S. a Babu, R. V. Crowdnet: A deep convolutional network for dense crowd counting. In: *Proceedings of the 24th ACM international conference on Multimedia*. 2016, s. 640–644.

[5] Chan, A. B., Liang, Z.-S. J. a Vasconcelos, N. Privacy preserving crowd monitoring: Counting people without people models or tracking. In: IEEE. *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008, s. 1–7.

[6] Chen, K., Gong, S., Xiang, T. a Change Loy, C. Cumulative attribute space for age and crowd density estimation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, s. 2467–2474.

[7] Chen, K., Loy, C. C., Gong, S. a Xiang, T. Feature mining for localised crowd counting. In: *BMVC*. 2012, s. 3.

[8] Fu, M., Xu, P., Li, X., Liu, Q., Ye, M. et al. Fast crowd density estimation with convolutional neural networks. *Engineering Applications of Artificial Intelligence*. Elsevier. 2015, roč. 43, s. 81–88.

[9] Gao, J., Lin, W., Zhao, B., Wang, D., Gao, C. et al. C^ 3 Framework: An Open-source PyTorch Code for Crowd Counting. *ArXiv preprint arXiv:1907.02724*. 2019.

[10] Gao, J., Wang, Q. a Yuan, Y. SCAR: Spatial-/channel-wise attention regression networks for crowd counting. *Neurocomputing*. Elsevier. 2019, roč. 363, s. 1–8.

[11] Haralick, R. M. Statistical and structural approaches to texture. *Proceedings of the IEEE*. May 1979, roč. 67, č. 5, s. 786–804. ISSN 1558-2256.

[12] Harris, C. a Stephens, M. A combined corner and edge detector. In: *In Proc. of Fourth Alvey Vision Conference*. 1988, s. 147–151.

[13] Holčík, J., Komenda, M. a kol (end.) a. *Matematická biologie: e-learningová učebnice*. 1. vydání. Brno: Masarykova univerzita, 2015. Dostupné z: https://portal.matematickabiologie.cz/. ISBN 978-80-210-8095-9.

[14] Hussain, N., Yatim, H. S. M., Hussain, N. L., Yan, J. L. S. a Haron, F. CDES: A pixel-based crowd density estimation system for Masjid al-Haram. *Safety Science*. 2011, roč. 49, č. 6, s. 824 – 833. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0925753511000075. ISSN 0925-7535.

[15] Idrees, H., Saleemi, I., Seibert, C. a Shah, M. Multi-source Multi-scale Counting in Extremely Dense Crowd Images. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2013.

[16] Idrees, H., Tayyab, M., Athrey, K., Zhang, D., Al Maadeed, S. et al. Composition Loss for Counting, Density Map Estimation and Localization in Dense Crowds. In: *The European Conference on Computer Vision (ECCV)*. September 2018.

[17] Jain, A. K., Jianchang Mao a Mohiuddin, K. M. Artificial neural networks: a tutorial. *Computer*. 1996, roč. 29, č. 3, s. 31–44.

[18] Jain, A. K., Mao, J. a Mohiuddin, K. M. Artificial Neural Networks: A Tutorial. *Computer*. Los Alamitos, CA, USA: IEEE Computer Society Press. březen 1996, roč. 29, č. 3, s. 31–44. Dostupné z: https://doi.org/10.1109/2.485891. ISSN 0018-9162.

[19] Kohonen, T. The self-organizing map. *Proceedings of the IEEE*. Sep. 1990, roč. 78, č. 9, s. 1464–1480. ISSN 1558-2256.

[20] Li, Y., Zhang, X. a Chen, D. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, s. 1091–1100.

[21] Liang, R., Zhu, Y. a Wang, H. Counting crowd flow based on feature points. *Neurocomputing*. 2014, roč. 133, s. 377 – 384. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0925231214000630. ISSN 0925-2312.

[22] Liu, W., Salzmann, M. a Fua, P. Context-aware crowd counting. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, s. 5099–5108.

[23] Marana, A. N., Velastin, S. A., Costa, L. F. a Lotufo, R. A. Estimation of crowd density using image processing. In: *IEE Colloquium on Image Processing for Security Applications (Digest No: 1997/074)*. March 1997, s. 11/1–11/8. ISSN null.

[24] Munakata, T. *Fundamentals of the new artificial intelligence: neural, evolutionary, fuzzy and more*. Springer Science & Business Media, 2008.

[25] Oñoro, D. a López Sastre, R. Towards Perspective-Free Object Counting with Deep Learning. In:. říjen 2016.

[26] Saleh, S. A. M., Suandi, S. A. a Ibrahim, H. Recent survey on crowd density estimation and counting for visual surveillance. *Engineering Applications of Artificial Intelligence*. 2015, roč. 41, s. 103 – 114. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0952197615000081. ISSN 0952-1976.

[27] Sazli, M. A brief review of feed-forward neural networks. *Communications, Faculty Of Science, University of Ankara*. Leden 2006, roč. 50, s. 11–17.

[28] Shang, C., Ai, H. a Bai, B. End-to-end crowd counting via joint learning local and global count. In: IEEE. *2016 IEEE International Conference on Image Processing (ICIP)*. 2016, s. 1215–1219.

[29] Sindagi, V. A. a Patel, V. M. A survey of recent advances in CNN-based single image crowd counting and density estimation. *Pattern Recognition Letters*. 2018, roč. 107, s. 3 – 16. Video Surveillance-oriented Biometrics. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0167865517302398. ISSN 0167-8655.

[30] Sindagi, V. A., Yasarla, R. a Patel, V. M. Pushing the frontiers of unconstrained crowd counting: New dataset and benchmark method. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, s. 1221–1231.

[31] Sindagi, V. A., Yasarla, R. a Patel, V. M. JHU-CROWD++: Large-Scale Crowd Counting Dataset and A Benchmark Method. *Technical Report*. 2020.

[32] Wang, C., Zhang, H., Yang, L., Liu, S. a Cao, X. Deep people counting in extremely dense crowds. In: *Proceedings of the 23rd ACM international conference on Multimedia*. 2015, s. 1299–1302.

[33] Wang, J., Lo, S., Wang, Q., Sun, J. a Mu, H. Risk of Large-Scale Evacuation Based on the Effectiveness of Rescue Strategies Under Different Crowd Densities. *Risk Analysis*. Wiley Online Library. Srpen 2013, roč. 33, č. 8, s. 1553–1563. Dostupné z: https://doi.org/10.1111/j.1539-6924.2012.01923.x. ISSN 1539-6924.

[34] Wang, Q., Gao, J., Lin, W. a Li, X. NWPU-Crowd: A Large-Scale Benchmark for Crowd Counting. *ArXiv preprint arXiv:2001.03360*. 2020.

[35] Wang, Q., Gao, J., Lin, W. a Yuan, Y. Learning from synthetic data for crowd counting in the wild. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019, s. 8198–8207.

[36] Wen, L., Du, D., Zhu, P., Hu, Q., Wang, Q. et al. Drone-based Joint Density Map Estimation, Localization and Tracking with Space-Time Multi-Scale Attention Network. *ArXiv preprint arXiv:1912.01811*. 2019.

[37] Zhang, C., Li, H., Wang, X. a Yang, X. Cross-scene crowd counting via deep convolutional neural networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, s. 833–841.

[38] Zhang, Y., Zhou, D., Chen, S., Gao, S. a Ma, Y. Single-Image Crowd Counting via Multi-Column Convolutional Neural Network. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[39] Zhu, P., Wen, L., Bian, X., Ling, H. a Hu, Q. Vision meets drones: A challenge. *ArXiv preprint arXiv:1804.07437*. 2018.

[40] Zhu, P., Wen, L., Du, D., Bian, X., Hu, Q. et al. Vision Meets Drones: Past, Present and Future. *ArXiv preprint arXiv:2001.06303*. 2020.