FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Algoritmy pro detekci trajektorie míče
Student:	Bc. Jakub Pečenka
Vedoucí:	doc. Ing. Ivan Šimeček, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Teoretická informatika
Katedra:	Katedra teoretické informatiky
Platnost zadání:	do konce zimního semestru 2022/2023

Pokyny pro vypracování

1) Nastudujte problém detekce kruhových objektů ve scéně. Zaměřte se na algoritmy pro detekování trajektorie tenisového míče na obrazovém záznamu. [1][2]
2) Určete vhodné výstupy řešení (například: trajektorie, rychlost, místo dopadu apod.).
3) Vyberte nebo navrhněte vhodný přístup k řešení problému a řešení naimplementujte. Zaměřte se na výpočetní náročnost řešení a zvažte možnosti paralelizace na CPU a GPU.
4) Vyhodnoťte implementované řešení z hlediska přesnosti na navržených výstupech, výpočetního času a proveďte vizualizaci výstupů řešení.

[1] P. R. Kamble, A. Keskar, and K. Bhurchandi. Ball tracking in sports: a survey. Artificial Intelligence Review, pages 1–51, 2017.
[2] Y.-C. Huang, I.-N. Liao, C.-H. Chen, T.-U. Ik, and W.-C. Peng. Tracknet: A deep learning network for tracking high-speed and tiny objects in sports applications*. In2019 16th IEEEInternational Conference on Advanced Video and Signal Based Surveillance (AVSS), pag

---

Elektronicky schválil/a doc. Ing. Jan Janoušek, Ph.D. dne 13. srpna 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Algoritmy pro detekci trajektorie míče

Bc. Jakub Pečenka

Katedra teoretické informatiky
Vedoucí práce: doc. Ing. Ivan Šimeček, Ph.D.

5. ledna 2022

Poděkování

Děkuji svému vedoucímu panu doc. Ing. Ivanu Šimečkovi, Ph.D. za rady a čas, který mi věnoval. Dále panu Ing. Marku Sušickému a Mgr. Adamu Szábovi za návrh tématu práce a rady a čas, který mi věnovali. Děkuji svým rodičům za podporu v době vysokoškolského studia.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 5. ledna 2022 . . . . . .

Odkaz na tuto práci

Pečenka, Jakub. *Algoritmy pro detekci trajektorie míče.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

V této práci je řešena problematika detekce trajektorie tenisového míčku, kurtu, hráčů a událostí (úder, dopad apod.) v televizním záznamu tenisového utkání. Práce předpokládá vyznačení začátku a konce tenisové výměny uživatelem a omezuje se na dvouhru. Řešení je vyhodnoceno z hlediska výpočetního času a správnosti detekovaných událostí. Řešení pracuje na testované hardwarové konfiguraci rychlostí 0,68 snímku za sekundu. Pro kategorie událostí úder a servis rozlišovaných dle hráče a dopad rozlišovaný dle poloviny hřiště je dosaženo u metriky F_1 hodnoty přes 85 % při vzdálenosti mezi predikcí a skutečnou událostí do 5 snímků.

Klíčová slova trajektorie, tenisový míček, tenis, anotace tenisových událostí, detekce kurtu, detekce hráčů, paralelizace

Abstract

This thesis focuses on detection of tennis ball trajectory, tennis court, players and events (hit, land etc.) in broadcast TV video of tennis match. Work assumes start and end of tennis game marked manually and is limited on singles matches. The solution is evaluated in terms of computational time and correctness of the detected events. The solution works on the tested hardware configuration at a rate of 0.68 frames per second. For the event categories of hit and serve differentiated by player and impact differentiated by half of the court, a value of over 85 % for measure F_1 is achieved with maximum distance 5 frames between prediction and true event.

Keywords trajectory, tennis ball, tennis, annotation of tennis events, court detection, player detection, parallelization

Obsah

Úvod **1**

1 Cíl práce **3**

2 Úvodní teorie **5**
2.1 Tenis jako hra 5
2.2 Charakteristika vstupních dat 6
2.3 Digitální obraz 8
2.3.1 Video 8
2.3.2 Základní operace a pojmy související se zpracováním obrazu 8
2.3.3 Počítačové vidění 9
2.4 Základy strojového učení 10
2.5 Neuronové sítě 13
2.5.1 Dopředné neuronové sítě 13
2.5.2 Konvoluční neuronové sítě 14
2.6 Paralelizace - základní pojmy 16

3 Analýza možných přístupů k řešení problému **17**
3.1 Popis systému pro detekci objektu zájmu ve sportu 17
3.2 Detekce objektu zájmu pomocí tradičních technik 18
3.3 Detekce objektu zájmu pomocí hlubokého učení 21
3.4 Odhadování trajektorie 24
3.4.1 Kalmanův filtr 25
3.4.2 Particle filtr 25
3.4.3 Přístup *Trajectory* 25
3.4.4 Přístup *Data association* 26
3.4.5 Přístup optimalizující *účelovou funkci* 27
3.4.6 Algoritmus *TLDA* 27
3.5 Vytěžování informací z trajektorie 30

4 Návrh řešení **31**
4.1 Detekce míčku na jednotlivých snímcích 32
4.2 Odhadování trajektorie 34
4.3 Detekce hráčů 35
4.4 Detekce hřiště 37
4.5 Detekce událostí 40

5 Implementace **53**
5.1 Základní struktura implementace 53
5.2 Popis zdrojových souborů 54
5.3 Implementace vybraných paralelizací 57
5.3.1 Paralelizace detekce kurtu 57
5.3.2 Paralelizace *Floyd-Warshallova* algoritmu 57

6 Testování **59**
6.1 Datové sady 59
6.1.1 Problém s benchmarky 60
6.2 Testování správnosti výstupů 61
6.2.1 Metodika testování 61
6.2.2 Výsledky měření správností výstupů 63
6.2.3 Vyhodnocení 64
6.3 Měření výpočetního času 65
6.3.1 Hardwarová konfigurace 65
6.3.2 Metodika měření 65
6.3.3 Výsledky měření výpočetního času 65
6.3.4 Vyhodnocení 66

Závěr **73**

Literatura **75**

A Seznam použitých zkratek **81**

B Seznam klíčových pojmů **83**

C Obsah přiloženého CD **85**

Seznam obrázků

2.1 Tenisový kurt s rozměry. Zdroj [1]. 6
2.2 Typický způsob snímání *tenisové výměny* v televizním záznamu. Zdroj [2]. 7
2.3 Jeden neuron, diagram. [3] 13
2.4 Neuronová síť, diagram. [3] 14

3.1 Přehled systému na detekci objektu. [4] 17
3.2 Diagram algoritmu *TLDA*. 29

4.1 Návrh systému na detekci trajektorie tenisového míčku. 31
4.2 Diagram pro řešení detekce míčku na snímku *tradičními technikami*. 33
4.3 Diagram pro řešení detekce míčku na snímku pomocí *CNN*. 34
4.4 Detekce míčku z *CNN TrackNet* jako červená kolečka a bílá čára detekovaná trajektorie míčku pomocí algoritmu *TLDA*. Původní obrázek pochází z [2]. 35
4.5 Diagram detekce hráčů. 36
4.6 Výstup *CNN YOLOv3* pro detekci osob. Původní obrázek pochází z [2]. 36
4.7 Diagram detekce tenisového kurtu. 39
4.8 Výsledek detekce tenisového kurtu. Červeně body získané průsečíky, modře body získané *projektivní transformací*, žlutě body získané jedním z předešlých dvou způsobů. Původní obrázek pochází z [2]. 39
4.9 Diagram detekce *událostí*. 40
4.10 Trajektorie s detekovanými *událostmi* světle modře. Původní obrázek pochází z [2]. 42
4.11 Konečný automat pro filtraci *událostí* tenisové výměny v dopředném směru. Přechodové hrany nastávají pro *události* odpovídající cílovému stavu přechodu. 47

4.12 Konečný automat pro filtraci *událostí* tenisové výměny ve zpětném směru. Přechodové hrany nastávají pro *události* odpovídající cílovému stavu přechodu. 48

6.1 Vzdálenosti mezi predikcí a anotací v počtu snímků pro *true positive* predikce. Bez omezení vzdálenost predikce a anotace v počtu snímků. Vyjádřeno pomocí *krabicového grafu (box plot)*, kde *vousy* reprezentují 1,5 násobek *kvartilového rozpětí*. 69
6.2 Vzdálenosti mezi predikcí a anotací v počtu snímků pro *true positive* predikce, které mají maximální povolenou vzdálenost 5 snímků. Vyjádřeno pomocí *krabicového grafu (box plot), kde vousy reprezentují 1,5 násobek kvartilového rozpětí.* 70
6.3 Skutečná a predikovaná rychlost podání. Červeně jsou označeny odlehlé hodnoty. 71

Seznam tabulek

3.1 Tabulka s výsledky z [5] pro *CNN Tracknet*. Tradiční technika [6] je popsaná v sekci 3.2, Tracknet 1-1 značí verzi s jedním vstupním snímkem, Tracknet 3-1 značí 3 vstupní snímky a Tracknet-2 značí trénování na rozšířené datové sadě. 22

3.2 Tabulka s výsledky z [7] pro *CNN Tracknet* a *TracknetV2* pro detekci badmintonového míčku. Značení 3-1 značí verzi s třemi vstupními a jedním výstupním snímkem jako predikcí, značení 3-3 značí 3 vstupní snímky a 3 výstupní snímky, kdy každý je brán jako nezávislá predikce. 23

6.1 Tabulka s počty anotovaných událostí. 60

6.2 Tabulka s výsledky testování korektnosti výstupů. Bez omezení vzdálenost predikce a anotace v počtu snímků. 67

6.3 Tabulka s výsledky testování korektnosti výstupů. Vzdálenost predikce a anotace omezena na 5 snímků. 68

6.4 Tabulka s výsledky testování korektnosti výstupů. Událostí v závorce jsou agregovány do jedné události. 68

6.5 Tabulka s výsledky testování korektnosti predikce rychlosti podání. 69

6.6 Tabulka s výsledky měření výpočetního času jednotlivých částí systému. Vyjádřeno v absolutním čase v sekundách a procentuálním vyjádření. Čas i procenta jsou zaokrouhlena na dvě desetinná místa. Zkratka *par.* znamená paralelní verze a zkratka *FW Floyd-Warshallův algoritmus*. 71

6.7 Tabulka s výsledky měření výpočetního času paralelní CPU verze a sekvenční verze detekce kurtu. Čas je zaokrouhlen na dvě desetinná místa. 72

6.8 Tabulka s výsledky měření výpočetního času paralelní GPU verze a sekvenční verze v jazyku *Python* a *C Floyd Warshallova* algoritmu. Čas je zaokrouhlen na tři desetinná místa. 72

6.9 Tabulka s výsledky měření výpočetního času části systému *hledání trajektorie* s různými verzemi algoritmu *FW*. Pro kombinaci C a PyCUDA je PyCUDA uplatněna pro matice od velikosti 800x800. 72

Úvod

Tenis je celosvětově oblíbená míčová hra. Záznamy tenisových utkání jsou běžně vysílány v televizi a jejich analýza může mít rozličné využití. Tato práce se zabývá detekcí trajektorie tenisového míčku a jejím následným využitím k popisu událostí v tenisové výměně. V práci se omezuji na dvouhru a standardní televizní záznam tenisového utkání a předpokládám manuální vyznačení začátku a konce tenisové výměny.

Druhá kapitola se skládá z vymezení pojmů a popisu teorie spojených se zpracovávanou problematikou.

V třetí kapitole jsou představeny přístupy používané pro detekci trajektorie tenisového míčku a jejího využití pro získání informací o událostech v tenisové výměně.

Ve čtvrté kapitole je navržen přístup pro detekci trajektorie tenisového míčku. Dále pro detekci hráčů a tenisového kurtu, což je využito jako informace umožňující lépe určit zmíněné události. Je navržen přístup pro detekci událostí v tenisové výměně. Diskutována je také možná paralelizace řešení pro zrychlení výpočetního času.

V páté kapitole je popsána implementace navrženého systému a vybraných paralelizací.

V šesté kapitole je systém otestován z hlediska správnosti poskytovaných výstupů a je změřen výpočetní čas. Výsledky jsou následně vyhodnoceny.

Práce byla zpracována ve spolupráci s laboratoří *opendatalab* zabývající se zpracováním otevřených dat, která vznikla ve spolupráci s *Fakultou informačních technologií ČVUT v Praze* a společnosti *Profinit EU*.

Kapitola 1

Cíl práce

Cílem práce je nastudovat problém detekování trajektorie tenisového míče v obrazovém záznamu a určit vhodné výstupy. Dále navrhnout vhodný přístup k detekci trajektorie tenisového míčku a následném získávání vhodných výstupů z této trajektorie. Navržené řešení implementovat s ohledem na výpočetní čas a zvážit jeho paralelizaci. Na závěr vyhodnotit správnost poskytovaných výstupů a výpočetní čas systému a provést vizualizaci řešení.

Kapitola 2

Úvodní teorie

V této kapitole budou uvedeny pojmy a teorie související s touto prací. Jsou popsány i pravidla tenisu pro lepší orientaci v následných kapitolách, kde je tato znalost předpokládána.

2.1 Tenis jako hra

Tenis je míčová hra, která vznikla ve 13. století ve Francii. Hrají ji dva, případně čtyři hráči, dle toho se typ hry nazývá *dvouhra* nebo *čtyřhra*. Každý z hráčů má raketu předepsaných parametrů, se kterou odráží tenisový míček žluté barvy, který se snaží dostat na polovinu hřiště soupeře.

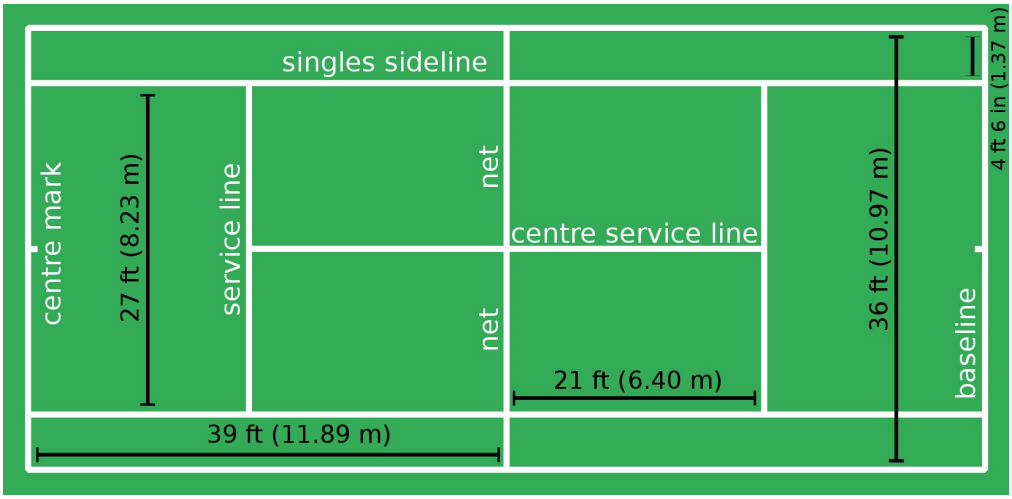
Hra se hraje na obdélníkovém hřišti, které je rozděleno na dvě poloviny sítí. Na každé polovině se nacházejí u sítě dva obdélníky pro umístění servisu. Dále dva obdélníky po stranách hřiště označují rozšíření hřiště pro hraní *čtyřhry*. Grafické znázornění včetně rozměrů je na obrázku 2.1. V textu budou čáry rovnoběžně se sítí nazývány horizontální a čáry kolmé na síť vertikální.

Hráči musí míček trefit do správné části hřiště soupeře. Cílem hry je donutit soupeře neodrazit míček zpět na hřiště druhé strany do určeného místa nebo ho nevrátit před dvěma dopady na jeho straně hřiště.

Jako *tenisová výměna* se označuje úsek hry započatý *servisem*, což je úder zahajující výměnu, a skončen *vítězným míčkem* nebo chybou. Za každou *tenisovou výměnu* dostane jedna ze stran bod. Po dosažení čtyř bodů dostává daná stran *gem*. Ovšem pokud obě strany dosáhli tří bodů je následně nutné obdržet dva body po sobě pro obdržení *gemu*.

Po získání šesti a více *gemů* obdrží hráč *set*, pokud ovšem soupeř má *gemů* alespoň o dva méně. Další variantou hry je po získání šesti *gemů* oběma stranami započít takzvaný *tie break*, tento se hraje na získání alespoň sedmi bodů s tím, že rozdíl musí být alespoň o dva body. Hraje se dle pravidel turnaje na dva nebo tři získané *sety*.

Obrázek 2.1: Tenisový kurt s rozměry. Zdroj [1].



Tenisovou výměnu začíná jedna strana *servisem*, kdy odpálí míček do předepsaného prostoru soupeřova hřiště, což je jeden z obdélníků u sítě. Po každém míčku se střídá *servírování* z pravé půlky hřiště do levého obdélníku z pohledu *servírujícího hráče* a naopak, kdy první podání v *gemu* je z pravé půlky. *Servis* se střídá mezi soupeřícími stranami po odehraném *gemu*. V *tie-breaku* se *servis* střídá po první *výměně* a následně vždy po dvou *výměnách*.

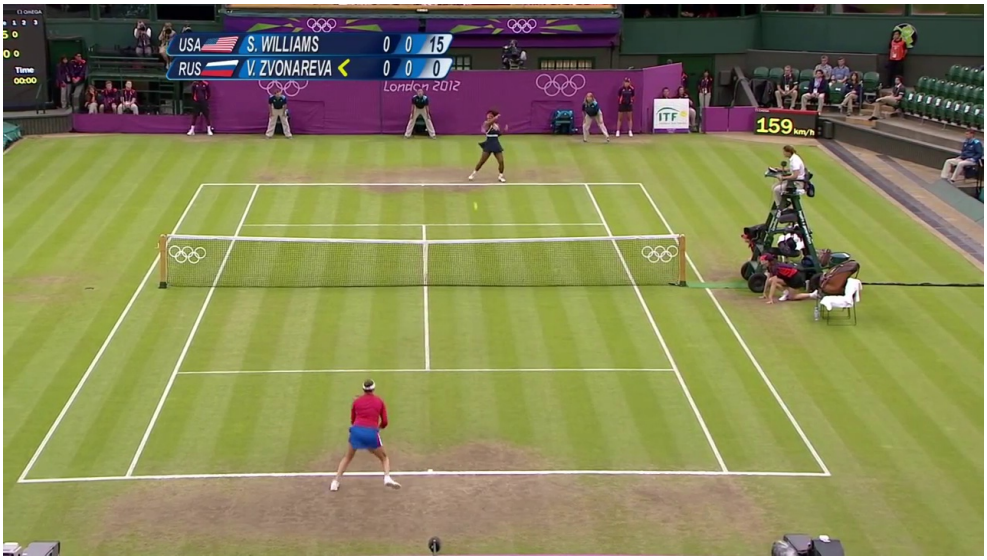
Rozlišuje se osm základních úderů *servis*, *forhend*, *bekhend*, *volej*, *halfvolej*, *smeč*, *zkrácení hry* a *lob*.

2.2 Charakteristika vstupních dat

Jedna ze základních vlastností vstupních obrazových dat je, zda pocházejí ze zdroje pod naši kontrolou. Vzhledem k tomu, že téma práce bylo navrženo laboratoří *opendatalab*, zabývající se využitím otevřených dostupných dat, bude tato práce zpracována s předpokladem absence kontroly nad pořízením obrazových dat.

Dle [8] patří detekce tenisového míčku k lehčím úlohám v kontextu sportu díky barevné odlišnosti míčku, malému prostoru na kterém se míček pohybuje a minimálnímu zakrytí míčku jinými objekty. Lze dodat, že z tohoto plyne většinou i statické umístění kamery během tenisové výměny, což úlohu dále zlehčuje, ovšem může docházet k malým pohybům kamery a malému přibližování a oddalování obrazu. Někdy je též snímán *servis* hráče jinou kamerou než zbytek výměny. Autoři v [9] uvádějí jako výhodu přehledné prostředí na rozdíl od her jako fotbal nebo basketbal.

Obrázek 2.2: Typický způsob snímání *tenisové výměny* v televizním záznamu. Zdroj [2].



Dle [10], [11], [9], [12] lze mezi výzvy spojené s detekováním tenisového míčku v televizním přenosu řadit:

1. malé rozměry míčku,
2. zastínění míčku sítí, publikem, hráči atd.,
3. změna tvaru míčku na snímcích díky změně rychlosti a poloze vůči kameře,
4. deformace míčku při úderu nebo odrazu,
5. měnící se světelné podmínky během letu míčku,
6. rozličný kontrast mezi míčkem a pozadím během letu tenisového míčku,
7. rychlost a z ní plynoucí rozmazání.
8. prudké změny rychlosti a směru,
9. splynutí s pozadím při rozmazání.

Dle [6] k obtížnosti také patří odlišná vzdálenost míčku od kamery, neboť hřiště je často snímáno z jednoho konce hřiště. Tedy míček se na videu zmenšuje a zvětšuje během tenisové výměny v závislosti na vzdálenosti od kamery.

2.3 Digitální obraz

Digitální obraz se skládá z diskrétních bodů zvaných *pixely*. Každý *pixel* nese informaci o podobě obrazu v daném místě na kterém se nachází. Většinou se jedná o barevnou informaci zakódovanou v určitém formátu, například *RGB*, *HSV* aj.

Rozlišení udává počet *pixelů* tvořící digitální obraz. Čím více *pixelů*, tím více detailů je zachyceno. Tento údaj se udává pomocí dvou čísel značících počet *pixelů* v horizontálním a vertikálním směru, celkový počet *pixelů* dostaneme vynásobením těchto dvou čísel.

Obraz je reprezentován v **barevném formátu**, který určuje způsob vyjádření barvy jednotlivých *pixelů*, kdy je výsledná barva vyjádřena kombinací jednotlivých barevných složek. Relevantní vzhledem k této práci jsou tyto:

- *RGB* nebo také *BGR* je barevný formát, kde je každý *pixel* vyjádřen třemi složkami zastupujícím červenou (R), zelenou (G) a modrou (B) barvu.
- *HSV* je barevný formát, kde jsou *pixely* vyjádřeny následujícími složkami *hue* (H) neboli odstín barvy, *saturation* (S) neboli sytost barvy a *value* (V) neboli jas.
- *Grayscale* je barevný formát, kde je barva reprezentována odstín šedi, tedy *pixel* se skládá z jedné složky určující jas pixelu.

2.3.1 Video

Video se skládá ze sekvence po sobě jdoucích snímků.

Počet snímků za sekundu, případně **frames per second (FPS)** je jednou z hlavních vlastností videa, tento údaj vyjadřuje kolikrát za sekundu je snímám reálný svět a tedy i ztrátu informace danou úseky, které nejsou zaznamenány.

Pokud jde o přesné zaznamenání dopadu míčku je třeba vysoké *FPS*, rozlišení a vhodný úhel snímání.

2.3.2 Základní operace a pojmy související se zpracováním obrazu

Segmentace obrazu je operace hledající v obrazu oblasti s určenými vlastnostmi.

Popředí je oblast zájmu na obrazu a **pozadí** je část obrazu mimo oblast zájmu.

Prahování neboli **thresholding** je druh *segmentace obrazu*, kdy je určen **práh** neboli **threshold**, což je hodnota kterou pokud *pixely* překročí jsou

určeny jako *popředí*, jinak jako *pozadí*. *Práh* se stanovuje pro všechny barevné složky pixelu.

Maska je obraz s vyznačeným *popředím* a *pozadím* výchozího snímku.

Diference snímků je operace, kdy jsou pro dva obrazy stejného rozlišení odečteny po složkách hodnoty odpovídajících *pixelů*, výsledná hodnota určuje hodnotu daného *pixelu* ve výsledném obrazu.

Filtrace je operace, která slouží k potlačení nežádoucího jevu v obraze. Většinou je hodnota *pixelu* vyjádřena vzhledem k jeho okolí, například jsou okolní hodnoty průměrovány nebo je vzat medián.

Kontura je křivka ohraničující shluk *pixelů* se sdílenými vlastnostmi.

Morfologické operace jsou druhy *filtrace*, která se snaží o zachování struktury *kontury*. Základními operacemi jsou *eroze*, která volí hodnotu *pixelu* jako minimum z hodnot *pixelů* v okolí a *dilatace*, která volí maximum, tedy první operace odstraňuje *pixely* z hranic kontury a druhá naopak *pixely* na hranici přidává. Operace jsou určeny *strukturními elementy*, které určují okolí *pixelu* uplatněné během zvolené operace.

Převodem barevného spektra se myslí převod do jiného *barevného spektra* pomocí definovaných vztahů mezi danými dvěma *barevnými spektry*. Například převod z barevného formátu *RGB* do *HSV*.

Vlastnosti kontur jsou charakteristiky *kontury*. Tyto vlastnosti slouží k popisu *kontury* dle kterých je často prováděna jejich filtrace.

Houghova transformace je metoda sloužící k hledání specifikovaných obrazců v obraze.

Pravděpodobnostní Houghova transformace je výpočetně a paměťově efektivnější verze *Houghovy transformace*, kdy nejsou brány v potaz všechny pixely snímku, ale jen náhodný výběr.

2.3.3 Počítačové vidění

Počítačové vidění se zabývá schopností pochopení scény v digitálním obrazu nebo videu. Scéna se chápe pomocí *příznaků*, což jsou informace popisující určitou vlastnost *pixelů* nebo jejich shluků.

Počítačové vidění se dělí na *tradiční techniky* a *hluboké učení*. Rozdíl v těchto přístupech spočívá především v tom, jestli tvůrce systému volí sám příznaky na základě kterých je chápána scéna (*tradiční techniky*), nebo jsou tyto příznaky naučeny modelem z trénovací datové sady (*hluboké učení*).

Úlohy počítačového vidění mají spoustu podob, podstatné pro tuto práci jsou následující uvedené v [13]:

Object Localization je úlohou, kdy jsou lokalizovány objekty na snímku většinou pomocí ohraničení obdélníkem, tzv. *bounding box*.

Object Detection obohacuje úlohu *object Localization* o klasifikaci lokalizovaných objektů.

Image Segmentation je úlohou, kdy jsou detekovanému objektu přiřazeny *pixely* které k němu náleží. Je vytvořena *maska* objektu na snímku. Úlohu lze upřesnit jako **Instance Segmentation** při které jsou tvořeny *masky* jen pro zájmové objekty nebo **Semantic Segmentation**, kdy je pro každý pixel přiřazena klasifikační třída, včetně pozadí.

2.4 Základy strojového učení

V této části budou popsány základní pojmy z oblasti *strojového učení*, které se vyskytují dále v textu.

Modelem ve strojovém učení rozumíme systém, který pro *vstupní data* poskytuje žádoucí odpovědi, neboli *výstupní data*. Vztah *vstupních* a *výstupních dat* lze zapsat následovně:

$$Y = f(X) + \epsilon, \tag{2.1}$$

kde Y jsou *výstupní data*, X jsou *vstupní data*, f je funkce, někdy nazývána *true function*, vyjadřující vztah mezi X a Y, ϵ je *náhodná chyba* nezávislá na X a s nulovou střední hodnotou. Chyba ϵ nelze vysvětlit *vstupními daty* X. *Náhodná chyba* ϵ může pocházet z nedostupných *vstupních dat* nebo šumu ve *vstupních datech*. *Modelem* se snažíme co nejlépe aproximovat funkci f, tedy *model* můžeme zapsat následovně:

$$\hat{Y} = \hat{f}(X), \tag{2.2}$$

kde $\hat{f}$ je *model* a $\hat{Y}$ *predikce*. Podle typu úlohy může *model* sloužit pro predikce, poté se jedná o *predikční úlohu*, nebo jako zdroj pochopení vztahu mezi X a Y, poté se jedná o úlohu *statistické inference*. [14]

Učení je proces hledání vhodného *modelu*.

Učení s učitelem neboli **supervised learning** je *učení* při kterém je ke vstupu dostupný žádaný výstup. *Model* tedy ví, jaká by měla být jeho odpověď pro daný vstup. Předpokládáme, že existuje funkce f z rovnice (2.1), kterou se snažíme aproximovat zvoleným *modelem*. [15]

Parametry *modelu* jsou nastavení, které určuje chování *modelu*. Tyto jsou určeny během *učení modelu* na *trénovací datové sadě*.

Hyperparametry *modelu* jsou *parametry*, které nejsou určeny během *učení* na *trénovací datové sadě*, ale musí být nastaveny explicitně před *procesem*

učení. Nastavení *hyperparemetrů* probíhá ve fázi *selekce modelu* a využívá se *validační datové sady*.

Parametrizovaný model je *model*, který přizpůsobuje své nastavení dle dat, která mu jsou poskytnuta k *trénování*. Nastavení *modelu* je dáno jeho parametry, kterých je pevně daný počet a tento počet nezávisí na množství poskytnutých dat. Oproti tomu **neparametrizovaný model** nelze charakterizovat pevně daným počtem *parametrů*. [15]

Třída hypotéz neboli **hypothesis class** je množina zvažovaných *modelů*. Z této třídy je konkrétní *model* vybrán pomocí procesu *učení*. Volbou *třídy hypotéz* omezujeme *učící algoritmus* v možnostech volby a zanášíme do *učení* takzvané *induktivní vychýlení* neboli *inductive bias*. Volba by měla reflektovat naši znalost problému a vhodnosti daných *modelů* k řešení úlohy. Konkrétní *model* z této třídy se nazývá **hypotéza**. [16]

Testovací chyba neboli **Test error** nebo také **generalization error** je chyba modelu na *testovací datové sadě* nezávislé na *trénovací datové sadě* na které byl model trénován. Lze zapsat následovně:

$$Error_T = E[L(Y, \hat{f}(X))|T], \tag{2.3}$$

kde L je ztrátová funkce a T *trénovací datová sada*. [17]

Trénovací chyba neboli **Training error** je průměrná chyba na *trénovací datové sadě*

$$\overline{Error} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i)). \text{ [17]} \tag{2.4}$$

Neredukovatelná chyba neboli **Irreducible error** je chyba, kterou není možné vysvětlit *vstupními daty*, viz rovnici (2.1). Tedy vždy bude přítomna, bez ohledu na zvolený model. [14]

Redukovatelná chyba neboli **reducible error** je chyba, která závisí na kvalitě *modelu* a s dokonalým *modelem* může být nulová. [14]

Přeučení neboli **overfitting** *modelu* nastává pokud v průběhu *procesu učení* jsou extrahovány šumové nebo chybné informace, které nevysvětlují zájmovou doménu, ale jsou důsledkem nedokonalosti *trénovací datové sady*. Pro *model* je žádoucí vlastnost **generalizace**, tedy zobecnění informací obsažených v *trénovací datové sadě* na celou doménu řešeného problému, respektive na nová data. V praxi se *přeučení modelu* ověřuje pomocí nezávislé *trénovací* a *validační*, případně *testovací* datové sady.

Nedoučení neboli **Underfitting** je problém, kdy *model* není schopen dostatečně dobře aproximovat *true function* f.

Dle [17] lze proces výběru *modelu* rozdělit na dva cíle:

- **Selekce modelu** neboli **Model selection**, je proces, kdy je vybírán nejlepší *model* dle zvolené evaluační metriky.
- **Posouzení modelu** neboli **Model assessment** je proces, při kterém je pro vybraný nejlepší *model* měřena chyba na nových datech nezávislých na *trénovací datové sadě*.

Dle [18] dva faktory ukazující úspěšnost *modelu* jsou:

1. velikost *trénovací chyby* a
2. rozdíl mezi velikostí *trénovací* a *validační/testovací chyby*.

První faktor vypovídá o schopnosti *modelu* obsáhnout relevantní informace z *trénovací datové sady*. Pokud je tento faktor příliš velký jedná se o problém *nedoučení*. Druhý faktor vypovídá o *přeučení (overfitting) modelu*. Toto nastává, pokud je *validační/testovací chyba* výrazně větší nežli *trénovací chyba*. [18]

Dělení datové sady

K selekci *modelu* se v praxi často přistupuje pomocí takzvané *validace*, kdy je část dostupných dat využita výhradně k určení úspěšnosti *modelu*, tedy není využita v *procesu učení*. *Model* lze následně zvolit dle úspěšnosti na *validační datové sadě*. Časově náročnější metodou je *k-násobná křížová validace (k-fold cross validation)*, kdy jsou data rozděleny na k stejně velkých částí, následně je *model natrénován* na $k - 1$ částech a poslední část slouží jako *validační datová sada*. Toto se opakuje pro každou z k částí a výsledná *validační chyba modelu* je dána jako průměr chyb z jednotlivých iterací. Dle [16] tato metoda funguje často v praxi velice dobře.

Dle [16] se v praxi se data většinou dělí na tři částí, kdy k *trénovací* a *validační* je přidána *testovací* část. Postup je následovný, *model* je *učen* na *trénovací* části, *validační* část je využita k výběru nejlepšího *modelu*, tedy k *selekci modelu*, a *testovací* k ohodnocení úspěšnosti vybraného nejúspěšnějšího *modelu*, tedy k *posouzení modelu*.

2.5 Neuronové sítě

V této sekci budou popsány typy *neuronových sítí* relevantních vzhledem k této práci.

2.5.1 Dopředné neuronové sítě

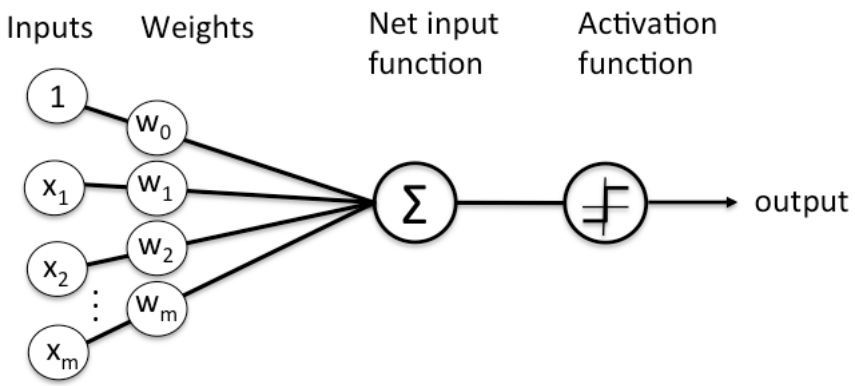
Dopředné neuronové sítě (FFNN) lze popsat jako funkci mající za cíl aproximovat námi hledanou funkci. Samotnou strukturu *neuronové sítě* lze definovat jako vnořené funkce, kdy každá funkce se nazývá *vrstva*. Počet funkcí/*vrstev* určuje *hloubku neuronové sítě*. [18]

Rozlišují se tří hlavní vrstvy *vstupní*, která reprezentuje vstup do sítě, *výstupní*, která je poslední a její výstup je výstupem celé sítě a *skryté*, které jsou všechny mezi *vstupní* a *výstupní*. [18]

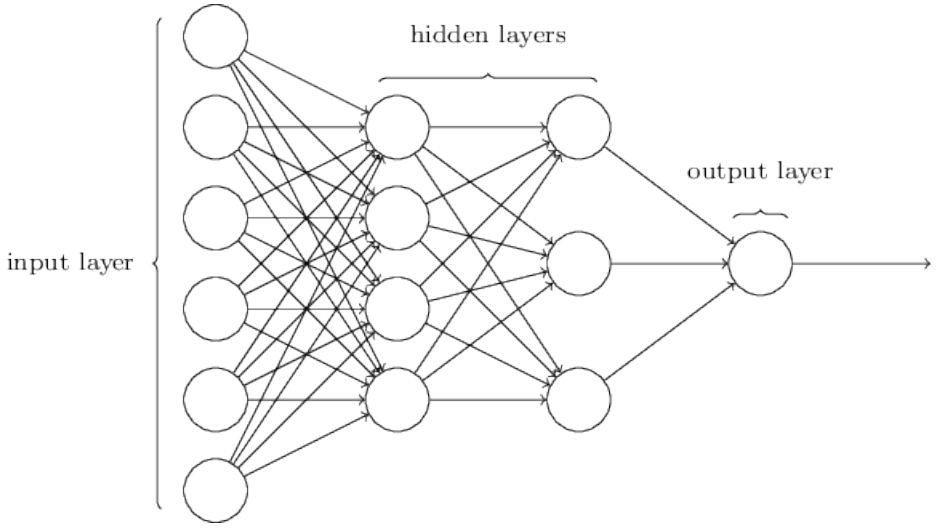
Tento typ *neuronové sítě* se nazývá *dopředný*, neboť informace sítí plyne pouze dopředným směrem, tedy nelze poslat žádnou informaci zpět, *neuronová síť* bere v potaz jen výstup z předchozí *vrstvy*, případně *vstupní data*. [18]

Často jsou *vrstvy FFNN* zobrazovány pomocí nezávislých jednotek zvaných *neurony* mající jako vstup vektor hodnot z předešlé *vrstvy* se kterým se provede pomocí *vah* pro jednotlivé vstupy operace *skalární součin*, výsledek následně slouží jako vstup do *aktivační funkce*, která je často nelineární a její výstup je skalár, který je výstupem *neuronu*. Výstupy ze všech *neuronů* z jedné *vrstvy* slouží jako vstup všem neuronům v další *vrstvě*. Diagram *neuronu* je zobrazen na obrázku 2.3. Celou *FFNN* je možné vizualizovat pomocí *grafu* skládajícího se z jednotlivých *neuronů*, viz 2.4. [18]

Obrázek 2.3: Jeden neuron, diagram. [3]



Obrázek 2.4: Neuronová síť, diagram. [3]



FFNN je popsána *parametry* a *hyperparametry*. Kdy *hyperparamtery* určují architekturu sítě a *parametry* nastavují váhy, které jsou určeny během procesu učení. [18]

2.5.2 Konvoluční neuronové sítě

Jedná se o speciální typ *dopředné neuronové sítě*, který je dle [18] definován takto:

"Konvoluční neuronové sítě jsou jednoduše neuronové sítě, které využívají operaci konvoluce na místo maticového násobení v alespoň jedné své vrstvě." (překlad autora práce z anglického jazyka)

Operace *konvoluce*:

$$s(t) = \int x(a)w(t-a), \tag{2.5}$$

operace *diskrétní konvoluce*:

$$s(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a), \tag{2.6}$$

operace *diskrétní vzájemná korelace/cross-correlation*

$$s(t) = \sum_{a=-\infty}^{\infty} x(a)w(t+a), \tag{2.7}$$

kde $x(t)$ je vstup, $w(t)$ je *konvoluční jádro*, $s(t)$ je výstup po provedení operace a t je index. [18]

Operace (2.6) a (2.7) jsou podobné operace lišící se jen otočením *konvolučního jádra*. V kontextu *neuronových sítí* jsou často zaměňovány, neboť při procesu učení jsou váhy zvoleny dle toho.

Obrazová data a *konvoluční jádro* jsou v počítači reprezentovány jako více dimenzionální pole o konečném rozměru. Vzhledem k diskrétnosti dat se využívá operace (2.6) nebo (2.7), které jsou ovšem definovány jako *suma* přes nekonečný prostor. Je tedy použit předpoklad, že v nekonečném prostoru jsou všude nuly kromě explicitně definovaných míst. Díky tomu lze operaci (2.6) provést jako konečnou sumaci, kdy operace probíhá jen přes definovaná místa. Jak obrazová data, tak *konvoluční jádro* mají většinou formu mřížky, která je vyplněna hodnotami. Okolo této mřížky předpokládáme nuly, jak bylo uvedeno výše.
Tří hlavní vlastnosti *CNN* odlišující je od *FFNN* dle [18] jsou:

- **Řídké propojení** mezi *vrstvami*, kdy v *FFNN* je každý *neuron* propojen s každým *neuronem* v předcházející *vrstvě*. U *CNN* je každý *neuron* propojen jen s tolika *neurony* z předcházející *vrstvy*, kolik je velikost *konvolučního jádra*. Díky tomu je snížena časová a paměťová náročnost *modelu*.
- **Sdílení parametrů**, kdy *neurony* v jedné *vrstvě* sdílí stejné *parametry*/*váhy*. Toto snižuje paměťové nároky *modelu*.
- **Ekvivalenci k translaci**, kdy pokud je na vstup uplatněna operace *translace*, výstup bude změněn stejným způsobem.

Kromě samotné operace *konvoluce* se v *CNN* uplatňují další dvě operace a to:

- **Nelineární funkce**, kdy je na vstup uplatněna funkce nelineárního charakteru.
- **Pooling**, což je operace, která hodnotu na určitém místě nahradí sumarizací jí a okolních hodnot. Například lze vybrat maximum, průměr apod. Toto napomáhá invarianci vůči malým posuvům. Také lze využít ke zmenšení dimenzí vstupu.

Typická architektura *CNN* obsahuje určitý počet *konvolučních vrstev* s nelineární funkcí následovaných vrstvou s operací *pooling*. Dle úlohy jsou přítomny na konci *neuronové sítě* vhodné vrstvy.

2.6 Paralelizace - základní pojmy

Paralelizace slouží k urychlení výpočetního času pomocí souběžného běhu stejné úlohy s různými daty *(datový paralelismus)* nebo souběžného běhu různých úloh se stejnými daty *(funkční paralelismus)* na více výpočetních jednotkách.

Základní metriky paralelního výpočtu jsou:

- *Zrychlení* dané výrazem
$$S_p = \frac{T_S}{T_p},$$
kde T_S je výpočetní čas sekvenční verze programu a T_p je výpočetní čas programu při běhu na p výpočetních jednotkách, vyjadřuje kolikrát je rychlejší paralelní verze algoritmu oproti sekvenční verzi.

- *Efektivita* daná výrazem
$$E = \frac{S_p}{p}$$
vyjadřuje podíl jedné výpočetní jednotky na dosaženém zrychlení.

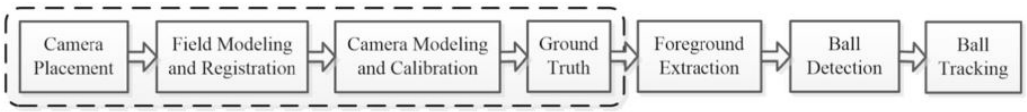
Kapitola 3

Analýza možných přístupů k řešení problému

3.1 Popis systému pro detekci objektu zájmu ve sportu

Každý objekt zájmu v různých problémových doménách má svoje význačné charakteristiky a výzvy spojené s jeho detekcí. K pochopení celého procesu detekce objektů zájmu ve sportu obecně a získání referencí na práce věnující se tomuto tématu lze vyjít z rozsáhlé práce [4] na téma detekce míčů ve sportu z roku 2017.

Obrázek 3.1: Přehled systému na detekci objektu. [4]



Ohraničení čárkovaným obdélníkem značí kroky předcházející samotné detekci objektu/ů zájmu. Všechny kroky z tohoto obrázku budou krátce popsány, kdy při tomto popisu vycházím výhradně z práce [4]. Částem obsaženým v této práci se budu věnovat šířeji dále v textu této práce, pro více informací lze doporučit již zmíněnou práci [4].

První krok *camera placement* zahrnuje výběr a umístění kamery nebo kamer vzhledem ke sledovanému záměru. Více kamer přináší benefity v možnosti 3D lokace a díky snímání z více míst i menší šanci na zastínění sledovaného objektu u všech kamer.

Druhý krok *field modeling and registration* zahrnuje detekci hřiště na snímaném obrazu včetně jeho rozčleněni na různé části, například v tenisu na části

pro podání atd. Jako používané metody lze uvést *Houghovu transforamci*, *cannyho detektor rohů* apod.

Třetí krok *camera model and calibration* zahrnuje nalezení vnitřních a vnějších parametrů kamery a výběr modelu definující vztah mezi tří dimenzionálním světem a dvou dimenzionálním snímaným obrazem.

Čtvrtý krok *ground truth* obsahuje anotování snímků dle oblasti zájmu za kterým je vyvíjen daný systém, například pozice tenisového míčku na snímku.

Pátý krok *foreground extraction* odděluje *popředí* a *pozadí*. Kdy popředím jsou myšleny v tomto kroku části snímku, které mění svoji pozici.

Šestý krok *ball detection* identifikuje v extrahovaném *popředí* objekty zájmu a odděluje je od objektů, které není žádoucí detekovat.

Sedmý krok *ball tracking* identifikuje pravou trajektorii objektu zájmu. Buď kvůli nejednoznačnosti díky *falešně pozitivním* detekcím z šestého kroku, chybějícím detekcím nebo přítomnosti šumu.

3.2 Detekce objektu zájmu pomocí tradičních technik

Pátý krok *foreground extraction* z obrázku 3.1 míří na separaci *pozadí*, které je mimo oblast zájmu a *popředí*, ve kterém se snažíme detekovat objekt zájmu. Jako *pozadí* se chápe část obrazu, která je statická nebo téměř statická. Jako *popředí* jsou brány objekty, které jsou v pohybu, tedy mění významně polohu mezi jednotlivými snímky. *Tradiční techniky* využívají k detekci pohybujících se objektů ve scéně odlišnosti po sobě jdoucích snímků nebo snímku a *modelu pozadí*. Tyto odlišnosti jsou extrahovány pomocí *diference snímků* buď s *modelem pozadí* a nebo se snímky v blízkosti aktuálního snímku. Je žádoucí odlišit pohybující se objekty mimo oblast zájmu, například diváky, hráče apod.

Jako *model pozadí* je chápáno co nejvěrnější zobrazení statické scény bez pohybujících se objektů. Je také žádoucí adaptovat *model pozadí* pokud dojde ke změně pozadí, například začne svítit slunce apod. V [19] je jako *model pozadí* volen medián přes interval sousedních snímků. Je předpokládáno, že pohybující se objekty nezůstanou na jednom místě po dlouhou dobu a tedy medián hodnot *pixelu* bude odpovídat statickému pozadí. V [20] je využita *Recursive Approximate Median Method*, kdy je zvolen první snímek jako *model pozadí* a následně je vždy srovnán s aktuálním snímkem a k *pixelu modelu pozadí* je přičtena hodnota pokud je jeho hodnota menší než odpovídajícího *pixelu* aktuálního snímku a naopak. Tímto by měl *model pozadí* konvergovat ke skutečnému pozadí. V [21] je pro každý *pixel* iterativně počítán *průměr* a *variance* z určeného počtu po sobě jdoucích snímků (tzv. *okno*), tyto parametry jsou spočítány pro následující *okno* snímků a zkombinovány se stávajícími

parametry, algoritmus takto iteruje. Ovšem parametry jsou počítány jen pro *pixely* jejichž hodnota se v daném *okně* příliš nemění, tedy je na nich zobrazeno jen statické pozadí. Další možností je hodnotu každého *pixelu* modelovat pomocí *směsi gaussovských rozdělení*, která je upravována dle hodnot *pixelu* na dalších snímcích. *Popředí* je poté bráno jako hodnota *pixelu* lišící se od všech *gaussovských rozdělení* ve *směsi* o určitý násobek jejich směrodatné odchylky. Tato metoda je popsána v [22] a využita například pro modelování pozadí u fotbalového zápasu v [23].

Diference s předcházejícím/následujícím snímkem extrahuje pohybující se objekty na obou snímcích, ale zájmem je získat jen ty na snímku aktuálním. Proto autoři v práci [24] pro detekci fotbalového míče navrhují *masky* z *diference* snímku s předcházejícím snímkem a s následujícím snímkem zkombinovat pomocí logické operace *AND*, čímž by na výsledné *masce* měly zůstat pouze objekty pohybující se na aktuálním snímku. Při tomto přístupu je nutné, aby se objekt zájmu pohyboval vůči oběma sousedícím snímkům, tedy aby byla získána kvalitní *maska* pro obě *diference*, pokud je kvalitní jen jedna, poté dojde díky logické operaci *AND* k vyrušení detekce na druhé masce.

V práci [6] zabývající se detekcí tenisového míče autoři kombinují pomocí logické operace *AND masku* z *diference* s následujícím snímkem a *masku* z *diference* s *modelem pozadí*, která zaručí extrakcí objektů pohybujících se pouze na aktuálním snímku.

Přístup pro detekci basketbalového míče, který kombinuje dva předešlé přístupy je uveden v [25]. Jsou vytvořeny 3 *masky* pomocí *diference* s předcházejícím snímkem (*maska A*), následujícím snímkem (*maska B*) a *modelem pozadí* (*maska C*). Dále je pomocí logické operace *AND* zkombinována *maska A* s *maskou C* a *maska B* s *maskou C*. Tyto dva výsledky jsou zkombinovány pomocí logické operace *OR*. V tomto postupu stačí, aby se objekt zájmu pohyboval jen vůči jednomu ze sousedních snímků.

V práci [26] autoři řeší problém detekce pohybu objektu zájmu pomocí *diference* snímků pokud se pohne kamera. Poté se jako v pohybu jeví například hrací čáry. Autoři se toto snaží řešit detekcí pohybu kamery a následnou segmentací hracích čar. Dle jejich tvrzení toto značně redukuje počet falešných detekcí z důvodu pohybu kamery.

V práci [27] využívají *diference* snímků od sebe více vzdálených, aby nedocházelo k překryvu tenisového míčku, což může nastat na sousedních snímcích. Místo *modelu pozadí* využívají barevnou segmentaci snímku dle přibližné barvy tenisového míčku a kromě *diference* snímků ve výchozím barevném spektru uplatňují i *diferenci* snímků *segmentovaných* dle přibližné barvy tenisového míčku.

Na výstup z výše zmíněného postupu je následně uplatněna operace *prahování (thresholding)*, která rozdělí *pixely* na dvě kategorie dle toho, zda je jejich hodnota větší nebo menší nežli zvolený *práh*. *Pixely* které mají hodnotu větší jsou

brány jako *popředí* a je s nimi dále pracováno. Výsledná *maska popředí* může být k odstranění šumu upravena pomocí *morfologických operací*. Například v [26] volí autoři operaci *otevření* následovanou operací *uzavření*, v [6] volí operaci *dilatace*. V [28] zabývající se detekcí míčku pro stolní tenis je uvedena metoda *growth of sampling points*, která slouží k doplnění obrazce na kruh. Tato metoda míří na problém překryvu míčku ve dvou po sobě jdoucích snímcích, kdy jejich *diference* vytvoří *"srp"*. Doplnění probíhá dle barvy *pixelů* okolo detekovaného obrazce.

Shluky *pixelů* přiřazených do *popředí* tvoří obrazce, které jsou hodnoceny vzhledem k příslušnosti k objektu zájmu. Dle [4] se jako hlavní charakteristiky pro detekci míče ve sportovních hrách využívají následující charakteristiky. Tyto mohou sloužit jak k lokalizaci míčku, tak k vyloučení falešných detekcí:

1. barva,
2. velikost,
3. tvar,
4. předpokládaná pozice a trajektorie,
5. míra shody s před připraveným vzorem neboli *template matching*,
6. rychlost.

Zmíněné charakteristiky, zvané také *příznaky*, mohou následně sloužit jako vstup do klasifikátoru jako *rozhodovací stromy*, *neuronové sítě* a jiné, nebo jsou intervaly jejich hodnot nalezeny jiným způsobem a následně slouží jako rozhodovací pravidlo pro klasifikaci objektů.

Pro klasifikaci *kontury* jako tenisového míčku je v [29] využita velikost a barva, v [6] velikost, kompaktnost a poměr hlavní a vedlejší osy, v [25] velikost, excentricita a kulatost, v [26] je využita velikost a poloha, kdy míčky detekované v oblastech s malou pravděpodobností výskytu jsou odstraněny, v [27] velikost, barva, excentricita a výstup z *Harrisova detektoru rohů*.

Z hlediska úspěšnosti detekce tenisového míčku tradičními technikami autoři v [6] uvádějí, že správně detekovali tenisový míček na 90 % snímcích. V [26] autoři uvádějí výsledky na metrice *precision* úspěšnost 96 % a na metrice *reccal* úspěšnost 98.1 %. Pro definice metrik viz kapitolu 6.

3.3 Detekce objektu zájmu pomocí hlubokého učení

Jak již bylo zmíněno v sekci 2.3.3 při zvolení přístupu *hlubokého učení* tvůrce systému nevolí *příznaky* určené pro detekci objektu zájmu, nýbrž jsou tyto *příznaky* zvoleny při *procesu učení*. Výhodou tohoto přístupu také je, že většinou nevyužívá žádné kroky předzpracovávající snímky před samotnou detekcí objektů zájmu oproti *tradičním technikám*.

Jedno z prvních využití *deep learning* technik pro detekci tenisového míčku přinesla práce [9]. *CNN* rozhoduje zda na obdélníkovém výřezu z obrázku je nebo není tenisový míček. Autoři zmiňují, že budoucí práce by měla vést k integraci *CNN* do komplexních detekčních systémů, kde bude *CNN* využívána spolu s *tradičními metodami*. Autoři uvádějí dosažení následujících výsledků pro detekci tenisového míčku *precision* 98 %, *recall* 96 % a F_1 *skóre* 97 %.

Dalším zástupcem využívajícím pro detekci tenisového míčku *CNN* z roku 2020 je práce [30]. Autoři využívali data z tenisového zápasu z [5] a obrázky míčků z datové sady *MS-COCO*. Autoři uvádějí dosažení následujících výsledků *precision* 91.77 %, *recall* 93.43 % a F_1 *skóre* 92.59 %.

Novějším zástupcem [31] z roku 2021 je přístup využívající detekci míčku na stolní tenis *CNN* a k predikci příští lokace *Long short term memory rekurentní neuronové sítě*, zkráceně *LSTM RNN*. Tato predikce slouží jako náhrada pokud selže detekce pomocí *CNN*.

Jako zástupce *3D CNN* přístupu pro detekci tenisového míčku, kdy jako vstup do *neuronové sítě* slouží více po sobě jdoucích snímků lze uvést práci [5] z roku 2019 ve které je představena *3D CNN* nazvaná *TrackNet*. *Neuronová síť* se *učí* z více po sobě jdoucích snímků, aby jí bylo umožněno *učit se* z a následně využit informace o trajektorii míčku. *Neuronová síť* se učí na datové sadě, kde je vyznačena pozice míčku pomocí 2D *gaussovy distribuce* se středem ve středu míčku a variancí rovnou přibližnému průměru míčku vyjádřenou v počtu *pixelů*. Samotná *neuronová síť* je složena z architektur *konvolučních neuronových sítí VGG-16* a *deconvNet*, které dohromady tvoří *encoder-decoder* architekturu. Výstupem je *teplotní mapa (heatmap)*, kde pro je každý *pixel* přiřazeno skóre. Následně jsou pomocí *prahování* zvoleny *pixely* se skórem vyšším než *práh* a v těchto je hledán pomocí *Houghovy transformace* kruhový útvar. Pokud je nalezen právě jeden, je vrácen jako detekce. Autoři testovali jako vstup jeden a tři po sobě jdoucí snímky, kdy při druhém dosáhli lepších výsledků a zmiňují i schopnost predikovat polohu zastíněných míčků ve 2 ze 7 případů. Dále autoři porovnali svůj přístup s přístupem z [6], který zastupuje *tradiční techniky*, popis viz 3.2.

V tabulce 3.1 pocházející z práce [5] jsou uvedeny výsledky pro *tradiční techniku* [6] popsanou v sekci 3.2, *Tracknet 1-1* značí verzi s jedním vstupním snímkem, *Tracknet 3-1* značí verzi s třemi vstupní snímky a *Tracknet-2* značí

Model	Precision	Recall	F1-measure
Tradiční techniky [6]	92.5 %	74.5 %	82.5 %
Tracknet 1-1	95.7 %	89.6 %	92.5 %
Tracknet 3-1	99.8 %	96.6 %	98.2 %
Tracknet-2 3-1	99.7 %	97.3 %	98.5 %
Tracknet-2 3-1 s 10-fold C.V.	95.3 %	75.7 %	84.3 %

Tabulka 3.1: Tabulka s výsledky z [5] pro *CNN Tracknet*. Tradiční technika [6] je popsaná v sekci 3.2, Tracknet 1-1 značí verzi s jedním vstupním snímkem, Tracknet 3-1 značí 3 vstupní snímky a Tracknet-2 značí trénování na rozšířené datové sadě.

trénování na rozšířené *trénovací datové sadě*. Všechny *modely* byly testovány na stejné *testovací datové sadě* kromě posledního uvedeného, který byl testován na rozšířené *testovací datové sadě* pomocí *10 násobné křižové validace (10-fold cross validation)*. Základní datová sada obsahuje 20844 snímků z jednoho tenisového utkání, rozšířená datová sada obsahuje navíc 16118 snímků z odlišných tenisových povrchů.

Lze nahlédnout, že *neuronová síť Tracknet* má výrazně lepší výsledky oproti *tradiční technice* [6] s výjimkou výsledku *Tracknet-2 3-1* s testováním a *trénováním* pomocí zmíněné *křížové validace* na rozšířené *trénovací datové sadě*, kdy lze pozorovat výrazné zhoršení na metrice *recall*, v důsledku toho i metriky *F1-measure*, a výsledky se blíží *tradiční technice* [6], která ovšem byla testována pouze na původní datové sadě. Toto je bohužel nedostatek, který stěžuje skutečné porovnání těchto metod. Autoři se důvody zmíněného propadu nezabývají. Nabízí se dvě hypotézy, *neuronová síť* mohla být původně *přeučena* na jeden tenisový povrch, kdy s obohacením o více tenisových povrchů mohlo dojít při *učení neuronové sítě* k větší generalizaci pro detekci míčku. Nicméně u výsledku *Tracknet-2 3-1*, kdy je síť testována na původní *testovací datové sadě* k propadu nedošlo ani při *trénování* na rozšířené *trénovací datové sadě*, což tuto hypotézu nabourává. Další možností je, že *testovací datová sada* utvořená ze základní datové sady byla jednodušší, nežli *testovací datové sady* utvořené při z rozšířené datové sady. Nicméně síť *Tracknet* si i tak počínala lépe než zástupce tradičních technik [6].

S *neuronovou sítí Tracknet* je dále pracováno pro použití na detekci badmintonových míčků v [7], kde tuto *neuronovou síť* autoři nazývají *TracknetV2*. *Neuronová síť* je vylepšena z hlediska výpočetního času a paměťové náročnosti pomocí dílčích úprav *neuronové sítě*. Dále je experimentováno s nastavením pro n vstupních snímku predikovat n výstupních místo pro n vstupních snímku predikovat 1 výstupní, tedy pro jeden snímek *neuronová síť* vyprodukuje až n predikcí, autoři neuvádějí jak těchto n snímků kombinují do výsledné predikce. Je přidána technika pro zachování informace přes více *vrstev neuronové sítě*,

tzv. *skip connections* a je změněna *ztrátová funkce*. Výsledky v tabulce 3.2 ukazují pro verzi s 3 výstupními snímky zlepšení v metrice *precision* a lehký pokles metriky *recall*. Tedy 3 predikce pro jeden snímek míček zaznamenají v méně případech, ale když ano, tak ve více případech správně.

Model	Precision	Recall	F1-measure
Tracknet 3-1	85.0 %	57.7 %	68.7 %
TracknetV2 3-1	90.7 %	89.2 %	89.9 %
TracknetV2 3-3	97.2 %	85.4 %	90.9 %

Tabulka 3.2: Tabulka s výsledky z [7] pro *CNN Tracknet* a *TracknetV2* pro detekci badmintonového míčku. Značení 3-1 značí verzi s třemi vstupními a jedním výstupním snímkem jako predikcí, značení 3-3 značí 3 vstupní snímky a 3 výstupní snímky, kdy každý je brán jako nezávislá predikce.

Vzhledem ke zveřejněným výsledkům lze porovnat pouze rychlost zpracování, kterou přinesli zmíněné úpravy. Úspěšnost detekce *neuronové sítě Tracknet trénované* na *trénovací datové sadě* pro badmintonový míček a *neuronové sítě TracknetV2 trénované* na odlišné a podstatně větší *trénovací datové sadě* pro badmintonový míček, 56 tisíc oproti 18 tisícům snímků, nelze díky této odlišnosti *trénovacích datových sad* přímo porovnat. *TracknetV2* na podstatně větší datové sadě oproti *neuronové síti Tracknet*, 56 tisíc oproti 18 tisícům snímků. Výsledky jsou uvedeny v tabulce 3.2. Na první pohled druhá verze *neuronové sítě* ukazuje lepší výsledky, bohužel ale nelze říci, zda je příčinou větší datová sada, nebo zmíněné změny.

V [8] je pro detekci fotbalového míče, což je podle autorů těžší úloha než detekce tenisového míčku, využita kombinace detekce *CNN* a předzpracovaní snímků *tradičními technikami*. V tomto přístupu jsou nejdříve předzpracovány jednotlivé snímky pomocí *diference* s modelem pozadí, který je předpokládán fixní, následně jsou použity *morfologické operace* pro zacelení *masek* pohybujících se objektů. Poté je na takto upravené snímky použita *CNN*, která klasifikuje objekty jako potencionální míče. Po zpracování všech snímků se v tabulce odstraní pravděpodobné *falešně pozitivní* detekce, které nemají v okolních snímcích blízkou detekci. Pokud je pro snímek více detekcí nežli jedna, zvolí se ta nejbližší detekci z předchozího snímku. Algoritmus má na datové sadě autorů *accuracy* 69 %, další metriky nejsou uvedeny.

Dalším přístupem uvedeným v [32] je hledání pohybujícího se objektu na snímku pomocí rozmazání (*blur*). Přístup stojí na myšlence, že směr rozmazání souvisí se směrem pohybu objektu. Metoda se snaží nejdříve separovat ze snímku rozmazání objektu a zbytek snímku. Z nalezeného rozmazání určit směr pohybu objektu na tomto snímku pomocí kvadratických křivek. Pomocí nalezeného směru pohybu na snímku se predikuje poloha na dalším snímku

a proces se opakuje pro další snímek. Tento přístup je odlišný tím, že hledá trajektorii pohybu i v jednotlivých snímcích a neodhaduje ji jen z polohy v po sobě jdoucích snímcích. V navazující práci [33] je z detekovaných trajektorií pomocí rozmazání na jednotlivých snímcích vytvořena spojitá trajektorie mezi snímky. Nejdříve je pohyb objektu rozdělen na sekvence mezi prudkými změnami směru (odrazy, dopady) a tyto sekvence jsou následně modelovány polynomem až do stupně 6. Přístup je dále rozpracován a implementován pomocí *CNN* ve formě *encoder-decoder* architektury v [34]. V kontextu záznamu tenisového utkání je otázkou, jak by si tento přístup počínal pro snímky na kterých není významné rozmazání, například když se objekt pohybuje pomaleji nebo je v takovém úhlu vůči kameře, kdy není rozmazán ani pří velké rychlosti.

3.4 Odhadování trajektorie

S detekcemi objektu zájmu na snímcích přichází úloha vybrat na snímcích ty, které korespondují se skutečnou polohou objektu, tedy odstranit *falešně pozitivní* detekce. Vybrané detekce by měli tvořit souvislou trajektorii odpovídající předpokládané dynamice objektu, případně dalším skutečnostem. Dle [12] jsou pro tuto úlohu dostupné dvě informace a to vzhled objektu a znalost dynamiky objektu. U tenisového míčku je dle [35] největším problémem náhlá změna rychlosti a směru, ke které dochází při dopadu míčku a úderu míčku hráčem. Dle [4] je významným problémem zastínění objektu, například hráči.

Dále jsou v [12] metody děleny na *track-after-detection (TAD)*, kdy jsou nejdříve detekovány potencionální pozice míčku ve všech snímcích a následně z těchto detekcí vytvořena trajektorie a *track-before-detection (TBD)*, kdy je nejdříve odhadována trajektorie, která je hodnocena podle toho, jak odpovídá detekcím míčku na snímcích. Autoři uvádějí, že *TAD* přístup je vhodnější k detekci malých, rychlých objektů s jednoduchou dynamikou, naopak *TBD* přístup je vhodný pro velké, pomalé objekty se složitější dynamikou.

Odhadování trajektorie lze dle [12] rozdělit na dvě podúlohy:

- *Data association*, kdy je cílem nalézt výskyt objektu zájmu a nedopouštět se *falešných detekcí*.
- *Data estimation* má za cíl odfiltrovat šum přítomný v datech a odhalit pravý stav objektu zájmu v reálném světě.

Tedy pomocí *data association* je odhalen pravý výskyt objektu na snímcích, například míčku, a následně jsou v kroku *data estimation* jednotlivé pozice vybraných detekcí případně usměrněny k pravému stavu objektu. [12]

Kvalita odhadu trajektorie závisí také na míře granularity se kterou je modelována dynamika pohybu objektu. [4]

Níže popíši přístupy ke sledování trajektorie a vybrané zástupce těchto metod pro ilustraci jejich využití, kdy převážně vycházím z práce [4].

3.4.1 Kalmanův filtr

Jednou z možností dle [4] je *Kalmanův filtr*, který se nejvíce hodí pro lineární systémy s *gaussovskými* stavy a šumem nebo pokud se tomu alespoň blíží. Dále uvádí, že *KF* je vhodný pokud objekt není dlouho zastíněn hráči nebo prostředím, pohybuje se lineárně, pomalu, s konstantní akcelerací a přiměřeně rychle. *Kalmanův filtr* může sloužit pouze k *data estimation* úloze nebo ho lze využit i k *data association* například využitím jeho predikce příští polohy pro vyloučení *falešně pozitivních detekcí*.

3.4.2 Particle filtr

Pokud je pohyb míčku silně nelineární je dle [4] vhodnější využít *Particle filter* místo *Kalmanova filtru*. Nicméně dle [4] si *Particle filter* nedokáže dobře poradit pokud se vyskytuje pozadí s podobnými objekty, malý rozměr, rozmazání nebo zastínění objektu, změna vizuální podoby apod., jsou tedy vyvíjeny speciální upravené verze pro danou doménu problému. Pokud díky těmto nebo jiným problémům dojde ke ztrátě informace o pozici objektu je vhodné mít *redetekční mechanismus*, který znovu nasměruje *Particle filter* ke sledování správné trajektorie. [4]

V práci [35] byly použity dva stavové modely pro *PF* ke sledování trajektorie tenisového míčku, jeden standardní a druhý pro moment, kdy je míček trefen hráčem a tedy významně mění svůj směr. Modely využívají 4 stavové proměnné a to pozici a rychlost ve vertikální a horizontální ose. Dle [4] se často využívá jako stavová proměnná též barva.

3.4.3 Přístup *Trajectory*

Další rodinou metod, kterou uvádějí autoři v [4] je *Trajectory*. Tyto metody detekují nejdříve potencionální kandidáty ke klasifikaci na jednotlivých snímcích. Následně pomocí těchto kandidátů tvoří trajektorii nebo více trajektorií, které jsou hodnoceny dle předem určeného kritéria, například fyzikálního modelu pohybu objektu, blízkosti potencionálním detekcím apod. Nebo dle predikce pozice na snímku přiřazují k budované trajektorii potencionální detekce. Tedy hlavní výzvou přístupu není přesná detekce právě jednoho pravého objektu, ale následné správné formování trajektorie. K tvorbě trajektorie je často využíván *Kalmanův filtr*. Tyto metody se lépe potýkají s chybějící detekcí na snímku než předešlé metody, tedy samotný *Kalmanův filtr* a *Particle filtr*. Tento přístup je většinou navrhnut jako *offline*, tedy potřebuje nejdříve detekovat potencionální kandidáty na všech snímcích. [4]

V práci [10], je použita detekce úderu hráčem jako informace o začátku a konci trajektorie tenisového míčku. Z místa úderu jsou pomocí nalezených potenciálních míčku generovány trajektorie mezi jednotlivými snímky *Kalmanovým filtrem* a jsou hodnoceny podle blízkosti k jednotlivým potencionálním detekcím. Poté jsou vybírány trajektorie s největším skóre a odstraněny ty, které se s nimi překrývají.

V práci [36] je využito fyzického modelu pohybu tenisového míčku k vytvoření množiny trajektorií, které se následně modifikují pomocí *genetického programování* k vytvoření optimální trajektorie, kdy trajektorie jsou ohodnoceny pomocí detekcí pozic míčku lokalizovaných na jednotlivých snímcích.

K nalezení trajektorie tenisového míčku je v [27] využit *Kalmanův filtr* s *pohybovými rovnicemi*, který odhaduje polohu tenisového míčku a následně je vybrána detekce odhadu nejbližší. Autoři uvádějí, že na jednom snímku jejich postup založený na *tradičních technikách* nalezne v průměru 8.5 potencionálních kandidátů.

3.4.4 Přístup *Data association*

Techniky sdružené v tomto přístupu se nazývají *data association*, kdy bohužel dochází k určité dvojznačnosti se samotným problémem *data association*, který je možné řešit i jinými technikami zmíněnými v této sekci.

Přístup *trajectory* budoval trajektorii a tu následně hodnotil, případně dle trajektorie určoval pravé detekce objektu. Tento přístup buduje trajektorii z potencionálních detekcí na jednotlivých snímcích jejich spojováním dle zvolených kritérií. Tímto vznikají kratší trajektorie, které jsou následně spojovány, čímž vznikne celá trajektorie míčku. Tyto techniky jsou většinou navrhnuty jako *offline*, tedy potřebují nejdříve detekovat potencionální kandidáty na všech snímcích. [4]

Jako hlavní zástupce uvádí [37] techniky *RDA*, *LDA*, *TLDA* a *Viterbi algoritmus*. Porovnání těchto přístupů lze nahlédnout v [37], kdy nejlépe z testování vyšla technika *TLDA*.

Dle [4] je technika *data association* vhodná pro využití pro sledovaní trajektorie tenisového míčku:

Layered DA methods...They solved the problem of occlusion only in Tennis sports. (citováno z [4])

DA methods have been introduced; but they were more confined to problem of ball tracking in Tennis that is considered solved. (citováno z [4])

3.4.5 Přístup optimalizující *účelovou funkci*

V tomto přístupu existuje *účelová funkce* hodnotící zvolené detekce objektu zájmu. Cílem modelu je minimalizovat tuto *účelovou funkci* svými rozhodnutími. *Účelová funkce* může být například model naučený predikovat budoucí pozicí objektu na základě minulých událostí včetně chování hráčů. Tato technika je využita pro sporty s více hráči, kde je míč často zastíněn a je často měněna jeho trajektorie hráči. [4]

3.4.6 Algoritmus *TLDA*

Algoritmus nejdříve rozdělí sekvencí snímků do po sobě jdoucích částí zvaných *okna*. Tyto *okna* mají stejnou velikost a mohou se mezi sebou překrývat dle nastavení velikosti *okna* a posunu *okna* přes snímky vůči předchozímu *oknu*.

Dále je vytvořen *orientovaný graf*, bude značen G, ve kterém *uzly* reprezentují jednotlivé potencionální detekce míčku a hrana z *uzlu* A vede do *uzlu* B, pokud jsou od sebe souřadnice detekcí, které *uzly* reprezentují, vzdáleny v definovaném rozmezí a snímky na kterých se detekce nachází jsou sousední a detekce *uzlu* A předchází detekci *uzlu* B.

V jednotlivých *oknech* pokrývajících jim přidělený interval snímků jsou budovány trajektorie hledáním všech možných *cest* v grafu G postupným iterováním od prvního po poslední snímek *okna*. Pokud nějaká *cesta* nemá v *orientovaném grafu* G následníka prodlužuje ji algoritmus omezeným množstvím umělých detekcí, aby mohla navázat na detekcí vzdálenou o více než jeden snímek. *Cesty* jsou ohodnoceny dle pohybového modelu využívajícího rychlost a akceleraci, kdy je učiněna predikce další detekce a je porovnána se skutečnou detekcí reprezentovanou *uzlem* přidávaným do *cesty*. Po každé iteraci je uložen určitý počet nejlépe ohodnocených *cest*, tedy jsou uloženy nejlepší *cesty* ze všech uvažovaných intervalů snímků *okna*.

Následně jsou odstraněny trajektorie, reprezentované nalezenými *cestami*, které jsou duplikátní napříč okny. Dále mohou být odstraněny *cesty*, které popírají nějakou apriorní vlastnost, kterou očekáváme, například směr pohybu míčku z jedné strany kurtu na druhou. Pokud *cesta* vykazuje příliš mnoho velkých odchylek od předpokládaného pohybového modelu vzhledem ke své délce, je smazána.

V dalším kroku jsou spojovány trajektorie, reprezentované *cestami*, v jednotlivých *oknech* a poté mezi *okny* a to následovně, je vytvořen *ohodnocený orientovaný acyklický graf* tak, že jednotlivé *cesty* jsou *uzly* a vede mezi nimi *hrana* pokud se překrývají časově a prostorově nebo pokud se nepřekrývají časově, ale jejich spojení pomocí *lineární interpolace* vyhoví určenému kritériu na délku interpolovaného propojení. *Váha hrany* je ohodnocení *cesty* do které *hrana* vede. *Hrana* je orientována dle časového hlediska. V případě spojování mezi *okny* vstupuje do podmínek také maximální rozestup v počtu *oken* mezi

spojovanými *cestami*, tento by neměl být ani příliš malý, neboť může dojít k chybějícím detekcím a tedy je nutné spojit trajektorie nacházející se v *oknech* více vzdálených, ale ani příliš velký, neboť by mohlo dojít k přeskočení části trajektorie, která nemá příliš dobré hodnocení, nicméně je to korektní část celé trajektorie.

Na závěr je pomocí *dynamického programování* nalezena *cesta* v grafu, který slouží pro spojování *cest* mezi *okny*, s nejlepším ohodnocením a tato slouží jako výstup reprezentující výslednou trajektorii míčku.

Zmíněný postup se snaží najít jednu trajektorii přes všechny snímky. Je tedy vhodný pro jednotlivé tenisové výměny. V práci [12] představující metodu *LDA* stojící též na přístupu *Data association* přidávají autoři jako poslední krok techniku, která dle jejich tvrzení dokáže odlišit trajektorie jednotlivých tenisových výměn. Postup spočívá ve výpočtu vzdálenosti mezi všemi *uzly* grafu, reprezentujícími dílčí trajektorie, a následném vybrání nejlepších *cest*, které se nepřekrývají, kdy je bonifikována kromě ohodnocení *cesty* též její délka. Tento postup je ovšem výpočetně náročný, neboť počítá vzdálenost mezi všemi *uzly*, kterých bude u záznamu tenisového utkání, obsahujícím mnoho tenisových výměn, velké množství. Výpočetní složitost například *Floyd-Warshallova* algoritmu je $\mathcal{O}(n^3)$. Za vhodnější považuji rozložení záznamu tenisového utkání na jednotlivé výměny jiným způsobem, v nichž bude následně hledána trajektorie.

Provedené změny v algoritmu *TLDA*

Nahradil jsem zmíněné *dynamické programování* algoritmem *Floyd-Warshall*, který spočte vzdálenost mezi všemi *uzly* grafu. Následně jsou hledány *cesty* s nejlepším skóre, které se nepřekrývají. Pokud je nalezeno více *cest*, reprezentujících trajektorie, o dostatečné délce v rámci jedné tenisové výměny, kdy je tato délka počítána pouze pro trajektorie obsahují detekci v oblasti mezi horizontálními čarami pro podání, čímž odstiňuji trajektorie, které mohou pocházet z šumu, je zvětšen maximální rozestup v počtu *oken* mezi spojovanými *uzly* grafu. Tímto se snažím řešit problém, kdy po delší čas chybí detekce míčku, ať už důvodu nedokonalosti předchozího kroku detekujícího míček na snímcích, tak z důvodu opuštění míčku záběru kamery. Pokud je i po tomto nalezeno v grafu více *cest*, jsou odstraněny ty, které nemají žádnou detekci v prostoru vytyčeném dvěma prostředními horizontálními čárami, tedy čárami pro podání, a ze zbylých je vybrána ta nejdelší.

Neodstraňuji duplicitní *cesty* mezi *okny*, neboť toto může omezovat spojování *cest* mezi *okny*, které je omezeno maximální vzdáleností v počtu těchto *oken*, kdy duplicitní *cesty* mohou sloužit jako spojnice mezi vzdálenějšími *okny*.

K omezení výpočetní náročnosti jsem přidal ořezání počtu budovaných *cest* dle jejich ohodnocení po každé iteraci zpracovávající další snímek v *okně*.

Dále jsem doplnil výběr určitého počtu nejlépe ohodnocených *cest* ze všech uložených cest pro každé *okno*.

Využívám odstranění trajektorií, které kopírují základní horizontální tenisové čáry, neboť u těchto čar docházelo k častým falešně pozitivním detekcím v předchozím kroku detekujícím tenisový míček na snímku. Je omezena maximální vzdálenost v souřadnici y pro spojení dvou trajektorií, reprezentované *cestami*, na délku hřiště v souřadnici y.

Změnil jsem výpočet skóre ohodnocující budované cesty následovně:

$$S_1 = \log\left(\frac{d - d_m}{d_M - d_m}\right) \text{ na } S_1 = \log\left(\frac{d - d_m}{d_M - d_m}\right) - \log\left(\frac{d_M/2 - d_m}{d_M - d_m}\right),$$

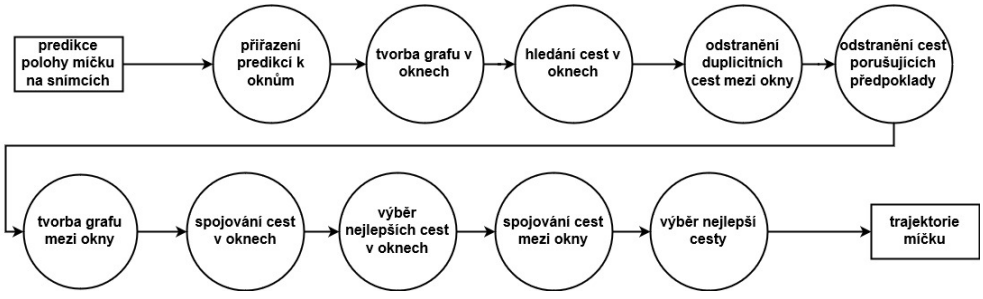
$$S_2 = \log\left(\frac{\theta}{180}\right) \text{ na } S_2 = \log\left(\frac{\theta}{180}\right) - \log\left(\frac{45}{180}\right),$$

$$Score = S_1 + S_2,$$

kde d je vzdálenost mezi predikovanou a skutečnou detekcí přidávající se do *cesty*, d_m a d_M je minimální a maximální vzdálenost mezi sousedními detekcemi v *cestě* a θ je úhel mezi vektorem vedoucím z poslední detekce v *cestě* do predikované detekce a do skutečné detekce přidávající se do *cesty*.

Před změnou skóre byla vždy hodnota S_1 a S_2 negativní, díky čemuž měla každá *cesta* negativní skóre a vyplatilo se ji přidat do výsledné trajektorie, pokud to bylo možné a pokud nebyla odstraněna díky příliš mnoho velkým odchylkám od předpokládaného pohybového modelu, kdy příliš velká odchylka je definována jako úhel θ větší než nastavená *prahová* hodnota. Po změně skóre je hodnota S_1 pozitivní, jestliže je predikce vzdálena o více než polovinu maximální povolené vzdálenosti mezi sousedními detekcemi a hodnota S_2 je pozitivní, jestliže je zmíněný úhel θ větší než 45 stupňů. Tímto dojde k ohodnocení pozitivní hodnotou cest, které se nechovají podle použitého pohybového modelu a i když by to bylo možné, tak nemusí být přidány do výsledné trajektorie a tedy není nutné používat zmíněnou prahovou hodnotu pro úhel θ pro odstraňování *cest*.

Obrázek 3.2: Diagram algoritmu *TLDA*.



3.5 Vytěžování informací z trajektorie

Informace o tenisových událostech, které poskytuje trajektorie, jsou obsaženy ve změnách směru, rychlosti a zrychlení míčku, které se dějí při dopadu nebo úderu [38], [39]. V [39] rozdělují událost na úder a dopad dle blízkosti události k pozici hráče. V [38] je pomocí datové sady *natrénován skrytý Markovův model* využívající informaci o pozici a první a druhé derivaci, který následně s pomocí *gramatiky* vyznačující povolené sekvence stavů klasifikuje jednotlivé události. V [40] je využívána zvuková informace, která obohacuje informace obsažené v trajektorii.

Kapitola 4

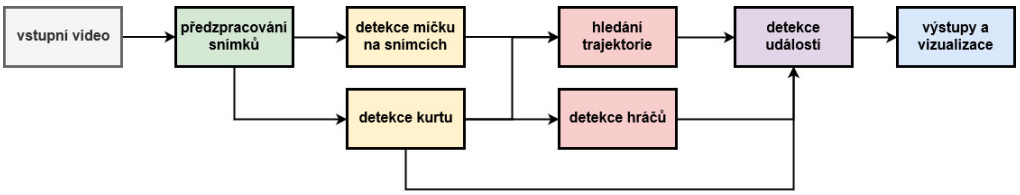
Návrh řešení

Struktura řešení problému je prezentována ve formě diagramu na obrázku 4.1. Systém se skládá ze sekvence kroků, které jsou na sobě různě závislé, což je značeno šipkami v diagramu. Jednotlivé části řešení budou popsány níže v příslušných podkapitolách.

Systém předpokládá jako vstup čísla počátečních a koncových snímků videa ohraničujících tenisovou výměnu, která začíná podáním jednoho z hráčů. K plně automatickému zpracováni záznamu tenisového utkání je nutné tuto funkcionalitu doplnit. V práci toto není obsaženo. Předpokládá se, že není během výměny změněna snímající kamera, tedy se radikálně nemění pohled na snímanou scénu. Také je očekáván typ snímání, kdy se kamera nachází za jedním z hráčů, kdy se páska sítě nachází mezi dvěma horizontálními čárami pro podání, viz obrázek 2.2.

Návrh řešení se omezuje na *dvouhru*.

Obrázek 4.1: Návrh systému na detekci trajektorie tenisového míčku.



4.1 Detekce míčku na jednotlivých snímcích

K detekci tenisového míčku na jednotlivých snímcích jsem se rozhodl vyzkoušet jak přístup využívající *tradiční techniky*, tak přístup založený na *hlubokém učení*.

Konvoluční neuronová síť

Pro přístup využívající *hluboké učení* jsem vybral model *CNN TrackNet*, viz sekci 3.3, která je určena přímo k detekci tenisového míčku a navíc je tento model volně dostupný včetně natrénovaných vah z [41]. *Model* má jako vstup snímek na kterém chceme detekovat tenisový míček a dva snímky předcházející. Výstupem je *teplotní mapa (heatmap)* zobrazující pravděpodobnost výskytu míčku na jednotlivých *pixelech* snímku. Tato *teplotní mapa* je pomocí *prahování* převedena na binární *masku* ve které jsou hledány *kontury* mající tvar kružnice. Tyto jsou určeny jako detekce tenisového míčku. Celý postup je znázorněn na obrázku 4.4.

Tradiční techniky

Pro přístup využívající *tradiční techniky* jsem vybral pro tvorbu *masky popředí* přístup [25], který bere v potaz rozdíl oproti předcházejícímu, následujícímu snímku a *modelu pozadí*, který jsem zvolil jako *medián* z okolních snímků, dále přístup [24], který bere v potaz pouze rozdíl oproti předcházejícímu a následujícímu snímku. Vybrané přístupy naleznou *masku popředí*, ve které jsou nalezeny *kontury*, jejichž charakteristiky slouží jako vstup do *modelu* strojového učení, který určí, zda *kontura* reprezentuje tenisový míček. Byly využity následující charakteristiky *kontur*, které uvádím v anglickém znění, *area*, *aspect ratio*, *circularity*, *solidity*, *extent*, *arc len*, *eccentricity*, *blue color*, *green color*, *red color*, *centroid x*, *centroid y*. Celý proces je zobrazen na obrázku 4.2. Pomocí datové sady [42] jsem vytvořil jednoduchou datovou sadu, kdy *kontury* vzdálené pár pixelů od skutečné pozice míčku byly brány jako *kontury* míčku. Na této datové sadě jsem následně natrénoval *model strojového učení*.

Výběr metody

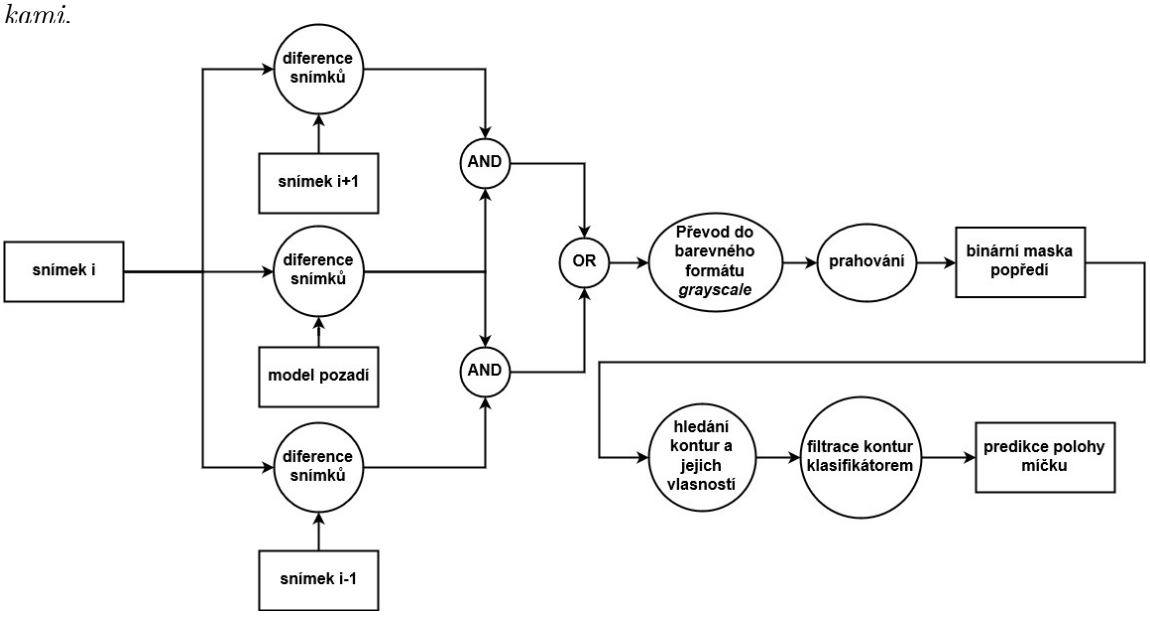
Bohužel se mi u *tradičních technik* výše zmíněným postupem nepodařilo v dostatečné míře odfiltrovat *falešně pozitivní* detekce míčku a to obzvláště vzhledem k dalším hýbajícím se objektům ve scéně, zejména hráčům. Dalším problémem je pohybující se kamera, kdy se v pohybu jeví celá scéna a tedy dojde k velkému množství *falešně pozitivních* detekcí míčku a u přístupu [25] by bylo nutné zvolit komplexnější *model pozadí* reagující na pohyb kamery. Tato

skutečnost jednak zapříčiní zvýšenou časovou náročnost navazujícího kroku, který hledá trajektorii míčku, ale také zvyšuje pravděpodobnost vybrání těchto *falešně pozitivních* detekcí do trajektorie a zanesení informace, která způsobí nesprávnou detekci *události*. Tento problém lze částečně řešit vynecháním detekcí z oblastí, kde se *falešně pozitivní* detekce vyskytují ve vysoké míře a to jsou tenisové čáry a oblast s obecenstvem. Nicméně tímto přístupem ztrácí postup robustnost, neboť míček se v těchto oblastech může vyskytovat. Z výše uvedených důvodů jsem se rozhodl využít zmíněnou *CNN TrackNet*, která sice také, obzvláště při pohybu kamery, detekuje *falešně pozitivní* pozice míčku, nicméně v podstatně menší míře. Dalším problémem u *CNN TrackNet* je občasné nedetekování míčku, pokud má jeho kontura nestandardní charakteristiky, jedná se například o tenisové podání, kdy míček nabude velké rychlosti a jeho kontura je výrazně roztažena do tvaru připomínající elipsu, dále pokud se míček překrývá s hráčem, poté není vždy detekován, toto jsou skutečnosti, kterých jsem si povšiml při pozorování.

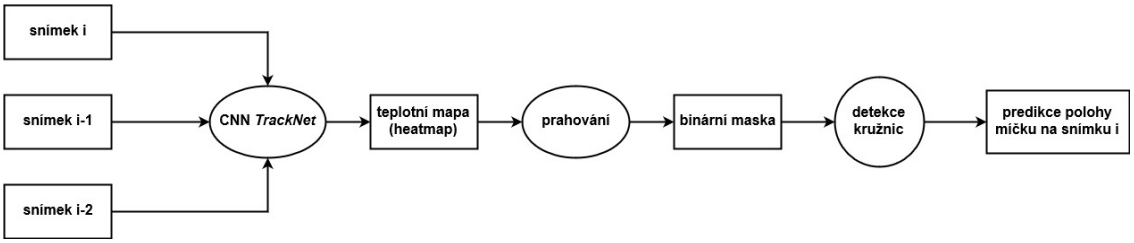
Návrh paralelizace

Implementace *CNN TrackNet* běží paralelně na *GPU*. Pro větší míru paralelizace by bylo nutné využít více *grafických karet*, kdy by na každé *grafické kartě* běžela tato *CNN* a snímky by byly rovnoměrně rozděleny, neboť je lze zpracovávat nezávisle. Toto není v této práci implementováno.

Obrázek 4.2: Diagram pro řešení detekce míčku na snímku *tradičními technikami*.



Obrázek 4.3: Diagram pro řešení detekce míčku na snímku pomocí *CNN*.



4.2 Odhadování trajektorie

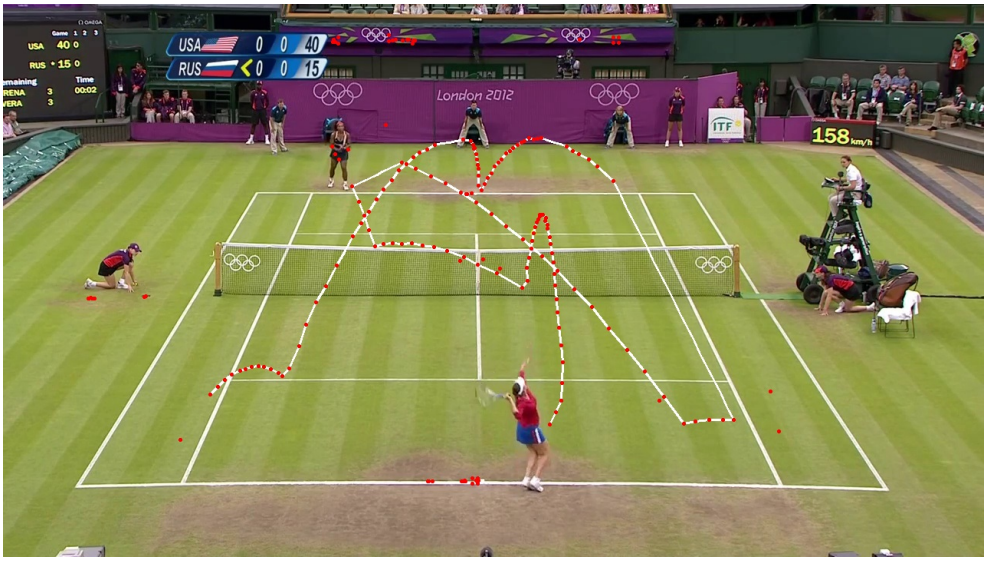
K řešení této úlohy jsem vybral přístup *Data association* a metodu *TLDA*, která dle jejich autorů dosahuje z pokročilých metod v rámci přístupu *Data association* nejlepších výsledků. Využívá lokální i globální informaci k určování trajektorie míčku z detekcí poskytnutých v předcházejícím kroku detekujícím míček na jednotlivých snímcích. Nejdříve jsou spojovány blízké detekce v určitém intervalu snímků, které tvoří krátké trajektorie a tyto jsou následně spojovány tak, aby vytvořili celou (globální) trajektorii míčku. Autoři volí tuto strategii konstruování kratších trajektorií za účelem omezení počtu falešných detekcí zahrnutých do trajektorie. Neboť dle jejich tvrzení čím delší trajektorie, tím větší šance zahrnutí falešné detekce. Je tedy upřednostněna konstrukce kratších trajektorií tvořených převážně pravými detekcemi, které jsou následně spojovány.

Přístup pomocí *Particle filtru* nebo přístup *Trajectory* s využitím *Kalmanova filtru* uvažují jen lokální informaci, kdy většinou z předchozí detekce hledají detekci následující, tedy je u nich oproti metodám uvažujících i globální informaci větší šance k odvedení *falešně pozitivními* detekcemi ze správné trajektorie. Ovšem lze je použít jako *online* metody a v jejich prospěch hraje také časová náročnost (u *Particle filtru* toto záleží na počtu vzorků). Přístup *trajectory* ve variantě hodnotící celou trajektorii vyžaduje kvalitní fyzikální model pohybu míčku, což je mimo můj obor vzdělání. Přístup *optimalizující účelovou funkci* vyžaduje formulaci *účelové funkce* zahrnující pohyb míče i hráčů pomocí níž predikuje polohu míče i při častém zakrytí hráči a častými výraznými změnami jeho pohybu. Tento přístup míří dle [4] spíše na míčové hry jako fotbal nebo basketbal, kde se hojně vyskytují zmíněné skutečnosti.

Návrh paralelizace

Z měření výpočetního času vyplynul jako nejnáročnější část implementace *Floyd-Warshallův algoritmus*, tento lze paralelizovat pomoci *GPU*. *Floyd-Warshallův algoritmus* je možné zapsat pomocí 3 vnořených cyklů. V [43] je navržen přístup ve kterém jsou dva vnitřní cykly prováděny na *GPU*.

Obrázek 4.4: Detekce míčku z *CNN TrackNet* jako červená kolečka a bílá čára detekovaná trajektorie míčku pomocí algoritmu *TLDA*. Původní obrázek pochází z [2].



4.3 Detekce hráčů

K detekci hráčů jsem využil *CNN YOLOv3* [44]. Tato má jako výstup *bounding boxy* okolo objektů ze zájmových tříd, která je v případě této práce pouze třída zahrnující člověka. Tuto *CNN* jsem vybral z důvodu její rychlosti, neboť projde snímek pouze jednou a zároveň doména na kterou bude použit není příliš složitá, tedy není důvod využít složitější, pomalejší, ale přesnější *CNN*.

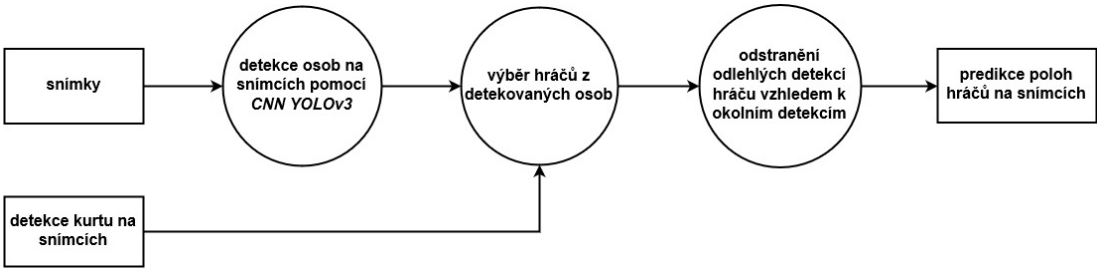
Ve scéně se často vyskytují další osoby kromě hráčů, zejména dochází k detekci čárových rozhodčí, je tedy třeba přistoupit k určení dvou hráčských osob. K tomuto je třeba detekce alespoň dvou osob a dále využívám detekci kurtu.

K určení osob hráčů jsem přistoupil hodnocením *bounding boxů* ohraničujících jednotlivé osoby ve scéně vzhledem k informaci o oblasti hřiště. Pokud je horní nebo dolní střed *bounding boxu* v oblasti detekovaného tenisového hřiště, je tento *bounding box* označen jako přináležející hráči. Pokud předchozím způsobem nejsou nalezeni dva hráči, poté jsou *bounding boxy* hodnoceny dle vzdálenosti k základním horizontálním čarám hřiště, kdy jsou jako hráči vybrány ty se vzdáleností nejmenší.

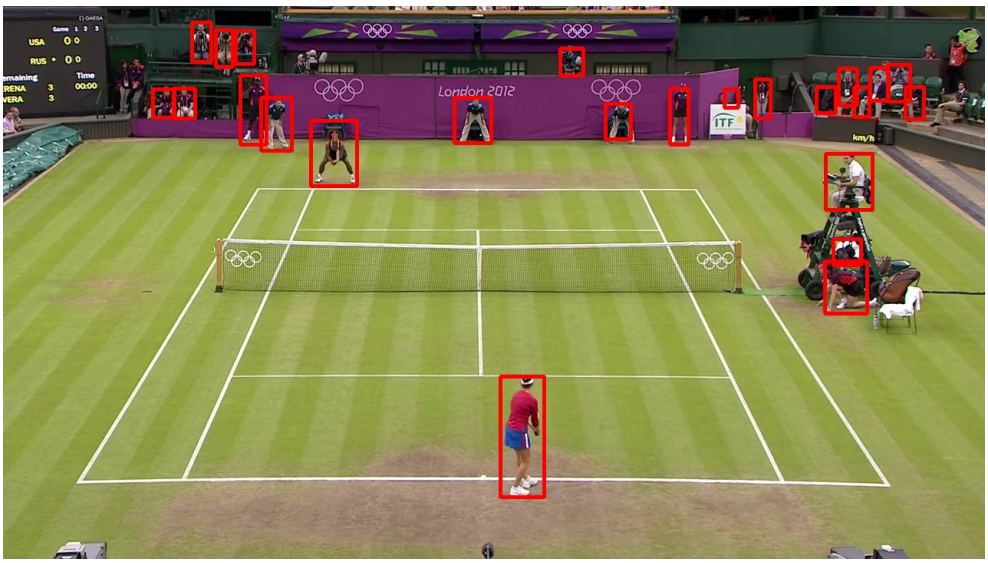
Využil jsem implementaci z [45] včetně již natrénovaných vah. Tato implementace mívá problém s detekcí hráčů, pokud jsou při úderech v nějaké nestandardní pozici.

Provádím korekci, pokud je místo hráče detekována jiná osoba tak, že je z časové řady souřadnic detekcí hráče vypočtena vzdálenost mezi sousedními dvěma detekcemi pro souřadnici x a y. Následně je ze spočtených vzdáleností vyjádřeno *normální rozdělení*. Pokud je nějaká vzdálenost více než 3 *směrodatné odchylky* od *střední hodnoty* je brána jako *odlehlé pozorováni (outlier)*. Pro detekovaná *odlehlá pozorování* je vypočten nový *bounding box*, jehož rozměry jsou stejné jako předcházející detekce a poloha je odvozena z určitého počtu okolních detekcí pomocí *lineární regrese*.

Obrázek 4.5: Diagram detekce hráčů.



Obrázek 4.6: Výstup *CNN YOLOv3* pro detekci osob. Původní obrázek pochází z [2].



Návrh paralelizace

Jak vyplývá z měření výpočetního času, viz 6.3, podílí se v této části systému na výpočetním času z drtivé části *CNN YOLOv3*, jejíž implementace běží

paralelně na *GPU*. Pro větší míru paralelizace lze navrhnout stejný přistup jako u *CNN TrackNet*. Toto není v této práci implementováno.

4.4 Detekce hřiště

Pro detekci tenisového hřiště jsem vyšel z přístupu uvedeném v [46], kde se k získání *masky* snímku využívá postup z [47] ve kterém se vychází z předpokladů, že čáry na tenisovém kurtu mají bílou barvu. To znamená, že v barevném formátu *HSV* mají vysokou hodnotu složky *value* reprezentující jas a nízkou hodnotu *saturation* reprezentující sytost barev vzhledem k ostatním částem scény. Pro snímek je spočten *histogram* pro složky *value* a *saturation*, ve kterých je nalezen nejvyšší bod (*mode*), následně probíhá proces, který by se dal přirovnat, s jistou dávkou představivosti, ke *"slézání kopce rovnoměrně z obou stran"*, který má vrchol v nalezeném nejvyšším bodě *histogramu*, kdy jeden krok ve "slézání"představuje jedno políčko histogramu. Výstupem procesu jsou dvě pozice reprezentující výsledek *"slézání kopce"*. Z těchto dvou pozic je určena *prahová hodnota* pro operaci *prahování (threshold)*, kdy pro složku *value* je prahová hodnota vyšší pozice a pro složku *saturation* nižší pozice. Tímto postupem jsou získány dvě *masky*, které jsou zkombinovány logickou operací *AND*.

V práci [46] následuje filtrace získané masky, kdy jsou odstraněny *kontury* mající velkou plochu a malou délku, neboť u čar kurtu se předpokládá opačná vlastnost. Dále je použita *pravděpodobnostní Houghova transformace* na hledání úseček v *masce*. Tato metoda kvůli nedokonalostem zobrazení reálného světa na snímku produkuje pro jednu tenisovou lajnu více úseček, které jsou následně spojovány dle zvolených kritérií. Ve své práci jsem zvolil jako kritéria úhel a překryv, případně blízkost koncových bodů úseček. Výsledné úsečky jsou následně rozděleny na vertikální a horizontální a seřazeny dle jejich souřadnic. Jejich konce jsou vzaty jako význačné body kurtu.

Ve své implementace jsem s v určitých částech od postupu v [46] odchýlil. V postupu [46] je předpokládáno, že pospojovaný výstup z *pravděpodobnostní Houghovy transformace* bude obsahovat pouze úsečky reprezentující čáry kurtu. Toto omezení obcházím rozdělením úseček do skupin, které tvoří souvislý segment. Segment, který splňuje kritéria je vybrán jako tenisový kurt, pokud je segmentů více, poté je vybrán největší. Kritéria ošetřují počet vertikálních a horizontálních úseček v segmentu. Vertikálních úseček je očekáváno 5, nicméně vzhledem k občas obtížně detekovatelné prostřední vertikální čáry na tenisovém kurtu, jsou akceptovány i 4. Horizontálních čar je očekáváno 5, včetně pásky ohraničující horní část sítě, pokud je sít významněji prověšena, je rozdělena na dvě úsečky a tedy je akceptováno i 6 horizontálních čar. Dále je zkontrolováno jestli úsečka reprezentující základní hrací čáru na horní části kurtu odpovídá umístěním koncům vertikálních čar, aby při detekci 5 hori-

zontálních úseček nereprezentovaly dvě pásku sítě. Detekované horizontální čáry se mezi sebou nesmějí protínat, totéž platí pro vertikální. Při detekci 4 vertikálních čar je zkontrolováno, že všechny vertikální čáry jsou blíž koncům horizontálních čar než jejich prostředku, aby při detekci těchto 4 vertikálních čar nebyla jedna z nich prostřední vertikální čára.

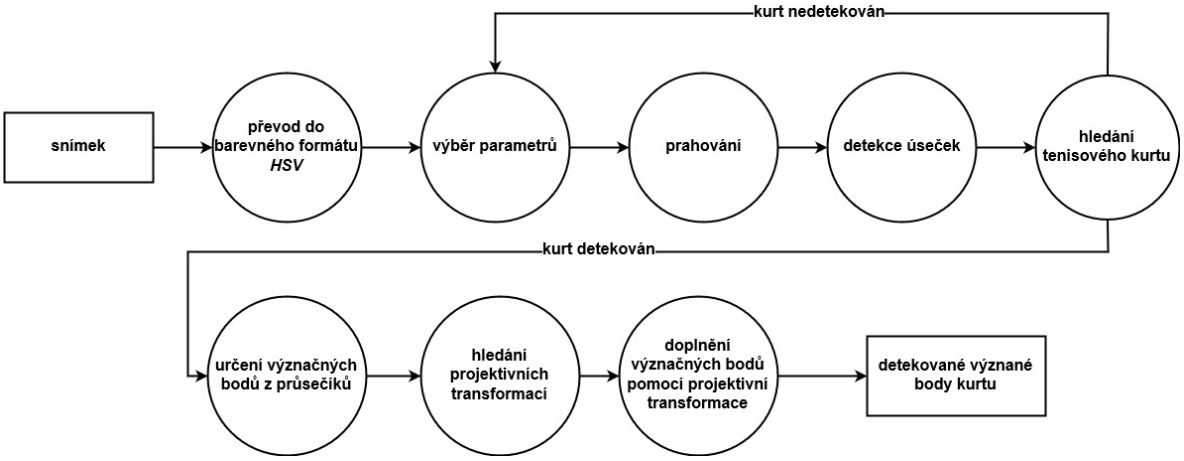
Další odlišností v mém postupu je, že pokud je detekován tenisový kurt jsou vyjádřeny význačné body kurtu z průsečíků úseček. Díky tomuto není nutné detekovat čáry kurtu celé, ale jen jejich část, případně nemusí být ve scéně celý tenisový kurt, ale jen dostatečně velká část každé čáry kurtu. Ovšem každá úsečka se musí dotýkat, nebo být dostatečně blízko, jiné, aby byla zařazena do jednoho souvislého segmentu.

Z význačných bodů také získávám *projektivní transformaci* mezi tenisovým kurtem ve scéně a umělým modelem kurtu a naopak. Pomocí *projektivní transformace* jsou následně dopočteny význačné body, které není možné získat průsečíky nalezených úseček, případně koncové body středové vertikální čáry kurtu, pokud není detekována. Pro přehled detekovaných význačných bodů kurtu viz obrázek 4.8.

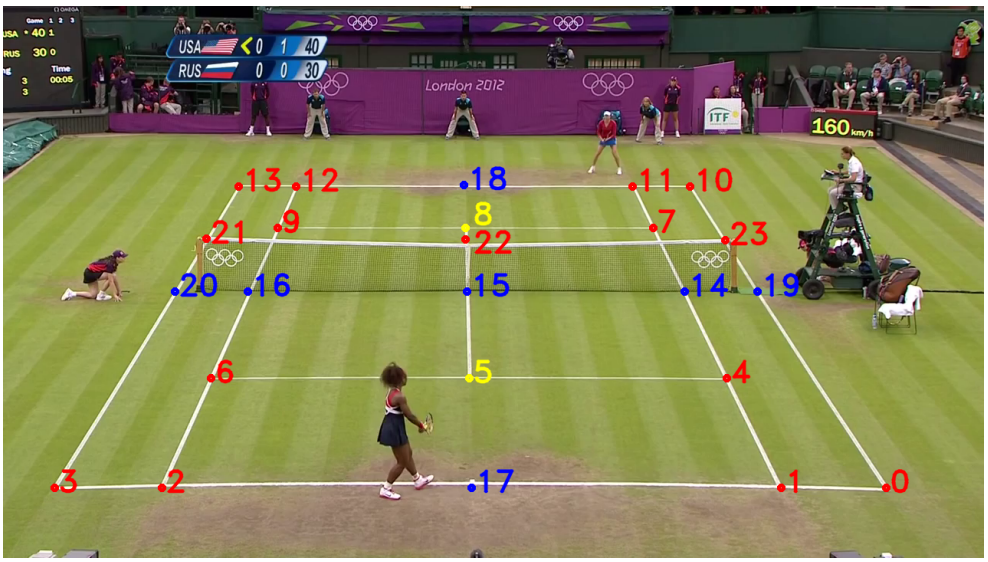
Pokud není na snímků detekován tenisový kurt jsou upravovány parametry algoritmu a to počet kroků v algoritmu hledajícím *prahy* v histogramech a minimální délka úsečky v *pravděpodobnostní Houghově transformaci*. Ke snížení výpočetního času je úspěšná kombinace vždy přesunuta na první místo v seznamu zkoušených kombinací. Pro další snímek je tedy pravděpodobné, že bude úspěšná hned první kombinace, neboť sousední snímku se od sebe příliš neliší. Pokud po vyčerpání všech kombinací nastavení není kurt nalezen je snímek označen jako snímek neobsahující tenisový kurt a dokud není nalezen snímek dostatečně se lišící od snímku bez tenisového kurtu, jsou snímky bez procesu hledání označeny jako snímky bez tenisového kurtu, neboť procházení všech kombinací nastavení je časově náročné. K měření podobnosti využívám míru *korelace histogramů*.

Dalším krokem urychlujícím výpočet je omezení počtu úseček z výstupu *pravděpodobnostní Houghovy transformace*. Maximální počet úseček je vypočten tak, že se předpokládá délka čar kurtu přes celý snímek a délka čar je vydělena minimální velikostí úsečky z *pravděpodobnostní Houghovy transformace*, tento počet je ještě navýšen, aby mohl být případně detekován i určitý počet úseček mimo čáry kurtu. Díky tomuto omezení se přeskočí kombinace hodnot nastavení algoritmu u kterých je detekován velký počet úseček v nichž by hledání tenisového kurtu bylo výpočetně náročné, nebo snímky na kterých tenisový kurt vůbec není, kdy je předpokládáno, že nějaká z následujících kombinací nastavení odstraní více úseček nepocházejících z čar kurtu.

Obrázek 4.7: Diagram detekce tenisového kurtu.



Obrázek 4.8: Výsledek detekce tenisového kurtu. Červeně body získané průsečíky, modře body získané *projektivní transformací*, žlutě body získané jedním z předešlých dvou způsobů. Původní obrázek pochází z [2].



Návrh paralelizace

Vzhledem k výpočetní náročnosti této části systému, viz tabulku 6.7, se nabízí možnost paralelizace, která zmenší výpočetní čas. Vzhledem ke složitosti složitosti programu je tato část vhodná k paralelizace na *CPU*. Tuto lze provést rovnoměrným rozdělením zpracovávaných snímků mezi jádra procesoru. Rozdělení by mělo pro jednotlivá jádra přiřadit po sobě jdoucí bloky snímků, aby byla

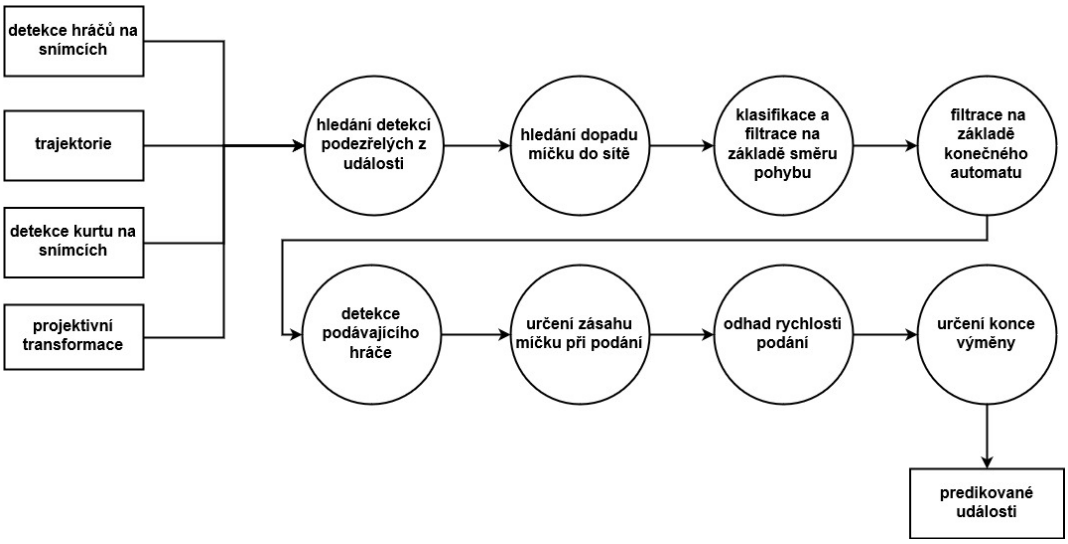
zachována informace o vhodném nastavení parametrů, která bývá společná pro blízké snímky.

4.5 Detekce událostí

Vzhledem k nedostupnosti dostatečně obsáhlé datové sady obsahující tenisovými událostmi anotovanou trajektorii jsem se rozhodl k problému přistoupit bez *strojového učení* pouze na základě *expertních pravidel*. Na obrázku 4.9 je vyobrazen diagram popisující postup *detekce událostí*.

V tomto kroku systému je využívána informace o trajektorii míčku, význačných bodech tenisového kurtu a umístění hráčů. Pro snímky s chybějící detekcí míčku je jeho pozice *lineárně interpolována* z nejbližších dvou okolních známých detekcí. Pro snímky s chybějící detekcí hráčů nebo kurtu jsou detekce míčku z trajektorie vymazány.

Obrázek 4.9: Diagram detekce *událostí*.



Detekované události jsou rozděleny dle toho, zda se stali na dolní části kurtu, respektive je učinil hráč na dolní části kurtu, nebo se stali na horní části kurtu, respektive je učinil hráč na horní části kurtu (je předpokládán standardní televizní záznam, kdy je kamera umístěna za horní nebo dolní základní čarou kurtu, viz obrázek 2.2). Jsou rozlišovány údery vlevo a vpravo od příslušného hráče, dopad v a vně kurtu, servis, dopad po servisu v a vně určené oblasti kurtu a dopad míčku do sítě rozdělený dle hráče, který ho do sítě odrazil. U servisu je odhadována rychlost.

Události na vzdálené části kurtu:

- land_far_in, LFI
- land_far_out, LFO
- hit_far_left, HFL
- hit_far_right, HFR
- serve_far_left, SFL
- serve_far_right, SFR
- land_net_far, LNF
- serve_land_far_in, SLFI
- serve_land_far_out, SLFO

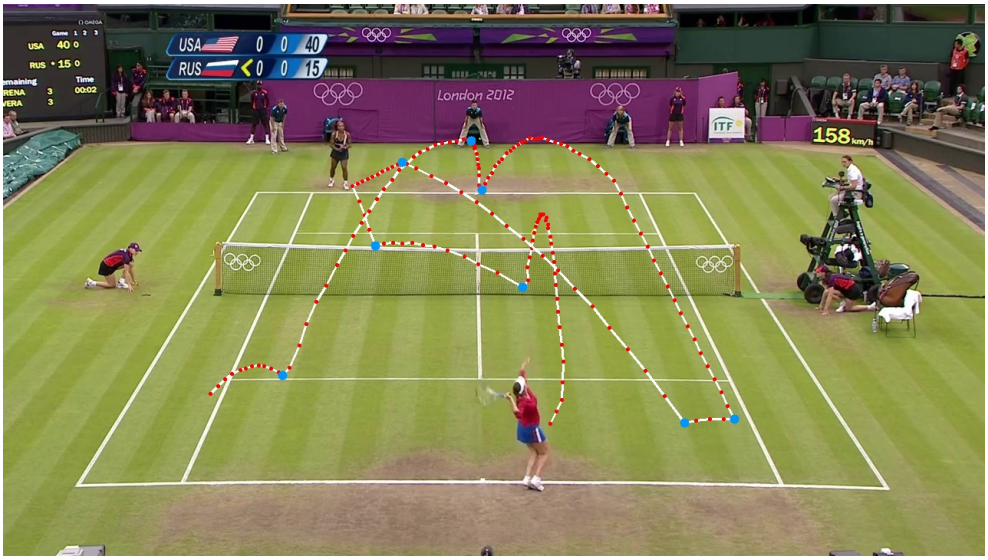
Události na bližší části kurtu:

- land_near_in, LNI
- land_near_out, LNO
- hit_near_left, HNL
- hit_near_right, HNR
- serve_near_left, SNL
- serve_near_right, SNR
- land_net_near, LNN
- serve_land_near_in, SLNI
- serve_land_near_out, SLNO

Další události:

- odhad rychlosti podání v km/h

Obrázek 4.10: Trajektorie s detekovanými *událostmi* světle modře. Původní obrázek pochází z [2].



Detekce podezřelé z události

Výše představené události se vyznačují tím, jak bylo uvedeno v sekci 3.5, že v jejich důsledku mění míček směr, rychlost a akceleraci. Tato podmínka je nutná, ale ne dostačující, jako příklad lze uvést kulminaci při stoupání míčku, kdy dochází ke změnám zmíněných veličin, ale není to událost, která je určena k detekci. Pro detekci dopadů a úderů, s výjimkou podání, je nejméně diskriminující a dostačující vlastností úhel mezi směrovými vektory před a po události, tuto jsem tedy využil.

Pro každý bod trajektorie je vypočten úhel dvou vektorů, kdy první míří z předcházející detekce do aktuálně zkoumané a druhý z aktuálně zkoumané do následující. Je testována vzdálenost okolních detekcí o 1 a o 2 pozice v trajektorii, neboť při vzdálenosti o 1 se úhel v určitých případech, zejména při dopadu s pozvolným klesáním a stoupáním, ještě nemusí projevit. Pokud některý úhel překročí stanovenou *horní prahovou hodnotu* a zároveň jsou všechny testované úhly větší než *spodní prahová hodnota*, je detekce určena jako *podezřelá z události*. V důsledku toho, že je volena i vzdálenost o 2 pozice, bude často úhel detekován i na pozicích vzdálených o 1 od pozice, kde došlo k události. Toto je filtrováno pomocí předpokladu, že pozice na které došlo k události, bude mít minimální úhel z testovaných možností větší nežli okolní dvě detekce vzdálené o jednu pozici.

Řešení šumu v trajektorii

Bohužel výstup z kroku detekujícím tenisový míček na jednotlivých snímcích pomocí *CNN TrackNet* nevrací vždy konzistentně stejnou pozici vzhledem ke *kontuře* míčku, tedy je přítomen šum a v trajektorii může vzniknout jakýsi *"zub"*, který může vyvolat *falešnou detekci události*. K filtraci jsou spočteny dva okolní úhly s detekcemi na pozicích $[i-2,\ i-1,\ i+1]$ a $[i-1,\ i+1,\ i+2]$, kdy i je pozice aktuálně zkoumané pozice, úhly tedy vynechávají aktuální detekci. Neboť při dopadu míčku mohou být pozice $i-1$ a $i+1$, tedy před a po dopadu, jakoby ve stejné výšce, čímž vznikne malý úhel, díky němuž by byla korektní detekce *události* dopadu vyřazena, jsou spočteny ještě úhly pro detekce na pozicích $[i-2,\ i-1,\ i+2]$ a $[i-2,\ i+1,\ i+2]$. Pokud tyto kontrolní úhly nepřekročí stanovenou *prahovou hodnotu*, která je menší než *horní prahová hodnota* pro předchozí krok, poté není bod brán jako *podezřelý z události*.

Tímto způsobem se mi bohužel nepodařilo odstranit *falešné detekce událostí* při nadhozu při podání, neboť trajektorie se při něm silně stáčí a tedy kontrolní úhly překročí stanovenou mez. Toto je zejména problém při podání hráče na dolní straně kurtu, pokud dojde během nadhozu míčku při podání k *falešné detekci* události *dopad na horní straně kurtu* a detekce úderu podání se následně hledá od špatné pozice dopadu, viz níže část textu o *detekci události podání*. Pro řešení tohoto problému jsem zvolil dvojitý průchod trajektorie, kdy se nejdříve prochází trajektorie filtrovaná *průměrovacím filtrem* o malé velikosti. Tato trajektorie je sice zbavena zmíněného šumu, nicméně detekce *událostí* jsou často posunuty o pár pozic oproti své pravé poloze. Tento první průchod využívám pouze pro určení přibližného dopadu míčku po podání. Druhý průchod je již aplikován na původní trajektorii s tím, že dopad míčku po podání je hledán vzhledem k dopadu nejblíže dopadu míčku po podání detekovanému v prvním průchodu. Pokud v prvním průchodu není nalezen dopad míčku po podání v důsledku přílišného *vyhlazení* trajektorie *průměrovacím filtrem*, neboť se úhel sníží pod *prahovou* hodnotu, je zmíněný dopad hledán v původní trajektorii v druhém průchodu.

Hledání dopadu míčku do sítě

Dále jsou zkoumány *detekce podezřelé z události*, zda se nejedná o *dopad míčku do sítě*. Tato událost musí splňovat umístění v oblasti sítě a detekce mezi touto a předchozí událostí a všechny detekce po této události musí být v odpovídajících oblastech tenisového kurtu. Pokud je tato událost detekována, je příslušně klasifikována a *detekce podezřelé z události* za touto událostí jsou vyřazeny z dalšího zkoumání a předešlé události, které se nemohou před dopadem do sítě vyskytovat, jsou odstraněny.

Klasifikace a filtrace na základě směru pohybu

Lze předběžně roztřídit události dle pohybu před a po jejím výskytu vzhledem k souřadnici y vůči předcházející a následující události, kdy souřadnice y je chápána jako vertikální směr vzhledem ke scéně a je rostoucí směrem od shora dolů. Pokud se jedná o událost dopadu míčku ve většině případů je před i po dopadu směr vzhledem k souřadnici y stejný, naopak u události úderu dochází většinou ke změně směru. Dle těchto kritérií jsou detekce rozděleny na kategorie *dopad* a *úder*, které ale nejsou definitivně dané. Událost *úder* je přičleněna k hráči na dolní nebo horní části hřiště dle blízkosti k detekcím hráčů a dle toho je nazvána *úder horního hráče* nebo *úder dolního hráče*.

U dopadu ani úderu nemusí předpoklad vždy platit, pokud se jedná o dopad míčku na spodní straně kurtu směřující od hráče na vzdálenější straně, kdy míček po dopadu výrazně stoupá, nebo trajektorie míčku po dopadu na spodní straně kurtu a před následným úderem může směřovat přibližně horizontálně doprava nebo doleva a míček je zasažen hráčem aniž by se dostal pod polohu dopadu v souřadnici y, poté bude před dopadem souřadnice y stoupající a po dopadu klesající, tedy bude mít stejné směrové charakteristiky jako událost *úder* a následující úder jako událost *dopad*.

Je prováděna filtrace celé sekvence událostí pomocí tří testů uvedených níže. Filtrace celé sekvence se opakuje dokud v předchozí nebyla odstraněna žádná událost. Testy nesmí být příliš přísně, neboť se mezi událostmi vyskytují i falešné detekce, které narušují předpoklady o chování daných událostí, nicméně lze odstranit nebo překlasifikovat událostí, které porušují pravidla uvedená v testech.

1. *Test dopadu*: Pokud jsou spojeny úsečkou detekce krátce před a za dopadem, úsečka je prodloužena pokud jsou detekce příliš u sebe ve směru souřadnice x, dojde k protnutí s polopřímkou vedenou od pozice dopadu se směrovým vektorem $(0, -1)$, tedy směrem kolmo nahoru, s případnými malými rotacemi.

2. *Test úderu dolním hráčem*: Detekce míčku musí byt blíže hráči na dolní straně hřiště a směry v souřadnici y následující:

   a) Před i po události klesající.

   b) Před událostí stoupající a po události klesající.

   c) Před událostí klesající a po události stoupající. Tuto podmínku splňují i detekce ve vrcholu *obloučku* po dopadu na dolní straně hřiště, viz obrázek 4.10 a trajektorii po dopadu na dolní straně hřiště vlevo. Proto je přidána podmínka, že mezi touto a další událostí se musí nacházet detekce na horní půlce hřiště.

3. *Test úderu horním hráčem*: Detekce míčku musí byt blíže hráči na horní straně hřiště, umístění detekce musí být na horní straně hřiště vzhledem k polovině hřiště a směry v souřadnici y následující:

   a) Před i po události stoupající.

   b) Před událostí klesající a po události stoupající.

Ve zmíněné filtraci jsou odstraňovány události *dopad*, pokud nesplňují *test dopadu* ani se je nepovede překlasifikovat na jeden z úderů pomocí uvedených testů pro údery. Jsou odstraňovány události *úder* pokud nesplňují příslušný *test úderu* ani se je nepovede překlasifikovat na dopad pomocí *testu dopadu*.

Pokud je událost odstraněna jsou sousední události znovu klasifikovány, neboť se jejich směrové charakteristiky vůči souřadnici y mohli po odstranění události změnit.

Události jsou během tohoto kroku klasifikovány do 4 kategorií *úder horního hráče*, *úder dolního hráče*, *dopad na dolní části hřiště* a *dopad na horní části hřiště* pomocí informací o postavení hráčů a o význačných bodech detekovaného kurtu.

Filtrace na základě konečného automatu

V tenisové výměně může dojít po určité události jen k omezenému množství událostí následujících. Tyto pravidla lze přenést do *konečného automatu*, který při sekvenci událostí mimo pravidla zahlásí *chybový stav*. *Chybový stav* znamená, že je nějaká událost detekována navíc, nebo je klasifikována špatně, nebo je určena k dodatečné kontrole. Na základě posledního korektního *stavu* a *přechodu konečného automatu*, který způsobil *chybový stav* je přistoupeno ke korekci nebo odstranění události nebo kontrole události. Události lze zpracovávat směrem od první k poslední nebo naopak od poslední k první. Vybral jsem druhou variantu, neboť na začátku trajektorie se ve větší míře nežli na konci trajektorie mohou vyskytovat falešné detekce *událostí* díky nadhozu hráče při podání, které mohou mít negativní efekt na tuto filtraci. Konečné automaty pro oba směry jsou uvedeny na obrázku 4.11 a 4.12. V automatech není přechod umožnující po *úderu dolního hráče dopad na dolní straně hřiště* a po *úderu horního hráče dopad na horní straně hřiště*, neboť tato událost je ještě kontrolována, protože její výskyt znamená konec výměny, tedy se vyskytuje pouze jednou za celou tenisovou výměnu, což znamená, že její výskyt je málo častý a tudíž je šance, že se jedná o falešnou detekci.

Nyní budou uvedeny sekvence událostí, které jsou kontrolovány nebo v určitých případech opraveny:

- Sekvence
  1. *úder dolního hráče - úder dolního hráče*,
  2. *úder horního hráče - úder horního hráče*.

  V těchto sekvencích nastane chybový stav kvůli dvěma úderům od stejného hráče za sebou. Dřívější událost je otestována *testem dopadu* a pokud test dopadne kladně, je událost změněna na *dopad* na odpovídající straně hřiště a tato sekvence se stane korektní. Pokud neprojde *test dopadu* je to samé opakováno pro pozdější událost. Pokud ani jeden z testů nedopadne kladně, poté je z těchto dvou událostí ponechána bližší k příslušnému hráči a druhá je odstraněna. Pokud se ovšem před prvním *úderem dolního hráče* vyskytuje další *úder dolního hráče* je nejdříve testována pozdější událost a následně dřívější.

- Sekvence
  1. *úder dolního hráče - dopad na dolní straně hřiště - dopad/úder na horní straně hřiště*,
  2. *úder horního hráče - dopad na horní straně hřiště - dopad/úder na dolní straně hřiště*,

  Tyto sekvence by byly korektní, pokud by pořadí prvních dvou bylo obrácené. Je proto proveden *test dopadu* první události *úder* a pokud test projde, jsou první dvě události prohozeny.

- Sekvence
  1. *dopad na dolní straně hřiště - úder horního hráče*,
  2. *dopad na horní straně hřiště - úder dolního hráče*.

  Tyto sekvence by byly korektní, pokud by jejich pořadí bylo obrácené. Je proto proveden *test dopadu* události *úder* a pokud test projde, jsou události prohozeny.

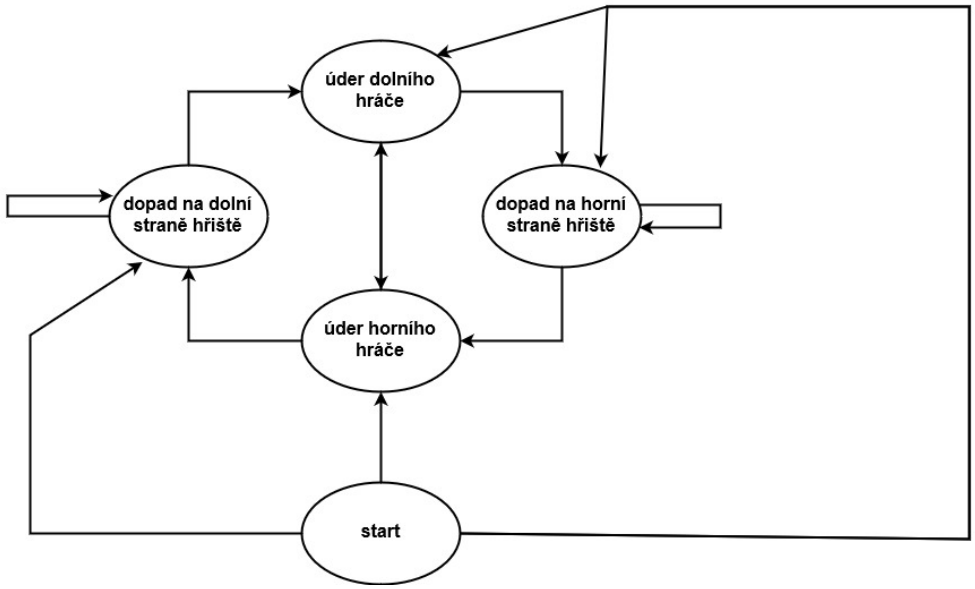
- Sekvence
  1. *úder dolního hráče - dopad na dolní straně hřiště*,
  2. *úder horního hráče - dopad na horní straně hřiště*.

  Tyto sekvence jsou korektní, neboť hráč odehraje míček a ten nepřejde přes síť a dopadne na stranu hřiště odehrávajícího hráče. Nicméně jak bylo uvedeno výše, je zkontrolováno, zda tato sekvence splňuje očekávané

předpoklady. Za událostmi v sekvenci *úder dolního hráče - dopad na dolní straně hřiště* se nevyskytuje *úder horního hráče* nebo *dopad na horní straně hřiště* a naopak pro druhou sekvenci. Pokud ano, je smazán dopad. Pokud ne a za úderem v kontrolované sekvenci se vyskytuje úder stejného hráče, je smazán vzdálenější úder od příslušného hráče. Jestliže nebyl smazán úder v kontrolované sekvenci, je zkontrolováno, jestli míček před a po dopadu směřuje od hráče, pokud ano jsou události ponechány, pokud ne, je smazán úder.

Pokud chybový stav nespadá do výše uvedených kategorií je odstraněna událost, která chybový stav způsobila.

Obrázek 4.11: Konečný automat pro filtraci *událostí* tenisové výměny v dopředném směru. Přechodové hrany nastávají pro *události* odpovídající cílovému stavu přechodu.

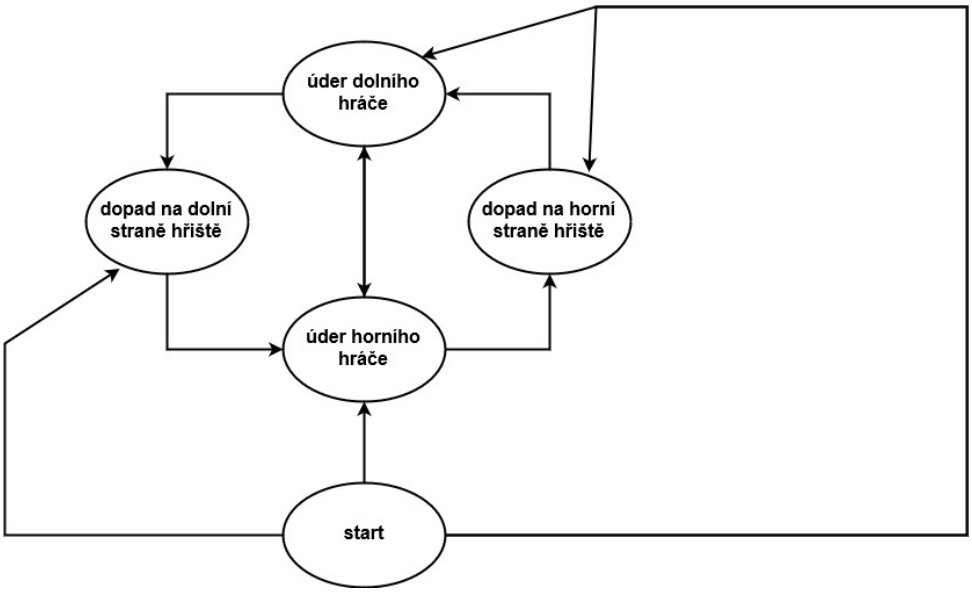


Určení podávajícího hráče a dopadu míčku po podání

Pokud je detekována událost *dopad do sítě* v sekvenci událostí není dvojice *dopad na horní straně hřiště - úder horního hráče* nebo *dopad na dolni straně hřiště - úder dolního hráče*, tedy dvojice nutná pro dopad servisu a následný úder hráče, je podání vyhodnoceno jako podání do sítě.

Jinak je nalezena první událost *dopad*, která se nachází na jedné ze stran kurtu a dle dané strany je určen podávající hráč pokud je splněno následující:

Obrázek 4.12: Konečný automat pro filtraci *událostí* tenisové výměny ve zpětném směru. Přechodové hrany nastávají pro *události* odpovídající cílovému stavu přechodu.



- V případě *podání hráče na horní straně hřiště* je ověřeno, zda se některá z detekcí předcházející události *dopad* nachází nad detekovanou páskou sítě a také zda se *dopad* nebo některá předcházející detekce vyskytuje na dolní polovině hřiště. První detekce míčku v trajektorii musí být blíže horní základní horizontální čáře než pásce sítě.
- V případě *podání hráče na dolní straně hřiště* je ověřeno, zda se některá z detekcí předcházející události *dopad* nachází nad detekovanou páskou sítě a zda první detekce míčku v trajektorii je blíže pásce sítě než horní základní horizontální čáře.

Pokud není podání nalezeno pomocí události *dopad*, je zkontrolováno, zda byl míček na dolní polovině hřiště během celé trajektorie, pokud ne, je podání klasifikováno jako podání do sítě hráčem na horní straně hřiště.

Následně je dle pozice detekce hráče vůči bodu v polovině základní horizontální hrací čáry určeno, zda hráč podává zleva či zprava.

Detekce události podání

Podání je speciální úder, který nemusí být nalezen jako *detekce podezřelá z události* pomocí změny směru, neboť směr pohybu před a po zásahu míčku může být podobný. Od nalezeného dopadu po podání je v čase zpět hledána událost zásahu míčku při podání. Detekce v čase zpět od dopadu po podání jsou hodnoceny dle následujících kritérií.

Pro podání hráče na horní/dolní straně hřiště se hodnotí:

1. změna rychlosti v kladném/záporném směru v souřadnici y vyjádřena podílem velikosti tohoto pohybu mezi předchozí a aktuální detekcí a mezi aktuální a následující detekcí, jednotlivé velikosti pohybu jsou děleny počtem snímků mezi detekcemi,
2. rychlost v kladném/záporném směru souřadnice y od aktuální detekce k následující dělená počtem snímků mezi detekcemi,
3. úhel mezi vektorem z předchozí detekce do aktuální a vektorem z aktuální do následující detekce,
4. vzdálenost aktuální detekce od podávajícího hráče.

Podání hráče na horní části hřiště: První kritérium stojí na předpokladu, že po úderu dojde ke zrychlení míčku žádoucím směrem, ale toto kritérium splňuje i padající míček po kulminaci při nadhozu během podání, je proto přidáno druhé kritérium na rychlost v žádoucím směru. Bohužel pří podání hráče na horní části hřiště se díky perspektivnímu zkreslení zvětšuje rychlost čím je míček blíže snímající kameře, tedy detekce blízko dopadu budou mít vysoké ohodnocení vzhledem k prvnímu a druhému kritériu, aby se tento jev potlačil bylo pro podání hráče na horní části hřiště přidáno čtvrté kritérium, které penalizuje detekce vzdalující se od hráče. Třetí kritérium hodnotí změnu úhlu, což je častý případ při úderu do míčku a také okolo kulminace při nadhozu.

Podání hráče na dolní části hřiště: Zde je uplatněno pouze první a třetí kritérium, neboť důvod pro přidání kritéria vzdálenosti od hráče odpadl a druhé kritérium upřednostňuje pozice míčku při nadhozu po vypuštění míčku z ruky, neboť se často pohybuje skoro přímo ve směru osy y v žádoucím směru na rozdíl od směru po úderu, který svírá s osou y podstatně větší úhel a tedy pohyb v žádoucím směru y může být menší.

Detekce jsou následně seřazeny zvlášť dle každého využitého kritéria tak, že více žádoucí skóre je v seřazení vpředu. Ke skóre každé detekce je přičteno umístění v seřazení dle daných kritéria, kdy z kritérií 1 a 2 je u podání hráče

na horní části hřiště přičteno pouze horší umístění, neboť detekce míčku při nadhozu před úderem při podání by měli být ohodnoceny dobře jen vzhledem k jednomu z těchto kritérií. Následně je určena detekce s minimálním skóre jako událost podání.

Závěrečná filtrace

V tomto kroku jsou mazány *události*, které jsou pravděpodobně *falešně pozitivní*, tedy jsou detekovány navíc a budou odstraněny. Pravidla pro tuto filtraci jsou následující:

1. Před posledním úderem je mezi všemi údery dovolen pouze jeden dopad. Pokud je mezi dvěma údery dopadů více, je ponechán pouze první z hlediska časové roviny. Je ponechán první dopad, neboť *falešně pozitivní* detekce dopadu se často vyskytují díky šumu v blízkosti hráče, dále je trajektorie po úderu a před dopadem složena z méně detekcí nežli trajektorie po dopadu a před úderem, neboť míček letí rychleji, je tedy menší pravděpodobnost přítomnosti šumu.
2. Neboť *filtrace na základě konečného automatu* hodnotí sekvenci událostí pouze lokálně, může být výsledkem například *dopad na horní straně hřiště - úder horního hráče- dopad na horní straně hřiště - úder horního hráče*, proto je zkontrolováno, že údery hráčů se střídají. Pokud jsou nalezeny dva údery stejného hráče mezi nimiž není úder hráče druhého, je odstraněn ten dále od příslušného hráče.
3. Údery po nichž se míček nevyskytuje v oblasti dané prodloužením základních čar přes celý snímek jsou odstraněny.
4. Před dopadem do sítě se může nacházet jen dopad na příslušné straně hřiště nebo úder od příslušného hráče, dokud není takový nalezen, jsou události před dopadem do sítě mazány.

Odhad rychlosti podání

U detekcí dochází k absenci skutečných souřadnic míčku vzhledem k reálnému světu kromě speciálních případů, kdy je tato informace odvozena vzhledem k určité události. Toto platí pro kontakt míčku se zemí, kdy pozice míčku určuje dvě souřadnice a třetí je odvozena z doménové znalosti, tedy že souřadnice z je nulová. S menší přesností lze odvodit souřadnici z při úderu a to z pozice hráče a předpokládané výšky ve které dochází k úderu. Při podání tuto výšku ovlivňují fyzické dispozice hráče, velikost rakety a výška výskoku. Pro zjednodušení je tato výška stanovena na 250 cm.

Ze snímku na kterém dochází k událostí úderu míčku při podání jsou vzaty souřadnice z poloviny dolní hrany *bounding boxu* ohraničujícího hráče, tím

dostáváme přibližnou polohu, kde hráč stojí, tímto jsou určené souřadnice x, y a souřadnice z je stanovena na hodnotu 250, jak bylo uvedeno výše. Z určeného dopadu podání jsou odvozeny souřadnice x, y a souřadnice z je stanovena na hodnotu 0. Dále jsou souřadnice x, y pomocí *projektivní transformace*, získané během detekce tenisového kurtu, převedeny na souřadnice reálného světa. Mezi těmito dvěma body, již reflektujícími reálný svět, je spočtena vzdálenost. Získaná vzdálenost je vydělena časem mezi dvěma zmíněnými událostmi, kdy čas je získán podílem počtu snímků mezi událostmi a *počtem snímků za vteřinu* videa.

Určení konce tenisové výměny

Dle posloupnosti událostí je možné určit typ konce tenisové výměny. Tyto lze shrnout do následujících pravidel:

1. Dopad míčku v korektní části hřiště následovanými druhým dopadem, kdy předcházejícím úderem je/není podání.

2. Dopad mimo korektní část hřiště, tedy *out*, kdy předcházejícím úderem je/není podání.

3. Poslední událostí je úder hráče bez následujícího detekovaného dopadu míčku.

4. Poslední událostí je dopad míčku do korektní části hřiště, kdy předcházejícím úderem je/není podání.

5. Dopad míčku do sítě, kdy předcházejícím úderem je/není podání.

Po určení konce tenisové výměny je možné následující události smazat, neboť nejsou pro analýzu tenisové výměny podstatné.

Určení dopadu v a vně kurtu

K určení zda se dopad míčku nachází v korektní oblasti kurtu je potřeba porovnat jeho polohu vůči polohám čar vyznačujících tenisový kurt. S omezenou přesností lze využít získanou *projektivní transformaci* ze snímku do modelu kurtu v němž lze již snadno určit, zda transformovaný bod leží v či vně určené oblasti kurtu.

Pro větší přesnost by bylo vhodné při blízkosti dopadu míčku k některé z čar přistoupit k metodě analyzující konturu míčku a čáry na úrovni *pixelů*. Toto není v práci implementováno.

Určení umístění detekce v zájmové oblasti

V příslušných částech textu bylo zmíněno testování, zda detekce leží v oblasti kurtu nad/pod páskou sítě. Vzhledem ke svěšení sítě je síť určena třemi body, respektive dvěma konci sítě a bodem průniku dvou úseček, kterými páska sítě byla určena. Pokud přidáme příslušně dvě vertikální čáry a jednu horizontální čáru kurtu, poté se jedná pro horní polovinu kurtu o *konvexní polygon* a pro dolní polovinu o *nekonvexní polygon*. Při rozdělení oblasti na dvě podoblasti dělených prostředním bodem sítě a polovinou horizontální čáry jsou tyto obě podoblasti *konvexními polygony*. Poté lze přítomnost bodu v a vně určit pomocí testování polohy bodu vůči jednotlivým úsečkám tvořících *polygon*, respektive bod musí ležet vůči všem úsečkám na stejné straně vzhledem k pohledu z prvního bodu úsečky, pokud jsou iterovány dvojice po sobě jdoucích bodů *polygonu* po/proti směru hodinových ručiček, postup lze nalézt v [48].

Kapitola 5

Implementace

5.1 Základní struktura implementace

Implementace jednotlivých částí návrhu je rozdělena do separátních zdrojových souborů, kdy pro žádoucí funkcionalitu slouží vždy volání jedné funkce s příslušnými parametry, kdy tato funkce vrací příslušné výstupy jako návratové hodnoty, bude popsáno níže. Pro závislosti mezi jednotlivými částmi viz obrázek 4.1. Jednotlivé části systému jsou spouštěny vůči sobě sekvenčně, tedy poté co ukončí výpočet daná část systému je spuštěna další, pořadí spouštění, kde neuvádím vstup a výstup souborového systému, je následující:

1. detekce kurtu
2. detekce hráčů
3. detekce tenisového míčku
4. hledání trajektorie
5. detekce událostí
6. vizualizace

Implementace popsaného návrhu řešení byla provedena v jazyku *Python*.

Použité modely neuronových sítí využívají knihovnu *Keras*.

K operacím s obrazem byla využita knihovna *OpenCV*.

K paralelizaci na *CPU* byla využita knihovna *multiprocessing* a k paralelizaci na *GPU* knihovna *PyCUDA*.

Je použita implementace *CNN TrackNet* z [41].

Je použita implementace *CNN YOLOv3* z [45].

Je použita implementace *Floyd-Warshalova algoritmu* v jazyce *Python* z [49].

Je použita implementace datové struktury *disjoint set* z [50].

Je použita implementace vzdálenosti bodu a úsečky z [51].

5.2 Popis zdrojových souborů

Zde budou popsány zdrojové soubory z hlediska poskytované funkcionality a základního rozhraní pro její plnění.

Zdrojový soubor *tracknet.py*

V tomto zdrojovém souboru je využita *CNN TrackNet*. Pro funkcionalitu slouží zavolání dvou funkcí:

- Název funkce: *get_tracknet_model*
  - Vstup: souborová cesta k vahám *CNN TrackNet*
  - Výstup: zkompilovaný model
- Název funkce: *get_detections_in_images*
  - Vstup: snímky v BGR formátu a zkompilovaný model
  - Výstup: detekce tenisového míčku na jednotlivých snímcích

Zdrojový soubor *TLDA.py*

Ve zdrojovém souboru je implementován algoritmus *TLDA*. Pro volbu verze algoritmu *Floyd-Warshall* slouží proměnná *VERSION* s přepínači:

- *C* - pro verzi v jazyce *C* ve formě *dynamické knihovny*
- *Python* - pro verzi v jazyce *Python*
- *PyCUDA* - pro paralelní *GPU* verzi pomocí knihovny *PyCUDA*
- *C_and_PyCUDA* - pro kombinaci verze *C* a *PyCUDA*

Pro funkcionalitu slouží zavolání jedné funkce:

- Název funkce: *run_TLDA*
  - Vstup: detekce míčku na jednotlivých snímcích a význačné body kurtu
  - Výstup: trajektorie

Zdrojový soubor *detect_players.py*
V tomto zdrojovém souboru je využita *CNN YOLOv3*. Pro funkcionalitu slouží zavolání dvou funkcí:

- Název funkce: *get_yolov3_model*
  - Vstup: souborová cesta k souboru s konfigurací *CNN YOLOv3*
  - Výstup: zkompilovaný model
- Název funkce: *get_players_bounding_boxes_in_images*
  - Vstup: snímky v BGR formátu, zkompilovaný model, význačné body kurtu a souborová cesta k souboru s konfigurací modelu
  - Výstup: *bounding boxy* ohraničující hráče

Zdrojový soubor *court_detection.py*
V tomto zdrojovém souboru je implementován algoritmus pro detekci význačných bodů tenisového kurtu. Pro funkcionalitu slouží zavolání jedné funkce:

- Název funkce: *get_specific_court_points_and_homographies*
  - Vstup: snímky v BGR formátu, délka základní horizontální čáry modelu tenisového kurtu v pixelech pro *projektivní transformaci*, velikost okolí v pixelech okolo modelu tenisového kurtu, výška snímků v *pixelech*, šířka snímků v *pixelech*
  - Výstup: význačné body tenisového kurtu
- Název funkce: *get_specific_court_points_and_homographies_parallel*
  - Vstup: snímky v BGR formátu, délka základní horizontální čáry modelu tenisového kurtu v pixelech pro *projektivní transformaci*, velikost okolí v pixelech okolo modelu tenisového kurtu, výška snímků v *pixelech*, šířka snímků v *pixelech*, počet úloh k vytvoření, struktura *pool* z knihovny *multiprocessing* s vytvořenými *procesy*
  - Výstup: význačné body tenisového kurtu

Zdrojový soubor *utils.py*
V tomto zdrojovém souboru je implementovány funkce využívané napříč ostatními zdrojovými soubory a funkce obecného charakteru jako například načítání obrázků apod.

Zdrojový soubor *events_detection.py*
V tomto zdrojovém souboru je implementován algoritmus pro detekci definovaných *událostí*. Pro funkcionalitu slouží zavolání jedné funkce:

- Název funkce: *get_events_sequence*
  - Vstup: trajektorie míčku, trajektorie míčku po filtraci, detekce hráčů, význačné body kurtu, *projektivní transformace* ze snímku do modelu kurtu, šířka snímků v *pixelech*, výška snímků v *pixelech*, délka základní horizontální čáry modelu tenisového kurtu v pixelech pro *projektivní transformaci*, velikost okolí v pixelech okolo modelu tenisového kurtu, *počet snímků za sekundu* zdrojového videa
  - Výstup: sekvence detekovaných událostí, odhad rychlosti servisu

Zdrojový soubor *visualize.py*
V tomto zdrojovém souboru je implementována vizualizace výstupů. Pro vizualizaci jsou spojovány úsečkami po sobě jdoucí detekce tenisové míčku v nalezené trajektorii. V této trajektorii jsou barevně vyznačeny nastalé události, jejichž popis je také vypisován do vyhrazeného místa. Pro funkcionalitu slouží zavolání jedné funkce:

- Název funkce: *visualize*
  - Vstup: snímky v BGR formátu, trajektorie míčku, sekvence detekovaných událostí, odhad rychlosti servisu
  - Výstup: snímky videa s vizualizacemi v BGR formátu

Zdrojový soubor *main.py*
V tomto zdrojovém souboru je implementována funkčnost systému, kdy jsou volány výše zmíněné funkce v příslušném pořadí. Jsou zde nastaveny parametry systému jako cesty k souborům apod. Program zapíše výsledky do specifikovaných souborů.

Zdrojový soubor *fw.c*
V tomto zdrojovém souboru je implementován *Floyd-Washallův* algoritmus v jazyku *C*. Kompilace do formy *sdílené knihovny* lze provést pomocí příkazů:

1. *gcc -c fw.c*
2. *gcc -shared fw.o -o fw.so*

Zdrojový soubor *evaluation.ipynb/py*
V tomto souboru je zdrojový kód využitý pro vyhodnocení správnosti poskytovaných výstupů řešení v kapitole 6.

5.3 Implementace vybraných paralelizací

Zde je popsána implementace vybraných paralelizací.

5.3.1 Paralelizace detekce kurtu

Vzhledem k omezení *global interpreter lock (GIL)* jazyka *Python*, kdy vždy pouze jedno *vlákno* může vykonávat kód programu je *více vláknová* paralelizace programu orientovaného na *CPU* čas nevhodná. K paralelizaci jsem tedy využil knihovnu *multiprocessing*, která umožňuje paralelizaci pomocí *procesů*. Neboť *procesy* nesdílejí paměťový prostor je kvůli snížení paměťové náročnosti datová struktura obsahující nahrané snímky sdílena pomocí třídy *Manager* ze zmíněné knihovny. Příslušná funkce očekává na vstupu navíc počet *procesů* k vytvoření odpovídajícího počtu úloh a tzv. *pool*, což je struktura s již vytvořenými *procesy*, které mohou být využívány pro různé poskytnuté úlohy, toto je vhodné, neboť vytváření nových *procesů* pro jiné úlohy je časově náročné.

5.3.2 Paralelizace *Floyd-Warshallova* algoritmu

Pro paralelizaci *Floyd-Warshallova* algoritmu na *GPU* jsem použil knihovnu *PyCUDA*, která umožňuje z jazyka *Python* využívat platformu *CUDA*. Kód spouštěný na *grafické kartě* musí být zapsán v syntaxi jazyka *CUDA C/C++* .

Kapitola 6

Testování

V této kapitole je uvedeno testování implementovaného řešení z hlediska výpočetního času a správnosti výstupů řešení.

6.1 Datové sady

Níže jsou uvedeny nalezené datové sady vztahující se k trajektorii míčku nebo událostem relevantním k trajektorii míčku:

1. Datová sada ve které jsou vyznačeny pozice míčku na jednotlivých snímcích. Zdroj: [42].

2. Datová sada ve které jsou anotovány následující události: *match*, *set*, *game*, *point*, *serve*, *hit*. Jednotlivé tenisové údery jsou anotovány přes interval snímků na kterých hráč provádí daný úder. Zdroj: [2]

3. Datová sada rychle se pohybujících objektů, ve které je určitá část věnována tenisu. Anotována je celá trajektorie. Zdroj: [52]

Ani jedna z nalezených datových sad plně nevyhovuje k testování přesnosti detekce tenisových *událostí*, proto byla vytvořena vlastní datová sada ze zápasu mezi *Serenou Williamsovou* a *Věrou Zvonarevovou* na Letních olympijských hrách v Londýně v roce 2012. Zdroj videa [2]. Rozlišení videa je 1280x720 a *počet snímků za vteřinu* je 25. Je vyznačen snímek, na kterém dochází k dané *události* vzhledem k trajektorii tenisového míčku. Tedy pro úder je anotován snímek na kterém míček změní směr v důsledku zásahu raketou, pro dopad je anotován snímek na kterém došlo k dopadu. Výčet anotovaných *událostí* je uveden v tabulce 6.1. Byly anotovány pouze výměny při kterých nedochází k přepnutí snímající kamery během výměny. Byly anotovány začátky a konce tenisových vyměň vzhledem k číslu snímku videa. Byla anotována rychlost podání, která je v případě tohoto videa měřena po úderu míčku raketou a pro

anotaci odečtena z informační tabule ve videu. Anotování bylo prováděno pomocí programu *CVAT* a anotace následně vygenerovány ve formátu *ImageNet*. Celkem bylo anotováno 93 tenisových výměn.

jméno události	zkratka	počet anotací	po konci výměny
land_far_in	LFI	99	29
land_far_out	LFO	57	30
hit_far_left	HFL	48	4
hit_far_right	HFR	42	2
serve_far_left	SFL	27	0
serve_far_right	SFR	22	0
land_net_far	LNF	11	0
serve_land_far_in	SLFI	32	1
serve_land_far_out	SLFO	4	0
land_near_in	LNI	103	38
land_near_out	LNO	17	5
hit_near_left	HNL	59	3
hit_near_right	HNR	38	4
serve_near_left	SNL	24	0
serve_near_right	SNR	20	0
land_net_near	LNN	15	0
serve_land_near_in	SLNI	35	0
serve_land_near_out	SLNO	6	0
rychlost podání	-	68	-
začátek a konec výměny	-	93	-

Tabulka 6.1: Tabulka s počty anotovaných událostí.

6.1.1 Problém s benchmarky

Výsledky metod používaných na detekcí objektu zájmu v různých sportech nejsou přímo porovnatelné díky odlišným podmínkám v těchto sportech. Jde například o jiné typy míče, míru zakrytí míče, četnost změny pohybu míče hráčem apod. Dále je problémem častá neexistence benchmarkových datových sad a databází výsledků pro jednotlivé sporty, což stěžuje porovnání jednotlivých metod a pokrok v tom kterém odvětví. [4]

6.2 Testování správnosti výstupů

6.2.1 Metodika testování

K testování správnosti výstupů řešení byly využity následující metriky.

1. Přesnost vzhledem k *editační vzdálenosti (edit distance)*, respektive *Levenštejnovy vzdálenosti (Levenshtein distance)*, která vyjadřuje minimální počet operací *vložení znaku (insert), smazání znaku (delete), náhrada znaku (substitution)* pro převedení jednoho řetězce na druhý, tedy v této práci převedení predikovaného řetězce událostí na skutečný řetězec události. Tato je dána vzorcem:

$$ED\ accuracy = \frac{N - D - S - I}{N},$$

   kde N je délka řetězce, D počet operací *delete*, S počet operací *substitution* a I počet operací *insert*.

2. Kategorie *true positive (TP)* pokud se predikovaná i skutečná anotace shodují, *false negative (FN)* pokud pro skutečnou anotaci není odpovídající predikce a *false positive (FP)* pokud se predikovaná anotace neshoduje se skutečnou. Z těchto údajů je možné vypočíst následující metriky:

   - metrika *precision*, která je dána vztahem

$$precision = \frac{TP}{TP + FP}\ ,$$

   - metrika *recall*, která je dána vztahem

$$recall = \frac{TP}{TP + FN}\ ,$$

   - metrika F_1 *score* vyjadřující *harmonický průměr* metrik *precision* a *recall* je dána vztahem

$$F_1 = \frac{2}{precision^{-1} * recall^{-1}}\ .$$

   Metrika *recall* vyjadřuje kolik anotací bylo správně predikováno, metrika *precision* relevanci predikcí vůči anotacím.

3. Průměrná absolutní procentuální chyba ($MAPE$) mezi odhadovanou a skutečnou rychlostí servisu:

$$MAPE = \frac{1}{N} \sum_{a=1}^{N} \frac{|skutečná\ rychlost - predikovaná\ rychlost|}{skutečná\ rychlost} \times 100.$$

4. Průměrná absolutní chyba (MAE) mezi odhadovanou a skutečnou rychlostí servisu:

$$MAE = \frac{1}{N}\sum_{a=1}^{N} |skutečná\ rychlost - predikovaná\ rychlost|.$$

5. *Míra korelace* mezi odhadovanou a skutečnou rychlostí servisu.

Pro uplatnění druhé metriky, respektive výpočtu kategorií TP, FP a FN je třeba rozdělit predikované události k anotacím. K tomuto je využito číslo snímku anotace a predikce a následující dvoufázový algoritmus:

1. V první fázi je interval snímků obsahující tenisovou výměnu rozdělen mezi jednotlivé anotace tak, že interval snímků mezi dvěma anotacemi je rozdělen na polovinu, kdy každá polovina připadne příslušné anotaci. Predikce v intervalu dané anotace jsou k ní přiřazeny, Pro každou anotaci je v intervalech jí přidělených hledána korektní predikce, pokud je nalezena je přiřazena trvale k anotaci, pokud je jich nalezeno více, je trvale přiřazena nejbližší, jestliže korektní predikce není nalezena je trvale přiřazena nejbližší nekorektní predikce.

2. Neboť korektní predikce k současné anotaci se může nacházet v intervalu přiřazeném jedné ze sousedních anotacím, jsou v druhé fázi prohledávány intervaly sousedních anotací. V těchto lze hledat hledat k zarážce vyjádřené jako bližší k číslu snímku současné anotace z

   - čísla snímku predikce trvale přiřazené k sousední anotaci
   - a čísla snímku sousední anotace.

   První zajistí, že predikce trvale přiřazené v první fázi nemohou být přeřazeny a druhé omezení zajistí, že predikce pro současnou anotaci se nemůže nacházet před nebo za sousední anotací vzhledem k číslům snímků.

   Jestliže je pro současnou anotaci v povoleném intervalu snímků sousední anotace nalezena korektní predikce, poté je interval snímků pro současnou anotaci prodloužen až po korektní predikci, čímž dojde k přiřazení korektní predikce k současné anotaci včetně predikcí mezi současnou anotací a korektní predikcí. Pokud je nalezena korektní predikce v intervalech snímků obou sousedních anotací, je přiřazena pouze bližší predikce.

Dále je třeba rozhodnout, jak vyhodnotit anotace událostí po korektním konci tenisové výměny. Například další dopady míčku po dopadu míčku do *outu*

nebo pokud je zahrán míček do *outu* a hráč míček i přesto odehraje na druhou stranu hřiště apod. Zvolil jsem možnost, že jsou hodnoceny všechny anotované události před koncem tenisové výměny. Anotované události po korektním konci tenisové výměny jsou pro metriky *TP, FP, FN* hodnoceny po poslední anotovanou událost, která má přiřazenou predikci a pro metriku *ED accuracy* je řetězec událostí tvořen anotovanými událostmi do konce tenisové výměny prodlužován postupně o jednu, dvě až všechny události po korektním konci tenisové výměny a je vzat nejlepší výsledek.

U metriky *TP* lze stanovit maximální vzdálenost predikce a anotace v počtu snímků. Pokud je maximální vzdálenost překročena je označena jako *FP*.

Metrika *ED accuracy* je použita v práci [38] sledující stejné výstupy, tedy anotaci tenisové výměny pomocí trajektorie tenisového míčku. Metrika F_1 *score* je použita v práci [2], kde jsou anotovány pouze údery hráčů, ale přes celý rozsah provádění úderu.

6.2.2 Výsledky měření správností výstupů

Výsledky testování korektnosti predikcí událostí tenisové výměny byly testovány na datové sadě uvedené v sekci 6.1.

V tabulce 6.2 jsou uvedeny výsledky metrik TP, FP, FN, *recall*, *precision* a F_1 *skóre* pro anotované události a jejich vybrané agregace bez omezení vzdálenosti v počtu snímků mezi predikci a anotací.

V tabulce 6.3 jsou uvedeny výsledky metrik TP, FP, FN, *recall*, *precision* a F_1 *skóre* pro anotované události a jejich vybrané agregace s omezením vzdálenosti v počtu snímků mezi predikcí a anotací na 5 snímků

Na obrázku 6.1 jsou uvedeny *krabicové grafy* vzdáleností v počtu snímků mezi predikcemi a anotacemi pro vybrané události a jejich agregace pro verzi bez omezení této vzdálenosti.

Na obrázku 6.2 jsou uvedeny *krabicové grafy* vzdáleností v počtu snímků mezi predikcemi a anotacemi pro vybrané události a jejich agregace pro verzi s omezením této vzdálenosti na 5 snímků.

V tabulce 6.4 jsou uvedeny výsledky metriky *ED accuracy*. Byly testovány agregace událostí určující na jaké straně vzhledem k hráči byl míček zasažen, zda hráč provádí úder podání zleva či zprava a zda je míček v či vně kurtu.

V tabulce 6.5 jsou uvedeny výsledky pro predikování rychlosti podání bez a s odstraněnými měřeními s příliš velkou chybou. Na obrázku 6.3 je uveden graf mající na ose x anotované rychlosti podání, na ose y hodnoty predikované a diagonálu ukazující polohu, kde nastává nulová chyba predikce.

6.2.3 Vyhodnocení

Výsledek pro metriku *ED accuracy* v tabulce 6.4 je 84,62 %, tedy na 10 anotací je potřeba 1-2 úpravy. Tento výsledek je srovnatelný s prací [38], která má *ED accuracy* 84.3 %, sledující stejný cíl pomocí jiného přístupu, využívající ovšem jinou datovou sadu a dělící anotace události dopadu servisu do korektní oblasti ($SLFI$ a $SLNI$) na čtyři anotace určující zda míček dopadl do oblasti pro dopad servisu vlevo či vpravo, což je určeno podáním hráče zleva či zprava, jinak jsou třídy anotovaných událostí totožné.

Pokud jsou události agregovány tak, že údery a podání jsou děleny jen dle hráče na horní a dolní části hřiště a dopady jsou děleny pouze dle horní a dolní části hřiště stoupne *ED accuracy* na 90,38 %.

V tabulce 6.2 lze nahlédnout, že kromě události LNO je dosaženo u metriky *recall* hodnot cca. 80 % a více. Pokud jsou události agregovány jak bylo uvedeno výše, stoupne hodnota k přibližně 90 %, kromě události LNF, která má hodnotu 81,82 %.

U metriky *precision* je kromě událostí LFO, LNO, $SLNO$, LNF a HNR dosahováno hodnot přes 80 %. Při agregaci, viz výše, stoupne metrika přes 90 %, kromě událostí LNF a LNN, kterých se agregace netýká.

Lze vyzdvihnout úspěšné určování jak podávajícího hráče, tak zda hráč podává zleva či zprava. Dále vysoké hodnoty *precision* pro dopady v kurtu, naopak dopady vně kurtu jsou slabinou řešení.

Bylo prozkoumáno, co způsobuje *falešně pozitivní* detekce u událostí LFO, LNO a $SLNO$, zda se nejedná o systematickou chybu. U událostí LFO a LNO jde o kombinaci nedostatečně přesného způsobu určování dopadu míčku, šumu zahrnutém v trajektorii, šumu v umístění detekcí míčku a chybějících detekcí, u události $SLNO$ byly způsobeny chybějícími detekcemi.

Na obrázku 6.1 lze nahlédnout, že pro událostí, kde je žádoucí přesnost predikce události vzhledem k vzdálenosti v počtu snímků od anotace, tedy události servis kvůli určováni rychlosti podání a události dopad pro určování téhož a zda je dopad v či vně kurtu, je většina predikcí vzdálena do dvou snímků kromě výjimek, jejichž výskyt je u těchto událostí okolo 6 %. V tabulce 6.3 lze pozorovat konkrétní změnu metrik po odstranění TP predikcí vzdálených od anotace o více než 5 snímků, kdy pro agregaci hit_far je odstraněno 11 TP predikcí, pro agregaci hit_near 6 TP predikcí, pro agregaci $land\_far$ 8 TP predikcí a pro agregaci $land\_near$ 3 TP predikce a $serve\_far$ 3 TP predikce.

U rychlosti podání lze pozorovat, při pominutí vyznačených odlehlých hodnot, vysokou míru korelace o hodnotě 0,84 mezi skutečnou a predikovanou rychlostí podání. Predikovaná rychlost je většinou menší nežli skutečná, což je očekávaný výsledek, neboť skutečná rychlost je rychlost míčku po udeření raketou a predikovaná rychlost je průměrná rychlost mezi udeřením míčku

a jeho dopadem. Vysoké chyby u třech predikcí podání byly způsobeny jednou zakrytím nadhozu míčku informační tabulkou se skóre a dvakrát špatnou detekci dopadu míčku po podání.

6.3 Měření výpočetního času

6.3.1 Hardwarová konfigurace

Výpočetní čas byl měřen na procesoru *intel i-53550 3.30 GHz*, grafické kartě *NVIDIA GeFroce GTX 660* s 960 *CUDA jádry*, dále počítač disponuje *8 GB* operační paměti a úložištěm typu *SSD*.

6.3.2 Metodika měření

Čas byl měřen pomocí funkce *time* z knihovny *time* v jazyce *Python*. Měření probíhalo pro více sekvencí snímků obsahujících tenisovou výměnu, kdy program běžel přes tyto sekvence kontinuálně. Byla nahrána informace o počátečním a koncovém snímku sekvence obsahující tenisovou výměnu, poté byla sekvence zpracována a toto se opakovalo pro další sekvenci snímků. Jednotlivé části systému běží vůči sobě sekvenčně v daném pořadí, jak bylo uvedeno v sekci 5.1.

6.3.3 Výsledky měření výpočetního času

Výpočetní čas byl měřen na 5 sekvencích snímků, kdy každá sekvence zahrnuje tenisovou výměnu a celkový počet snímků je 1719. Rozlišení snímků je *1280*x*720*. Detekce kurtu běží paralelně na 4 jádrech *CPU* a pro algoritmus *Floyd-Warshall* je využita kombinace sekvenční verze v jazyce *C* ve formě *dynamické knihovny* a paralelní *GPU* verze pomocí knihovny *PyCUDA*. *CNN TrackNet* a *YOLOv3* běží s využitím paralelizace na *GPU*. Výsledky jsou uvedeny v tabulce 6.6.

V tabulce 6.9 jsou uvedeny měření části systému hledající trajektorii míčku s různými verzemi algoritmu *FW*.

Čas byl měřen pro každou část systému zvlášť a u detekce hráčů byl měřen čas zvlášť pro *CNN YOLOv3* a zbytek programu.

Dále byla změřena paralelní verze *detekce kurtu* pro různý počet snímků. Dvakrát větší počet snímků lze brát jen jako zhruba dvakrát náročnější úlohu, neboť každý snímek je odlišný a tedy hledání kurtu může být jinak náročné. Výsledky jsou uvedeny v tabulce 6.7.

Pro algoritmus *Floyd-Warshall* byla změřena paralelizace na *GPU* pomocí knihovny *PyCUDA* jazyka *Python* a převedení algoritmu do jazyka *C* s následným využitím ve formě *dynamické knihovny* v jazyku *Python*. Je měřen čas běhu samotného algoritmu *Floyd-Warshall* a všech věcí příslušných k dané verzi, tedy

přenosů dat z/na *GPU*, inicializace prostředí, nahraní *dynamické knihovny*. Výsledky jsou uvedeny v tabulce 6.8.

6.3.4 Vyhodnocení

Z měření uvedených v tabulce 6.6 lze pozorovat, že na výpočetním čase se podílí z drtivé části 4 části systému a to dvě *CNN*, detekce kurtu a hledání trajektorie. Ostatní části jsou oproti nim zanedbatelné. Při převedení do jazyka *C* a paralelizaci na *GPU Floyd-Warshallova* algoritmu, který je přidán k algoritmu *TLDA*, bylo dosaženo hodnoty *zrychlení* 5,8 pro část systému hledající trajektorii tenisového míčku. Při paralelizaci detekce kurtu s 4 *procesy* na 4 *procesorových jádrech* bylo dosaženo hodnoty *zrychlení* 2,85 a *efektivity* pro jedno *jádro CPU* 0,71 pro testovaný případ s 1719 snímky.

Paralelizace *Floyd-Warshallova* algoritmu je vhodná až od velikosti matice přibližně 800x800, do této velikosti je rychlejší verze v jazyce *C*, neboť režie spojená s paralelizací je přibližně 1,4 sekundy, viz tabulku 6.8. V měření v tabulce 6.9 nepřináší kombinace verze algoritmu *FW* v jazyce *C* a paralelizace pomoci knihovny *PyCUDA* oproti verzi s jazyken *C* zrychlení, neboť testované matice měli menši velikost než 800x800 od které je uplatněna verze *PyCUDA*. Velikost matice záleží na délce tenisové výměny a nastaveni parametrů algoritmu *TLDA*, tedy v případě větších matic by se benefity kombinované verze projevily.

CNN TrackNet má velmi vysoký výpočetní čas oproti ostatním částem systému a to včetně *CNN YOLOv3*. Bylo by vhodné se zabývat úpravami této *CNN*, kdy jako výchozí bod lze využít její přepracování pro detekci badmintonového míčku v práci [7], kde je kladen důraz na redukci výpočetního času, viz kapitolu 3.3.

Celková rychlost řešení na této hardwarové konfiguraci je zpracování 0,68 snímku za sekundu.

Dále by bylo vhodné implementovat paralelismus mezi jednotlivými částmi systému, neboť vzhledem k závislostem lze například umožnit běh detekce kurtu na *CPU* a zároveň běh některé *CNN* na *GPU*. Tedy vytěžit dostupné výpočetní prostředky tak, aby byl omezen čas po který nejsou využity na minimum. Tímto se v práci více nezabývám.

jméno události	zkratka	TP	FP	FN	*recall*	*precision*	F_1 *skóre*
land_far_in	LFI	66	5	14	82,50 %	92,96 %	87,42 %
land_far_out	LFO	31	10	8	79,49 %	75,61 %	77,5 %
hit_far_left	HFL	43	6	5	89,58 %	87,76 %	88,66 %
hit_far_right	HFR	34	7	8	80,95 %	82,93 %	81,93 %
serve_far_left	SFL	27	2	0	100,00 %	93,10 %	96,43 %
serve_far_right	SFR	18	1	4	81,82 %	94,73 %	87,80 %
land_net_far	LNF	9	3	2	81,82 %	75,00 %	78,26 %
serve_land_far_in	SLFI	31	0	0	100,00 %	100,00 %	100,00 %
serve_land_far_out	SLFO	4	0	0	100,00 %	100,00 %	100,00 %
land_near_in	LNI	59	2	12	83,10 %	96,72 %	89,39 %
land_near_out	LNO	12	6	5	70,59 %	66,67 %	68,57 %
hit_near_left	HNL	51	7	8	86,44 %	87,93 %	87,18 %
hit_near_right	HNR	31	9	7	81,58 %	77,50 %	79,49 %
serve_near_left	SNL	23	0	1	95,83 %	100,00 %	97,87 %
serve_near_right	SNR	19	1	1	95,00 %	95,00 %	95,00 %
land_net_near	LNN	14	3	1	93,33 %	82,35 %	87,50 %
serve_land_near_in	SLNI	30	0	5	85,71 %	100,00 %	92,31 %
serve_land_near_out	SLNO	5	3	1	83,33 %	62,5 %	71,43 %
jméno agregace	zkratky v agregaci	TP	FP	FN	*recall*	*precision*	F_1 *skóre*
hit_far	HFL, HFR	84	6	6	93,33 %	93,33 %	93,33 %
hit_near	HNL, HNR	89	9	8	91.75 %	90,82 %	91.28 %
land_far	SLFI, SLFO, LFO, LFI	136	11	18	88,31 %	92,52 %	90,37 %
land_near	SLNI, SLNO, LNO, LNI	113	4	16	87,60 %	96,58 %	91,87 %
serve_far	SFL, SFR	47	1	2	95,92 %	97,92 %	96,91 %
serve_near	SNL, SNR	43	0	1	97,73 %	100,00 %	98,85 %

Tabulka 6.2: Tabulka s výsledky testování korektnosti výstupů. Bez omezení vzdálenost predikce a anotace v počtu snímků.

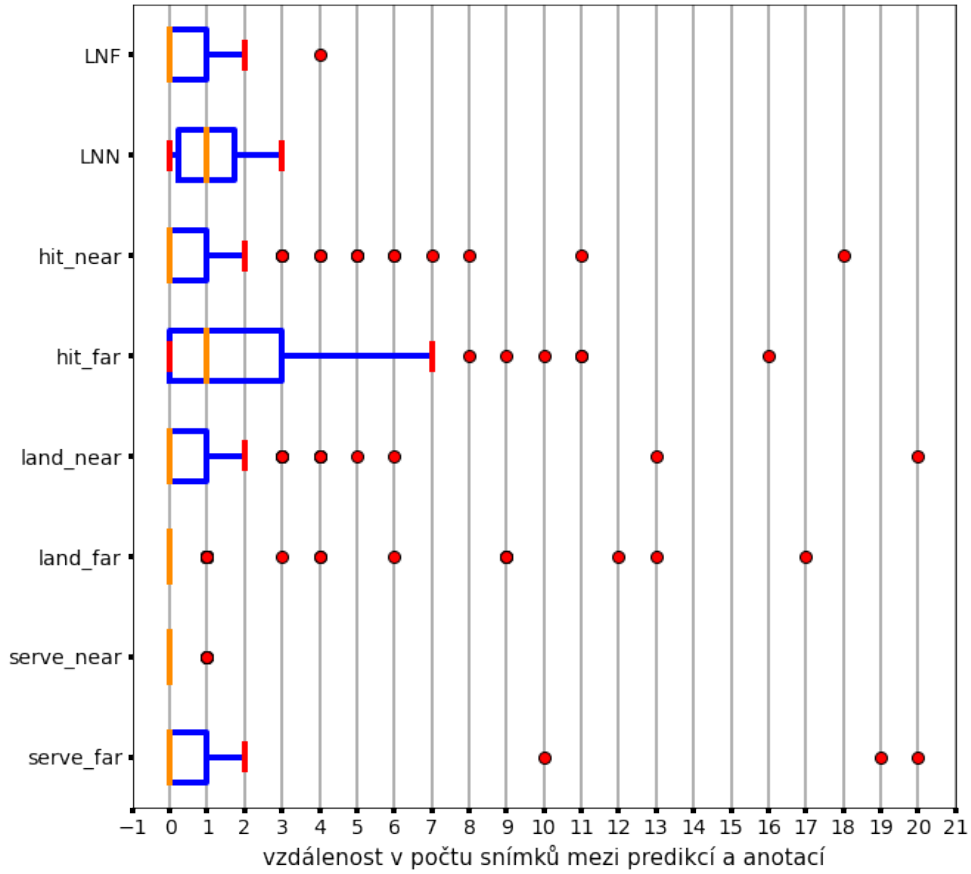
jméno události	zkratka	TP	FP	FN	*recall*	*precision*	F_1 *skóre*
land_far_in	LFI	63	8	14	81,82 %	88,73 %	85,14 %
land_far_out	LFO	28	12	7	77,78 %	68,29 %	72,73 %
hit_far_left	HFL	37	12	5	88,09 %	75,51 %	81,32 %
hit_far_right	HFR	32	9	8	80,00 %	78,05 %	79,01 %
serve_far_left	SFL	27	2	0	100,00 %	93,10 %	96,43 %
serve_far_right	SFR	15	4	4	78,95 %	78,95 %	78,95 %
land_net_far	LNF	9	3	2	81,82 %	75,00 %	78,26 %
serve_land_far_in	SLFI	31	0	0	100,00 %	100,00 %	100,00 %
serve_land_far_out	SLFO	4	0	0	100,00 %	100,00 %	100,00 %
land_near_in	LNI	59	2	12	83,10 %	96,72 %	89,39 %
land_near_out	LNO	9	9	5	64,29 %	50,00 %	56,25 %
hit_near_left	HNL	49	9	8	85,96 %	84,48 %	85,22 %
hit_near_right	HNR	31	9	7	81,58 %	77,50 %	79,49 %
serve_near_left	SNL	23	0	1	95,83 %	100,00 %	97,87 %
serve_near_right	SNR	19	1	1	95,00 %	95,00 %	95,00 %
land_net_near	LNN	14	3	1	93,33 %	82,35 %	87,50 %
serve_land_near_in	SLNI	30	0	5	85,71 %	100,00 %	92,31 %
serve_land_near_out	SLNO	5	3	1	83,33 %	62,5 %	71,43 %
jméno agregace	zkratky v agregaci	TP	FP	FN	*recall*	*precision*	F_1 *skóre*
hit_far	HFL, HFR	73	17	6	92,41 %	81,12 %	85,39 %
hit_near	HNL, HNR	83	15	8	92.31 %	85,71 %	88,89 %
land_far	SLFI, SLFO, LFO, LFI	128	18	18	87,76 %	87,76 %	87,76 %
land_near	SLNI, SLNO, LNO, LNI	110	7	16	87,30 %	94,02 %	90,53 %
serve_far	SFL, SFR	44	4	2	95,65 %	91,67 %	93,61 %
serve_near	SNL, SNR	43	0	1	97,73 %	100,00 %	98,85 %

Tabulka 6.3: Tabulka s výsledky testování korektnosti výstupů. Vzdálenost predikce a anotace omezena na 5 snímků.

agregace událostí	*ED accuracy*	délka řetězce predikcí	počet ED oprav
-	84,62 %	572	88
(HFL, HFR), (HNL, HNR)	87,41 %	572	72
(SFR, SFL), (SNR, SNL)	85,14 %	572	85
(LNI, LNO, SLNI, SLNO), (LFI, LFO, SLFI, SLFO)	86,89 %	572	75
(LNI, LNO, SLNI, SLNO), (LFI, LFO, SLFI, SLFO), (HFL, HFR), (HNL, HNR) (SFR, SFL), (SNR, SNL)	90,38 %	572	55

Tabulka 6.4: Tabulka s výsledky testování korektnosti výstupů. Událostí v závorce jsou agregovány do jedné události.

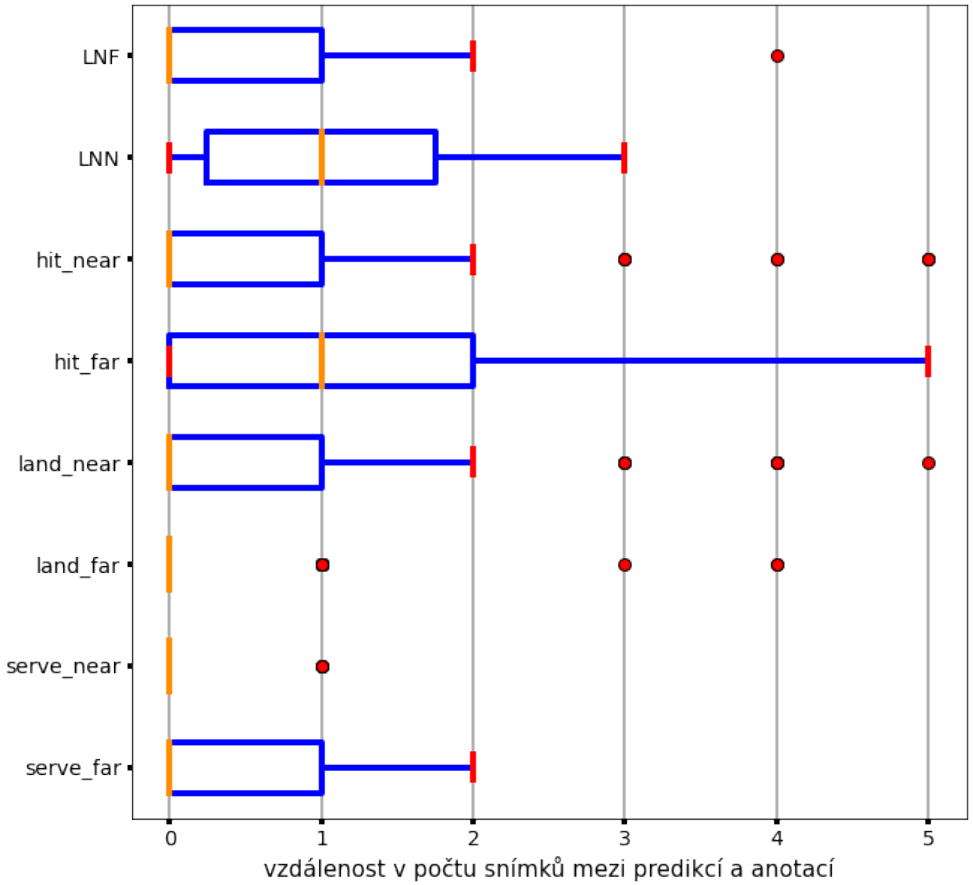
Obrázek 6.1: Vzdálenosti mezi predikcí a anotací v počtu snímků pro *true positive* predikce. Bez omezení vzdálenost predikce a anotace v počtu snímků. Vyjádřeno pomocí *krabicového grafu (box plot)*, kde *vousy* reprezentují 1,5 násobek *kvartilového rozpětí*.



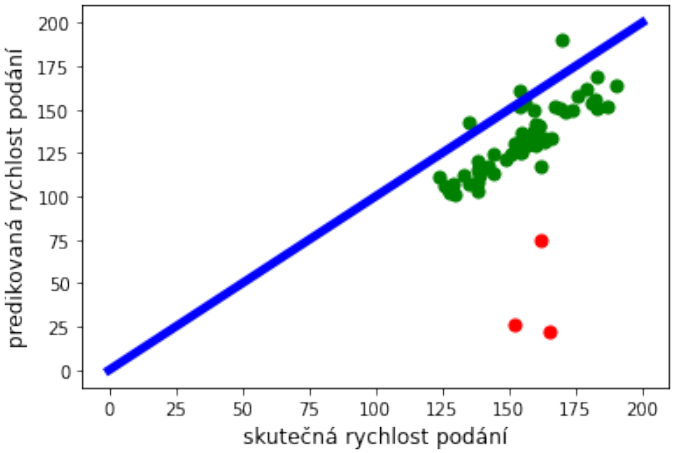
verze	počet predikcí	*MAE*	*MAPE*	*míra korelace*
všechna měření	65	27,81 km/h	18,00 %	0,53
bez odlehlých měření	62	23,41 km/h	15,25 %	0,84

Tabulka 6.5: Tabulka s výsledky testování korektnosti predikce rychlosti podání.

Obrázek 6.2: Vzdálenosti mezi predikcí a anotací v počtu snímků pro *true positive* predikce, které mají maximální povolenou vzdálenost 5 snímků. Vyjádřeno pomocí *krabicového grafu (box plot), kde vousy reprezentují 1,5 násobek kvartilového rozpětí.*



Obrázek 6.3: Skutečná a predikovaná rychlost podání. Červeně jsou označeny odlehlé hodnoty.



Měření výpočetního času jednotlivých částí systému s paralelizacemi		
část systému	čas v sekundách	procentuální vyjádření
TrackNet	1702,90 *s*	67,02 %
detekce hráčů - YOLOv3	482,40*s*	18,99 %
detekce hráčů - zbytek	19,66 *s*	0,77 %
detekce kurtu - par. 4 jádra	223,65 *s*	8,80 %
hledání trajektorie, FW kombinace C a PyCUDA	73.87 *s*	2,91 %
nahrání modelů CNN	20,11 *s*	0,79 %
vstup/výstup	15,05 *s*	0,59 %
detekce událostí	2,23 *s*	0,09 %
vizualizace	1,06 *s*	0,04 %
celkový čas	2521.45 *s*	100 %

Tabulka 6.6: Tabulka s výsledky měření výpočetního času jednotlivých částí systému. Vyjádřeno v absolutním čase v sekundách a procentuálním vyjádření. Čas i procenta jsou zaokrouhlena na dvě desetinná místa. Zkratka *par.* znamená paralelní verze a zkratka *FW Floyd-Warshallův algoritmus.*

Detekce kurtu - paralelní verze				
-	čas v sekundách			
počet snímků/verze	sekvenční verze	1 proces	2 procesy	4 procesy
50	12,72 *s*	16,57 *s*	11,31 *s*	9,89 *s*
100	21,47 *s*	25,77 *s*	17,36 *s*	13,38 *s*
200	47,43 *s*	52,47 *s*	30,53 *s*	21,44 *s*
400	126,85 *s*	130,84 *s*	75,36 *s*	49,74 *s*
800	364,72 *s*	387,44 *s*	221,80 *s*	130,87 *s*
1719	587,55 *s*	634,59 *s*	328,92 *s*	206,49 *s*

Tabulka 6.7: Tabulka s výsledky měření výpočetního času paralelní *CPU* verze a sekvenční verze detekce kurtu. Čas je zaokrouhlen na dvě desetinná místa.

Floyd Warshallův algoritmus - paralelní verze			
-	čas v sekundách		
velikost matice/verze	sekv. - Python	sekv. - C	GPU - PyCUDA
50x50	0,172 *s*	0 *s*	1,441 *s*
100x100	1,797 *s*	0,016 *s*	1,426 *s*
200x200	16,048 *s*	0,046 *s*	1.464 *s*
400x400	134,583 *s*	0,406 *s*	1,602 *s*
600x600	-	1,422 *s*	1,807 *s*
800x800	-	3,365 *s*	2,053 *s*
1600x1600	-	27,185 *s*	3,825 *s*

Tabulka 6.8: Tabulka s výsledky měření výpočetního času paralelní *GPU* verze a sekvenční verze v jazyku *Python* a *C Floyd Warshallova* algoritmu. Čas je zaokrouhlen na tři desetinná místa.

Měření výpočetního času části systému hledání trajektorie.	
část systému	čas v sekundách
FW - Python	434.06 *s*
FW - C	74.50 *s*
FW - PyCUDA	82.55 *s*
FW - C a PyCUDA	73.87 *s*

Tabulka 6.9: Tabulka s výsledky měření výpočetního času části systému *hledání trajektorie* s různými verzemi algoritmu *FW*. Pro kombinaci C a PyCUDA je PyCUDA uplatněna pro matice od velikosti 800x800.

Závěr

V práci byla učiněna rešerše přístupů využívaných pro detekcí tenisového míčku a jeho trajektorie v obrazovém záznamu. Byly určeny vhodné výstupy z trajektorie. Byl navrhnut přistup detekující tyto výstupy využívající informace o trajektorii míčku, tenisovém kurtu a hráčích. Navržený přístup byl implementován. Byly navrhnuty paralelizace a vybrané paralelizace také implementovány. Řešení bylo otestováno z hlediska výpočetního času a korektnosti poskytovaných výstupů pro trajektorii míčku.

Výsledky práce ukazují, že zvolený přístup lze pro danou problematiku využít, neboť na zvolených metrikách dosahuje relativně vysokých hodnot. Řešení by mohlo být vylepšeno přesnějším určováním dopadu míčku v a vně tenisového kurtu a obohacením o například o zvukovou informaci. Řešení má vysoký výpočetní čas, bylo by proto dále vhodné implementovat navržené změny.

Literatura

[1] NielsF: The dimensions of a tennis court. 2006, cit. 2022-01-01, licence: CC-BY-SA 3.0. Dostupné z: `https://en.wikipedia.org/wiki/File:Atomic_force_microscope_block_diagram.svg`

[2] Faulkner, H.; Dick, A.: TenniSet: A Dataset for Dense Fine-Grained Event Recognition, Localisation and Description. In *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, IEEE, s. 1–8.

[3] Posted by Kamal Jain on August 11, . a. a. B.: Understanding of Artificial Neural Networks. Cit. 2021-11-5. Dostupné z: `https://www.datasciencecentral.com/profiles/blogs/understanding-of-artificial-neural-networks`

[4] Kamble, P. R.; Keskar, A.; Bhurchandi, K.: Ball tracking in sports: a survey. *Artificial Intelligence Review*, 2017: s. 1–51.

[5] Huang, Y.-C.; Liao, I.-N.; Chen, C.-H.; aj.: TrackNet: A Deep Learning Network for Tracking High-speed and Tiny Objects in Sports Applications*. In *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2019, s. 1–8, doi: 10.1109/AVSS.2019.8909871.

[6] Archana, M.; Geetha, M.: Object Detection and Tracking Based on Trajectory in Broadcast Tennis Video. *Procedia Computer Science*, ročník 58, 12 2015: s. 225–232, doi:10.1016/j.procs.2015.08.060.

[7] Sun, N.-E.; Lin, Y.-C.; Chuang, S.-P.; aj.: TrackNetV2: Efficient Shuttlecock Tracking Network. In *2020 International Conference on Pervasive Artificial Intelligence (ICPAI)*, 2020, s. 86–91, doi:10.1109/ICPAI51961.2020.00023.

[8] Kamble, P. R.; Keskar, A. G.; Bhurchandi, K. M.: A convolutional neural network based 3D ball tracking by detection in soccer videos. In *Eleventh International Conference on Machine Vision (ICMV 2018)*, ročník 11041, editace A. Verikas; D. P. Nikolaev; P. Radeva; J. Zhou, International Society for Optics and Photonics, SPIE, 2019, s. 730 – 737, doi:10.1117/12.2522844. Dostupné z: https://doi.org/10.1117/12.2522844

[9] Renò, V.; Mosca, N.; Marani, R.; aj.: Convolutional Neural Networks Based Ball Detection in Tennis Games. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, s. 1839–18396, doi:10.1109/CVPRW.2018.00228.

[10] Yu, X.; Sim, C.-H.; Wang, J.; aj.: A trajectory-based ball detection and tracking algorithm in broadcast tennis video. In *2004 International Conference on Image Processing, 2004. ICIP '04.*, ročník 2, 2004, s. 1049–1052 Vol.2, doi:10.1109/ICIP.2004.1419482.

[11] Pingali, G.; Opalach, A.; Jean, Y.: Ball tracking and virtual replays for innovative tennis broadcasts. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, ročník 4, 2000, s. 152–156 vol.4, doi:10.1109/ICPR.2000.902885.

[12] Yan, F.; Christmas, W.; Kittler, J.: *Ball Tracking for Tennis Video Annotation*. Cham: Springer International Publishing, 2014, ISBN 978-3-319-09396-3, s. 25–45, doi:10.1007/978-3-319-09396-3_2. Dostupné z: https://doi.org/10.1007/978-3-319-09396-3_2

[13] Saxena, P.: Object detection vs object recognition vs image segmentation. Feb 2020, [cit. 2021-10-15]. Dostupné z: https://www.geeksforgeeks.org/object-detection-vs-object-recognition-vs-image-segmentation/

[14] James, G.; Witten, D.; Hastie, T.; aj.: *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. Dostupné z: https://faculty.marshall.usc.edu/gareth-james/ISL/

[15] Russell, S.; Norvig, P.: *Artificial Intelligence: A Modern Approach*. USA: Prentice Hall Press, třetí vydání, 2009, ISBN 0136042597.

[16] Shalev-Shwartz, S.; Ben-David, S.: *Understanding Machine Learning: From Theory to Algorithms*. USA: Cambridge University Press, 2014, ISBN 1107057132.

[17] Hastie, T.; Tibshirani, R.; Friedman, J.: *The elements of statistical learning: data mining, inference and prediction*. Springer, druhé vydání, 2009. Dostupné z: http://www-stat.stanford.edu/~tibs/ElemStatLearn/

[18] Goodfellow, I.; Bengio, Y.; Courville, A.: *Deep Learning*. The MIT Press, 2016, ISBN 0262035618.

[19] Ekinci, B. D.; Gokmen, M.: A Ball Tracking System for Offline Tennis Videos. In *Proceedings of the 1st WSEAS International Conference on Visualization, Imaging and Simulation*, VIS'08, Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2008, ISBN 9789604740222, str. 45–48.

[20] Chakraborty, B.; Meher, S.: A trajectory-based ball detection and tracking system with applications to shot-type identification in volleyball videos. In *2012 International Conference on Signal Processing and Communications (SPCOM)*, 2012, s. 1–5, doi:10.1109/SPCOM.2012.6290005.

[21] D'Orazio, T.; Leo, M.; Spagnolo, P.; aj.: An Investigation Into the Feasibility of Real-Time Soccer Offside Detection From a Multiple Camera System. *Circuits and Systems for Video Technology, IEEE Transactions on*, ročník 19, 01 2010: s. 1804 – 1818, doi:10.1109/TCSVT.2009.2026817.

[22] Stauffer, C.; Grimson, W.: Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, ročník 2, 1999, s. 246–252 Vol. 2, doi:10.1109/CVPR.1999.784637.

[23] Ren, J.; Orwell, J.; Jones, G. A.; aj.: Real-Time Modeling of 3-D Soccer Ball Trajectories From Multiple Fixed Cameras. *IEEE Transactions on Circuits and Systems for Video Technology*, ročník 18, č. 3, 2008: s. 350–362, doi:10.1109/TCSVT.2008.918276.

[24] Ishii, N.; Kitahara, I.; Kameda, Y.; aj.: 3D Tracking of a Soccer Ball Using Two Synchronized Cameras. 12 2007, ISBN 978-3-540-77254-5, s. 196–205, doi:10.1007/978-3-540-77255-2_22.

[25] Chakraborty, B.; Meher, S.: A trajectory-based ball detection and tracking system with applications to shooting angle and velocity estimation in basketball videos. In *2013 Annual IEEE India Conference (INDICON)*, 2013, s. 1–6, doi:10.1109/INDCON.2013.6725963.

[26] Teachabarikiti, K.; Chalidabhongse, T. H.; Thammano, A.: Players tracking and ball detection for an automatic tennis video annotation. In *2010 11th International Conference on Control Automation Robotics Vision*, 2010, s. 2461–2494, doi:10.1109/ICARCV.2010.5707906.

[27] Fazio, M.; Fisher, K.; Fujinami, T.: Tennis Ball Tracking : 3 D Trajectory Estimation using Smartphone Videos. 2018.

[28] Zhang, Z.; Xu, D.; Tan, M.: Visual Measurement and Prediction of Ball Trajectory for Table Tennis Robot. *IEEE Transactions on Instrumentation and Measurement*, ročník 59, č. 12, 2010: s. 3195–3205, doi: 10.1109/TIM.2010.2047128.

[29] Pingali, G.; Jean, Y.; Carlbom, I.: Real time tracking for enhanced tennis broadcasts. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*, 1998, s. 260–265, doi:10.1109/CVPR.1998.698618.

[30] Tian, B.; Zhang, D.; Zhang, C.: High-Speed Tiny Tennis Ball Detection Based on Deep Convolutional Neural Networks. In *2020 IEEE 14th International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, 2020, s. 30–33, doi:10.1109/ASID50160.2020.9271695.

[31] Qiao, F.: Application of deep learning in automatic detection of technical and tactical indicators of table tennis. *PLOS ONE*, ročník 16, č. 3, 03 2021: s. 1–16, doi:10.1371/journal.pone.0245259. Dostupné z: `https://doi.org/10.1371/journal.pone.0245259`

[32] Kotera, J.; Rozumnyi, D.; Sroubek, F.; aj.: Intra-frame Object Tracking by Deblatting. *CoRR*, ročník abs/1905.03633, 2019, `1905.03633`. Dostupné z: `http://arxiv.org/abs/1905.03633`

[33] Rozumnyi, D.; Kotera, J.; Šroubek, F.; aj.: Non-causal Tracking by Deblatting. In *Pattern Recognition*, editace G. A. Fink; S. Frintrop; X. Jiang, Cham: Springer International Publishing, 2019, ISBN 978-3-030-33676-9, s. 122–135.

[34] Rozumnyi, D.; Matas, J.; Sroubek, F.; aj.: FMODetect: Robust Detection and Trajectory Estimation of Fast Moving Objects. *CoRR*, ročník abs/2012.08216, 2020, `2012.08216`. Dostupné z: `https://arxiv.org/abs/2012.08216`

[35] Yan, F.; Kittler, J.: A Tennis Ball Tracking Algorithm for Automatic Annotation of Tennis Match. *British Machine Vision Conference*, 01 2005, doi:10.5244/C.19.67.

[36] Poliakov, A.; Marraud, D.; Reithler, L.; aj.: Physics based 3D ball tracking for tennis videos. In *2010 International Workshop on Content Based Multimedia Indexing (CBMI)*, 2010, s. 1–6, doi:10.1109/CBMI.2010.5529897.

[37] Zhou, X.; Xie, L.; Huang, Q.; aj.: Tennis Ball Tracking Using a Two-Layered Data Association Approach. *IEEE Transactions on Multimedia*, ročník 17, 2015: s. 145–156.

[38] Almajai, I.; Kittler, J.; de Campos, T.; aj.: Ball event recognition using hmm for automatic tennis annotation. In *2010 IEEE International Conference on Image Processing*, 2010, s. 1509–1512, doi:10.1109/ICIP.2010.5652415.

[39] Christmas, W. J.; Kostin, A.; Yan, F.; aj.: A system for the automatic annotation of tennis matches. In *Fourth International Workshop on Content based Multimedia Indexing*, 2005.

[40] Yan, F.; Kittler, J.; Windridge, D.; aj.: Automatic annotation of tennis games: An integration of audio, vision, and learning. *Image and Vision Computing*, ročník 32, č. 11, 2014: s. 896–903, ISSN 0262-8856, doi:https://doi.org/10.1016/j.imavis.2014.08.004. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0262885614001309

[41] Yu-Chuan Huang, C.-H. C. T.-U. W.-C. P., I-No Liao: TrackNet: Tennis Ball Tracking from Broadcast Video by Deep Learning Networks. Nov 2019, cit. 2021-10-10. Dostupné z: https://nol.cs.nctu.edu.tw:234/open-source/TrackNet

[42] NOL, N.: Tracknet dataset. https://nol.cs.nctu.edu.tw/ndo3je6av9/, cit. 2021-05-10.

[43] Mahmud, S.: Parallel Programming With CUDA Tutorial (Part-4: The Floyd-Warshall Algorithm). Jul 2020, cit. 2021-12-10. Dostupné z: https://saadmahmud14.medium.com/parallel-programming-with-cuda-tutorial-part-4-the-floyd-warshall-algorithm-5e1281c46bf6

[44] Redmon, J.; Farhadi, A.: YOLOv3: An Incremental Improvement. 2018, 1804.02767.

[45] Anh, H. N.: Training and Detecting Objects with YOLO3. Jan 2020, cit. 2021-11-10. Dostupné z: https://github.com/experiencor/keras-yolo3

[46] Farhat, M.; Khalfallah, A.; Bouhlel, M.: A Real-Time Court Detection and Tracking System to Tennis Videos. *Journal of Testing and Evaluation*, ročník 47, 07 2019, doi:10.1520/JTE20170099.

[47] Rea, N.; Dahyot, R.; Kokaram, A.: Classification and representation of semantic content in broadcast tennis videos. In *IEEE International Conference on Image Processing 2005*, ročník 3, 2005, s. III–1204, doi:10.1109/ICIP.2005.1530614.

[48] Lugade, V.: Determine if a point is located within a convex polygon. Aug 2018, cit. 2021-11-10. Dostupné z: https://matlabgeeks.com/tips-tutorials/computational-geometry/determine-if-a-point-is-located-within-a-convex-polygon/

[49] GeeksforGeeks: Floyd Warshall Algorithm — DP-16. Nov 2021, [cit. 2022-01-05]. Dostupné z: https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/

[50] GeeksforGeeks: Disjoint Set Data Structures. Jun 2020, [cit. 2022-01-05]. Dostupné z: https://www.geeksforgeeks.org/disjoint-set-data-structures/

[51] Bourke, P.: Minimum Distance between a Point and a Line. Oct 1988, [cit. 2022-01-05]. Dostupné z: http://paulbourke.net/geometry/pointlineplane/

[52] Aleš Hrabalík, D. R.: FMO dataset. Cit. 2021-05-10. Dostupné z: http://cmp.felk.cvut.cz/fmo/

Příloha **A**

Seznam použitých zkratek

CNN Convolutional neural network

FFNN Feed-Forward neural network

TLDA Two Layered Data Associaton - metoda pro hledání trajektorie z detekcí objektu

horizontální čáry čáry kurtu kolmé na síť

vertikální čáry čáry kurtu rovnoběžné se sítí

YOLOv3 konvoluční neuronová síť YOLOv3

TrackNet konvoluční neuronová síť TrackNet

RGB barevný formát RGB

HSV barevný formát HSV

Příloha B

Seznam klíčových pojmů

Tracknet Konvoluční neuronová síť detekující tenisový míček

událost událost v tenisové výměně, například dopad, úder apod.

Vertikální čáry/lajny čáry tenisového hřiště kolmé na síť

Horizontální čáry/lajny čáry tenisového hřiště rovnoběžně se sítí

horní část hřiště horní část hřiště dělená polovinou kurtu

dolní část hříště dolní část hřiště dělená polovinou kurtu

CNN Convolutional neural network

FFNN Feed-Forward neural network

TLDA Two Layered Data Associaton - metoda pro hledání trajektorie z detekcí objektu

YOLOv3 konvoluční neuronová síť YOLOv3

TrackNet konvoluční neuronová síť TrackNet

RGB barevný formát RGB

HSV barevný formát HSV

souřadnice y vertikální směr na snímku

negativní směr v souřadnici y vertikální směr nahoru vzhledem k snímku

pozitivní směr v souřadnici y vertikální směr dolů vzhledem k snímku

Příloha C

Obsah přiloženého CD

```
readme.txt ..... stručný popis obsahu CD
src
    impl ..... zdrojové kódy implementace
    thesis ..... zdrojová forma práce ve formátu LaTeX
text ..... text práce
dataset ..... vytvořená datová sada
vizualizace ..... ukázka vizualizace
```