



# MATEMATICKO-FYZIKÁLNÍ FAKULTA Univerzita Karlova

## BAKALÁŘSKÁ PRÁCE

Matěj Ščerba

### Analýza videozáznamu skoku o tyči

Katedra softwarového inženýrství

Vedoucí bakalářské práce: Ing. Michal Bartoš, Ph.D.

Studijní program: Informatika (B1801)

Studijní obor: IOI (1801R008)

Praha 2021

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne .....  
Podpis autora

Poděkování.

Název práce: Analýza videozáznamu skoku o tyči

Autor: Matěj Ščerba

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: Ing. Michal Bartoš, Ph.D., Katedra softwarového inženýrství

**Abstrakt:** Skok o tyči je jedna z technicky náročnějších lehkoatletických disciplín, proto se analýzou skoků na většině mezinárodních závodů zabývají vědci zkoumající pohyb lidského těla. Pro trenéry a závodníky by bylo vhodné, aby měli k dispozici podobné prostředky i při trénincích, aniž by museli investovat do specializované techniky. Má práce tuto problematiku řeší. S použitím počítače a kamery lze z jediného videozáznamu získat model atleta a zároveň analyzovat parametry jím provedeného skoku. Výsledné parametry se posléze zanesou do grafů a jejich hodnoty lze zkoumat i v různých fázích skoku za použití prohlížeče videozáznamu.

**Klíčová slova:** skok o tyči, detekce pozice těla, videozáznam

Title: Analysis of pole-vault video recording

Author: Matěj Ščerba

Department: Department of Software Engineering

Supervisor: Ing. Michal Bartoš, Ph.D., Department of Software Engineering

Abstract: Abstract.

Keywords: pole vault, human pose detection, video recording

# Obsah

<b>Úvod</b>	<b>3</b>
Související práce . . . . .	4
<b>1 Teorie</b>	<b>5</b>
1.1 Skok o tyči . . . . .	5
1.1.1 Analýza technického provedení skoku . . . . .	5
1.1.2 Analýza biomechanických parametrů skoku . . . . .	6
1.2 Zpracování obrazu . . . . .	8
1.2.1 Detekce člověka . . . . .	8
1.2.2 Trasování objektů . . . . .	10
1.2.3 Odhad pozice těla . . . . .	11
<b>2 Návrh řešení</b>	<b>13</b>
2.1 Souřadné systémy . . . . .	13
2.1.1 Snímek . . . . .	13
2.1.2 Video . . . . .	13
2.1.3 Prostor . . . . .	13
2.2 Vstup programu . . . . .	15
2.3 Analýza videa . . . . .	15
2.3.1 Zpracování videa . . . . .	15
2.3.2 Nalezení atleta . . . . .	15
2.3.3 Detekce kostry . . . . .	16
2.3.4 Konverze kostry do 3D modelu . . . . .	20
2.3.5 Analýza parametrů . . . . .	20
2.4 Analýza modelu . . . . .	22
2.5 Výstup programu . . . . .	23
2.5.1 Uložení výstupu . . . . .	23
2.5.2 Zobrazení výstupu uživateli . . . . .	24
2.6 Programové vybavení . . . . .	26
2.6.1 Počítačové vidění a zpracování obrazu . . . . .	26
2.6.2 Grafická reprezentace výstupu . . . . .	26
<b>3 Realizace</b>	<b>28</b>
3.1 Zvolené prostředky . . . . .	28
3.2 Konverze souřadných systémů . . . . .	28
3.3 Struktura programu . . . . .	29
3.4 Vstup programu . . . . .	29
3.5 Analýza videa . . . . .	29
3.5.1 Zpracování videa . . . . .	29
3.5.2 Nalezení atleta . . . . .	30
3.5.3 Detekce kostry . . . . .	33
3.5.4 Konverze kostry do 3D modelu . . . . .	37
3.5.5 Analýza parametrů . . . . .	37
3.6 Analýza modelu . . . . .	40
3.7 Výstup programu . . . . .	41

3.7.1	Uložení výstupu . . . . .	41
3.7.2	Zobrazení výstupu uživateli . . . . .	41
<b>4</b>	<b>Experiments</b>	<b>43</b>
4.1	Sběr dat . . . . .	43
4.2	Hledání atleta ve videu . . . . .	43
4.3	Volba úvodní pozice atleta . . . . .	45
4.3.1	Pozice ohraničujícího rámečku . . . . .	45
4.3.2	Velikost ohraničujícího rámečku . . . . .	45
4.4	Pohyb kamery . . . . .	46
4.4.1	Vertikální pohyb kamery . . . . .	46
4.4.2	Rotace kamery . . . . .	46
4.5	Pozice kamery . . . . .	47
4.5.1	Směr rozběhu . . . . .	47
4.5.2	Úhel záběru . . . . .	48
4.6	Další postavy ve videu . . . . .	48
4.7	Splynutí s pozadím . . . . .	48
4.8	Frekvence snímků . . . . .	50
4.9	Shrnutí . . . . .	50
4.9.1	Navrhovaná vylepšení . . . . .	51
<b>Závěr</b>		<b>54</b>
<b>Seznam použité literatury</b>		<b>55</b>
<b>Seznam obrázků</b>		<b>58</b>
<b>A Přílohy</b>		<b>59</b>
A.1	První příloha . . . . .	59

# Úvod

Při trénincích a soutěžích ve skoku o tyči využívají atleti spolu s trenéry videozáznamy provedených skoků. Videozáznam poté využívají pro detekci chyb v technickém provedení skoku a jejich případnému vlivu na výsledný výkon. Pro výkon při skoku o tyči jsou kromě technického provedení důležité také hodnoty biomechanické parametry skoku. Mezi tyto parametry patří například rychlosť atleta při rozběhu.

V tréninkovém procesu je důležité sledovat výkonnost atleta. K tomuto účelu slouží především měření času běhu různých vzdáleností. Pro skok o tyči je vhodné aplikovat podobné metody právě při skokanských trénincích. Jednou z možností, jak toho docílit, je použití specializované techniky pro měření rychlosti atleta při rozběhu. Mezi specializovanou techniku lze zařadit radary nebo fotobuňky. Za použití vhodně rozmištěných fotobuněk lze poměrně přesně určit rychlosť v různých částech rozběhu. Využití těchto metod je finančně náročné a jediným parametrem, který lze sledovat, je pouze rychlosť atleta při rozběhu.

S rozvojem počítačového vidění přibývá na světových atletických soutěžích studií zabývajících se pohyby sportovců (Athletics, 2021). Výsledky těchto studií jsou pro atlety a jejich trenéry vynikající zpětnou vazbou. Tyto studie využívají několik kamer k zachycení pohybu atleta, kamery nejprve kalibrují za použití přesně změřených konstrukcí s vyznačenými body. Pro analýzu pohybu poté využívají 3D model atleta, který z videí získají. Výsledný model zachycuje polohu kloubů těla a z ní vyplývající pozice jednotlivých segmentů těla, které umožňují analyzovat pohyb celého těla. Takovýto model je možné použít k analýze mnohem více parametrů skoku o tyči než jen rychlosť běhu atleta.

Studie se většinou zabývají porovnáváním pohybu různých atletů na základě získaných parametrů. V tréninkovém prostředí je zvykem využívat pouze videozáznamy a případné porovnání skoků probíhá spuštěním záznamů, které dané skoky zachycují, následným rozbořením techniky jednotlivých skoků a hledáním případných rozdílů. Tato forma porovnání skoků ovšem nebude v potaz biomechanické aspekty pohybu atleta, které mohou techniku skoku ovlivnit. Hodnoty parametrů, kterými se specializované studie zabývají, lze z videa odhadnout pro účely porovnání skoků, ale ani zkušení trenéři nejsou schopni přesně určit hodnoty některých parametrů.

Podobné studie pohybu jsou užitečným tréninkovým prostředkem, ovšem jejich využití byt jen v malém počtu tréninků se nejeví jako reálné. Dostupnějším přístupem k tomuto problému by byla aplikace, běžící na mobilním zařízení, která by analyzovala pohyb atleta na základě 3D modelu získaného z jediného videa. Výsledkem této analýzy by byly graficky znázorněny parametry skoku, které by se daly využít pro porovnání různých skoků. Takováto aplikace by výrazně přispěla k přesnější analýze tréninkových a závodních skoků mimo soutěže světové úrovni a pomohla by atletům lépe sledovat vlastní progres z fyzické stránky.

Cílem mé práce je základní funkcionality, kterou by takováto aplikace disponovala. Jedná se konkrétně o extrakci modelu atleta z jediného videozáznamu, detekci důležitých momentů skoku, mezi něž patří začátek rozběhu, odraz a kulminace nad laťkou. Výsledný model poté analyzuji pro zisk hodnot biomechanických parametrů, které je při skoku vhodné zkoumat, tyto parametry poté zobrazím

uživateli.

Text je rozdělen do čtyř kapitol. První kapitola se věnuje teorii skoku o tyči a počítáčovému vidění. Ve druhé kapitole popisují obecný návrh mého řešení. Třetí kapitola popisuje konkrétní implementaci a ve čtvrté, závěrečné, kapitole se věnuji experimentům, které popisují funkcionality programu.

## Související práce

Obecnou analýzou pohybu postav se zabývá práce Human motion analysis: a review (Aggarwal a Cai, 1997), implementaci analýzy pohybu při chůzi popisuje článek The model-based human body motion analysis system (Chang a Huang, 2000), detekci pozice těla se věnují projekty Cao a kol. (2021), Sun a kol. (2019), Pishchulin a kol. (2016), Toshev a Szegedy (2014), a Fang a kol. (2018).

Extrakcí a analýzou 3D modelu těla z videa se zabývá společnost MobiDev (Tatarants, 2021), extrakcí 3D modelu postavy z několika videí se zabývá aplikace Simi Motion (Simi Reality Motion Systems GmbH (2014)),

# 1. Teorie

## 1.1 Skok o tyči

Skok o tyči je jedna z technicky náročných atletických disciplín. Atlet se nejprve s tyčí rozběhne po rozběžišti, následně zasune tyč do zasouvací skříňky, poté se odrazí, provede skok a dopadne do doskočiště. Zasouvací skříňka je místo v zemi, kam atlet při rozběhu zasune tyč, aby měl při skoku stabilní oporu.

Atlet se při skoku pohybuje vzhůru díky energii, kterou přenesl z rozběhu do tyče. Kinetickou energii získanou v průběhu rozběhu převede v momentu odrazu do tyče, čímž vytvoří potenciální elastickou energii. Jakmile hodnota potenciální energie převýší hodnotu kinetické a gravitační energie atleta, začne se převádět v kinetickou energii, která se projeví katapultací atleta směrem vzhůru. Pro ideální směr katapultace je potřeba, aby byl poměr kinetické energie rozběhu a elastické energie tyče takový, že výsledný směr katapultace bude přes latku. Atlet může při skoku tento směr, a tedy i pozici svého těžiště, ovlivnit i po odrazu, což je jedinečnou vlastností této atletické disciplíny.

Pro pozici těla atleta při rozběhové fázi je typický značně omezený pohyb paží a mírné natočení trupu. Důvodem tohoto nestandardního způsobu běhu je skutečnost, že atlet nese tyč, a přesto se snaží vyvinout maximální kontrolovanou rychlosť. Náběhovou rychlostí se rozumí atletova rychlosť před odrazem. Při skoku o tyči (a všech ostatních skokanských disciplínách) má atlet vyznačené místo, ve kterém začíná svůj pokus. Toto místo si volí sám, vliv na jeho polohu má především fyzická zdatnost atleta a počet kroků rozběhu. Počet kroků závodního rozběhu atletů světové úrovni se pohybuje v rozmezí 16 až 20 kroků, tedy přibližně 35 až 45 metrů od zadní hrany zasouvací skříňky. Atlet se rozbíhá s tyčí ve vzduchu, postupně ji spouští, dokud není rovnoběžně se zemí. Následně provádí zásun - pohyb, při němž zasune tyč do zasouvací skříňky a dostane paže nad hlavu. V pozici s pažemi nad hlavou se atlet odráží a přenáší energii do tyče. Na charakter skoku má vliv také místo odrazu. Následný skok lze rozdělit do několika částí.

První z nich je odraz, při němž je kladen důraz na efektivitu přenosu energie získané při rozběhu do tyče. Po této krátké fázi skoku následuje zvrat. Jedná se o pohyb způsobený švihnutím odrazové nohy a paže, která se tyče drží výše, směrem k sobě, tedy dopředu. Po provedení zvratu se atlet dostane do pozice vzhůru nohama. V této pozici atlet provádí obrat, při němž se otáčí o  $180^\circ$  podél osy svého těla, aby byl čelem k latce. Následuje odraz od tyče a přechod latky, v této fázi se atlet snaží dostat boky co nejvýš a zajít tak plynulý skok přes latku. Latku atlet překonává nohama napřed, čelem k latce.

### 1.1.1 Analýza technického provedení skoku

Způsob provedení skoku je podstatným ukazatelem pro výsledný výkon. Jen drobná změna v jediné fázi pokusu může dramaticky ovlivnit charakter a výšku celého skoku. Tím pádem mají skokané o tyči pro nácvik techniky vyhrazeno několik tréninků týdně. Jedná se o skokanské tréninky a tréninky zabývající se nácvikem techniky prostřednictvím gymnastických prvků simulujících pohyb atleta na tyči. Pro mou práci jsou důležitější skokanské tréninky.

Skokanské tréninky začínají podobně, jako ostatní atletické tréninky, tedy rozcvičením. Po klasickém rozcvičení následuje příprava na samotné skákání, tato příprava se u jednotlivých atletů může lišit. Mnoho atletů před samotným skákáním provede několik cvičných zásunů a rozběhů s tyčí. Následují skoky z krátkého rozběhu, standardně se jedná o 6 až 8 kroků, některí atleti provádějí tyto skoky bez ohybání tyče. Již na základě těchto cviků lze určit, na jaké prvky skoku by se měl atlet zaměřit. Po této fázi následuje skákání z dlouhého rozběhu. Počet kroků se liší podle fáze sezony, ve které se atlet nachází. V závodním období jsou typické rozběhy delší, v přípravném období kratší, jelikož tréninky v přípravném období jsou zpravidla fyzicky náročnější.

Při tréninku se opakuje následující situace. Atlet provede skok a následně ho konzultuje s trenérem. Předmětem konzultace je především technické provedení skoku. S rozvojem moderních technologií jsou konzultace ve většině případů doprovázeny sledováním a rozborem videozáznamu zahycujícího právě provedený skok. S pomocí tohoto videozáznamu lze přesně určit místo odrazu a pohyb atleta při rozběhu a následném skoku. Na pořízeném videozáznamu lze spolehlivě detektovat technické nedostatky skoků, které by se atlet měl v následujících pokusech snažit eliminovat.

Videozánam je nejčastěji pořizován na mobilní telefon, případně tablet. Pozice kamery se nejčastěji nachází na kolmici k rozběžišti, která prochází místem odrazu. Z tohoto místa je poměrně dobře vidět jak rozběh, tak skok. Navíc se takto s velikou přenosností dá určit místo odrazu a pozice atletova těla při přenosu energie do tyče při odrazu. Strana, ze které je skok natočen, se často mění, záleží na prvku skoku, který má trenér s atletem v plánu zkoumat.

Závody probíhají podobně, po každém skoku opět dochází ke konzultaci, ovšem na mezinárodních závodech není běžné, že by atlet viděl záznam skoku, který trenér pořídil, na většině soutěží je to zakázané pravidly. Trenér tedy pouze popisuje nedostatky skoku a probíhá diskuse s atletem zabývající se následujícím postupem. Při závodech není kladen takový důraz na změny v technice jako při tréninku, typicky probíhá jen rozbor detailů nebo posouvání místa odrazu pro optimální přenos energie do tyče.

### 1.1.2 Analýza biomechanických parametrů skoku

S rozvojem moderní techniky přibývá studií zabývajících se pohybem těla atleta při skoku o tyči. Podkladem pro tyto studie jsou především mezinárodní závody, případně mistrovství republiky. Příkladem je biomechanická zpráva z finále mužů Mistrovství světa v atletice 2017 (Gravestock a kol., 2017), na níž se podíleli pracovníci Leeds Beckett University.

K analýze parametrů se využívá několik kamer, jejichž záznam slouží jako podklad k vytvoření modelu atleta v průběhu celého skoku. Tento model je následně analyzován pro zisk konkrétních parametrů skoku. Podobné studie se nejčastěji zabývají následujícími parametry:

#### Délka jednotlivých kroků rozběhu

Na efektivitu přenosu energie do tyče má vliv délka kroků na konci rozběhu. U většiny atletů je poslední krok rozběhu kratší oproti předešlým krokům. Rozdíl v délce předposledního a posledního kroku se může pohybovat v řádech desítek

centimetrů. Důvodem pro zkrácení posledního kroku je sešlápnutí posledního kroku pod tělo atleta, tím atlet ztratí méně horizontální rychlosti při odrazu. Zkrácení posledního kroku doprovází odraz pod menším úhlem, což pro je skok o tyči do jisté míry vhodné.

### Doba oporové fáze kroku

Technika běhu je značně ovlivněna dobou oporové fáze jednotlivých kroků. Doba oporové fáze souvisí s rychlostí běhu, mírou pokrčení stojné nohy a výškou boků a kolen. Z prostého videozáznamu se složitě určuje doba oporové fáze kroku, a tudíž se při této analýze věnuje zvýšená pozornost právě míře pokrčení stojné nohy a výše boků a kolen.

Doba oporové fáze kroku a délka kroku se kromě analýzy videozáznamu dá měřit speciálními pásy, které se položí na zem po obou stranách rozběžště.

### Náběhová rychlosť

Náběhová rychlosť má značný vliv na přenos energie rozběhu do tyče, rychlejší atleti tedy zpravidla používají tvrdší a delší tyče, které umožňují vyšší skoky.

Náběhovou rychlosť lze měřit několika způsoby. Pro tréninkové účely se nejčastěji používají rovnoměrně rozmístěné fotobuňky, z doby běhu mezi nimi lze snadno vypočítat průměrnou rychlosť v daném úseku. Tuto metodu používají především sprinteré, pro skok o tyči není příliš vhodná, rychlosť se v momentu odrazu dramaticky mění a tyč může ovlivnit detekci fotobuněk. Následující možností je použití radaru, s kterým lze zanést aktuální rychlosť do grafu. Při snímání atleta ze zadu téměř nedochází k chybám měření. Způsob měření náběhové rychlosti použitý ve výše zmíněné studii (Gravestock a kol., 2017) spočívá v analýze detekovaného modelu v souřadném systému, který umožňuje výpočet rychlosti jednotlivých částí těla ve standardních jednotkách, typicky se uvádí v  $\text{ms}^{-1}$ .

### Výška boků v průběhu rozběhu

V průběhu rozběhu je zajímavé pozorovat výšku boků, výrazné výkyvy výšky boků mají za následek výkyvy rychlosti. Plynulost rozběhu tedy koreluje s plynulostí výšky boků v průběhu rozběhu. Aby se atletovo těžiště pohybovalo po osě rozběžště a neztrácela se tak energie je důležité, aby se atlet rozeběhl kontrolovaně, což se může projevit ve výšce boků, výkyvy ve výšce boků také souvisí s proměnlivou délkou kroků a dobou oporové fáze kroku. Nejdůležitější moment pro zkoumání výšky boků jsou poslední kroky rozběhu.

### Místo odrazu

Vzdálenost místa odrazu od zadní hrany zasouvací skříňky je podstatným ukazatelem efektivity přenosu energie do tyče. Je vhodné, aby měl atlet v místě odrazu horní paži přímo nad sebou. Vhodná vzdálenost místa odrazu je individuální, některí atleti preferují odraz blíže k zasouvací skřínce než jiní. Pozice odrazu se odvíjí od technického provedení skoku, výšky a odrazových schopností atleta.

## Výška úchopu

Hodnota tohoto parametru souvisí s délkou tyče a odvíjí se od ní místo odrazu. Může se uvádět jako vzdálenost úchopu horní ruky od spodního konce tyče nebo jako výška úchopu horní ruky nad zemí, když je tyč svislá v zasouvací skříňce. Zasouvací skříňka je hluboká 20 cm, tedy hodnota po měření první metodou je o 20 cm vyšší než s použitím druhé metody. Hodnota tohoto parametru má vliv na dobu trvání skoku, vyšší úchup znamená delší dobu trvání skoku.

## Úhel odrazu

Pro správný převod energie rozběhu do tyče je užitečné zkoumat úhel odrazu. Uvádí se jako úhel mezi zemí a směrem, kterým se po odrazu pohybuje těžiště. Větší úhel znamená vyšší odraz, ale také větší ztrátu rychlosti při odrazu. Optimální hodnota úhlu odrazu různých atletů se liší například v závislosti na jejich tělesné výšce.

## Úhly v kloubech při odrazu

Pozice jednotlivých segmentů těla při odrazu je pro přenos energie do tyče důležitá. Pokud je atletův trup nakloněn dopředu, jeho ramena a paže jsou v lepší pozici pro roztačení tyče. Naopak pokud je atlet v záklonu, jeho boky po odrazu snáze ujedou směrem dopředu a ramena zůstanou vzadu. To má za důsledek horší roztačení tyče.

## Doba trvání skoku

Doba trvání skoku je vymezena odrazem od země a opuštěním tyče. Na hodnotu tohoto parametru má vliv mnoho faktorů. Technika je jedním z nich, standardně se atleti odráží od tyče vzpaženou paží, jsou ovšem atleti, kteří se tyče pustí dříve. Doba skoku za použití klasického odražení se vzpaženou paží se pohybuje okolo 1,3 s, nestandardní technika může tuto hodnotu stlačit až k 0,7 s. Tyto hodnoty se týkají mužského finále z Mistrovství světa v atletice 2017. Dále má na délku skoku vliv tvrdost a délka tyče, tvrdší tyč se snáze - a tedy i rychleji - narovná a skok tak trvá kratší dobu. S použitím kratší tyče atletovo tělo překoná kratší vzdálenost a doba trvání skoku je tedy kratší.

## Převýšení

S opuštěním tyče souvisí i převýšení. Jedná se o rozdíl zdolané výšky a výšky úchopu nad zemí. Atleti světové úrovni převýšují okolo 130 cm, ženy okolo 70 cm. Převýšení je jedním z hlavních parametrů, které určují výkonnost atleta. Na jeho hodnotu má vliv jak fyzická zdatnost, tak technické provedení skoku.

## 1.2 Zpracování obrazu

### 1.2.1 Detekce člověka

Postup detekce konkrétního objektu ve snímku se skládá ze dvou kroků. Prvním je zvolení vhodné reprezentace snímku (extrakce jeho vlastností), druhým krokem

je spuštění klasifikátoru, který určí, zda (a kde) se ve snímku hledaný objekt nachází.

Pro účely detekce se využívá klasifikační úloha, která dostane na vstupu část snímku. Poté určí, zda se v tomto okně nachází hledaný objekt, pokud tomu tak je, určí se jeho poloha podle pozice zpracovávaného okna ve snímku. Detekce se tedy spouští na několika vstupech, což jsou různé části snímku. Aby bylo možné detekovat různé velké postavy, využívá se zmenšování a zvětšování snímku. Velikost oken v pixelech se nemění, aby bylo možné spustit detekci na všech oknech.

## Extrakce vlastností snímku

Velice úspěšným způsobem extrakce vlastností okna pro detekci postav je určení histogramu orientovaných gradientů (Dalal a Triggs, 2005). Tato metoda spočívá v rozdelení okna na části (buňky), v nichž se určí histogram, popisující směr gradientů - a tedy i směr hran. Jelikož gradient závisí na kontrastu snímku, části snímku je tedy vhodné normalizovat, co se týče kontrastu, aby byly jejich gradienty porovnatelné. Typicky se normalizují skupiny buněk (bloky), ne jednotlivé buňky oken. Získané hodnoty z takto normalizovaných bloků, které pokrývají okno a z části se překrývají, reprezentují vlastnosti daného okna, které se následně předají klasifikátoru.

Výsledné vlastnosti jsou invariantní k lokálním geometrickým transformacím, mezi které patří posuny, škálování a rotace. Tyto transformace mají na výslednou reprezentaci objektu vliv v případě, že jsou příliš velké. Díky této vlastnosti je detekce člověka úspěšná, pokud se nachází ve víceméně vzpřímené pozici. O předpoklad vzpřímené pozice těla se opírá většina detektorů.

Extrakce těchto vlastností dosahuje vysoké úspěšnosti při použití přístupu Scale-Invariant Feature Transform (Lindeberg, 2012). Tento přístup představuje invariantní vlastnosti vzhledem ke škálování a rotaci.

## Klasifikace

Při využití histogramu orientovaných gradientů se nejčastěji využívá lineární Support vector machines (SVM) (Cortes a Vapnik, 1995). SVM se používá k binární klasifikaci.

Základ metody SVM spočívá v rozdelení prostoru na dva poloprostory s využitím nadroviny. Při trénování mám daná data rozdelená do dvou tříd. Lineární SVM předpokládá lineárně separabilní data, tedy je lze oddělit nadrovinou. Nelineární SVM využívá k dosažení stejného cíle prostorové transformace. Při trénování hledám nadrovinu, která data odděluje a vzdálenost nadroviny od nejbližšího bodu z trénovacích dat je maximální možná.

Každý bod v prostoru odpovídá příslušné kombinaci vlastností vstupu. Podle vlastností vstupu určím pozici v prostoru a podle poloprostoru, do kterého spadá, vstup klasifikuju.

V mé případě určím vlastnosti vstupního snímku, na základě těchto vlastností určím poloprostor, do něhož snímek spadá a následně se rozhodnu, zda je na obrázku postava či nikoliv.

## 1.2.2 Trasování objektů

Trasování objektů ve videu je metoda, díky níž lze na základě pozice objektu ve snímku určit jeho pozici v následujícím snímku. Tímto postupem lze objekt sledovat v průběhu celého videa. Pro lepší přesnost detekce je vhodné stav objektu postupně aktualizovat, jelikož se v průběhu videa může menit. Algoritmům, které trasování objektů zajišťují, se říká trackery.

Přístupů k řešení tohoto problému je několik, lze je rozdělit do následujících kategorií:

- feature based,
- segmentation based,
- estimation based a
- learning based.

Trasovací metody lze rozdělit také podle jiných kritérií, tato kritéria použil Soleimanitaleb a kol. (2019) příkladem je rozdělení pouze do 2 kategorií - trackery využívající korelační filtr a ostatní trackery (Fiaz a kol., 2018).

### Feature based

Tento přístup spočívá v extrakci vlastností objektu. Mezi tyto vlastnosti se řadí například barva nebo textura. Na základě extrahovaných vlastností se v následujícím snímku najde objekt, který má tyto vlastnosti nejpodobnější a splňuje další podmínky, mezi které lze zařadit vzdálenost nalezeného objektu od jeho pozice v předchozím snímku.

### Segmentation based

Trasování založené na principu segmentace odděluje objekty od pozadí. Typicky je trasován objekt v popředí videa, tudíž oddelení všech objektů v popředí od pozadí zjednoduší problém trasování.

### Estimation based

Metody založené na odhadech se opírají o reprezentaci objektu stavovým vektorem. Tento vektor popisuje směr pohybu objektu, který se neustále upravuje. Trasování tímto způsobem lze rozdělit do dvou kroků - predikce a aktualizace.

Predikce využívá stavový vektor k určení pozice objektu v následujícím snímku a aktualizační část predikovanou pozici upraví na základě analýzy této části snímku podobným způsobem, jako při použití feature based trackeru.

### Learning based

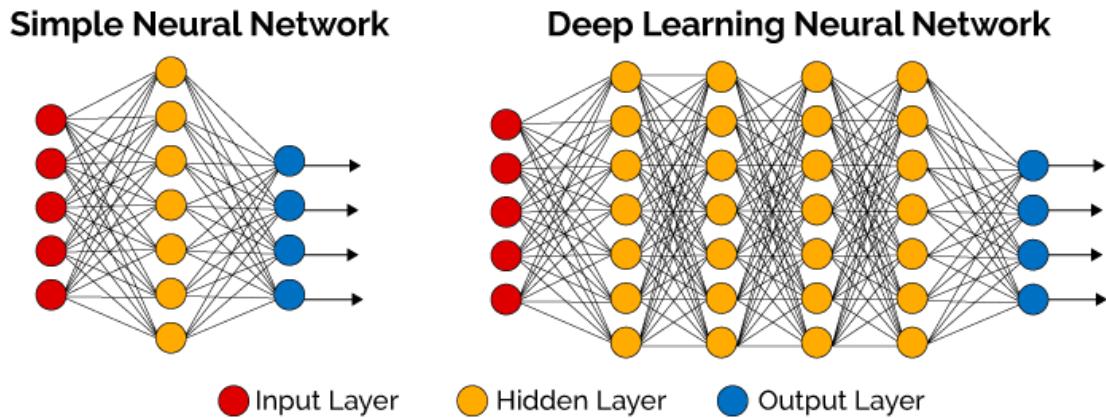
Tento přístup zahrnuje metody využití strojového učení. K detekci objektů v následujícím snímku je potřeba extraovat ze snímku vlastnosti a na jejich základě trénovat tracker.

Využití konvolučních sítí dosahuje výborných výsledků, ve výsledcích soutěže The Visual Object Tracking VOT2017 challenge (Kristan a kol., 2017) byly mezi

nejpřesnějšími algoritmy využívající porovnávání tvarů za použití konvolučních sítí (CNN matching). Nejpřesnější trackery využívají korelační filtry.

### 1.2.3 Odhad pozice těla

Odhad pozice těla spočívá v detekci kloubů těla, k této detekci se využívá především hluboké učení (Toshev a Szegedy, 2014). Hluboké učení využívá rozsáhlé neuronové sítě, které se skládají z několika vrstev. Grafické znázornění sítí je vidět na obrázku 1.1.



Obrázek 1.1: Jednoduchá a hluboká neuronová síť. Jednotlivá kolečka reprezentují neurony, úsečky mezi nimi přechody mezi neurony (váhy). Převzato z <https://becominghuman.ai/deep-learning-made-easy-with-deep-cognition-403fbe445351> dne 26. 5. 2021.

První vrstva bývá označovaná jako vstupní, poslední jako výstupní. Vrstvy, které se nachází mezi nimi jsou označovány jako skryté. Každou vrstvu tvoří neurony, které při zpracování vstupu nabývají číselných hodnot. Přechody mezi vrstvami definují váhy, kterými se hodnoty příslušných neuronů násobí. Většina vrstev využívá aktivační funkci, která hodnoty neuronů upravuje než je předá další vrstvě. Aktivační funkce jsou typicky nelineární, mezi nejpoužívanější se řadí ReLU a tanh, která vznikla modifikací funkce sigmoid, jež se využívala v minulosti a slouží k binární klasifikaci na výstupní vrstvě. Výstupem sítě jsou hodnoty neuronů výstupní vrstvy, které nejčastěji reprezentují pravděpodobnost, že vstup spadá do třídy, kterou daný neuron reprezentuje. Aby se jednalo o pravděpodobnostní distribuci, musí se hodnoty neuronů normovat. K tomu slouží funkce softmax, což je rozšíření funkce sigmoid pro klasifikaci do několika tříd. Definice popsaných aktivačních funkcí jsou následující:

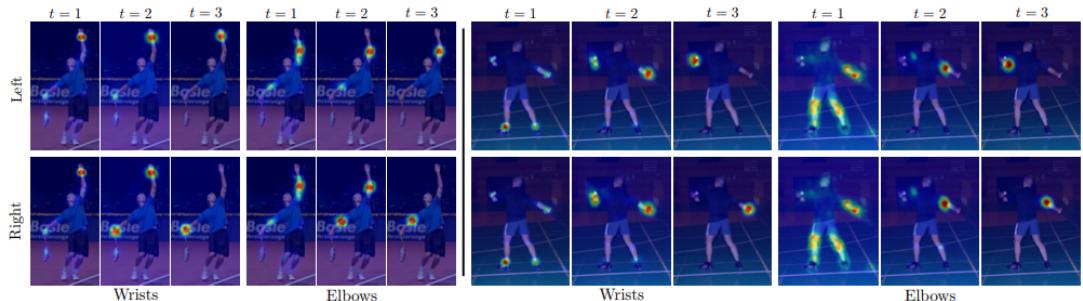
$$\begin{aligned} \text{ReLU:} & \quad \text{ReLU}(x) = \max(0, x), \\ \text{sigmoid:} & \quad \sigma(x) = \frac{1}{1 + e^{-x}}, \\ \text{tanh:} & \quad \tanh(x) = 2\sigma(2x) - 1, \\ \text{softmax:} & \quad \text{softmax}(\mathbf{x}) = \frac{\mathbf{x}}{\sum_i e^{\mathbf{x}_i}}. \end{aligned}$$

Podrobnější analýzou aktivačních funkcí se zabývá Szandała (2020).

Existují různé typy neuronových sítí podle typu vrstev, které obsahují. První síť, které se využívaly, obsahovaly vrstvy, které byly kompletně propojené. Hodnoty všech neuronů dané vrstvy v takové síti závisí na hodnotách všech neuronů předchozí vrstvy. Tyto síť není příliš vhodné používat pro detekci objektů ve snímcích, jelikož závisí na mnoha vlastnostech množiny dat, na kterých jsou natrénovány. Hlavní nevýhoda spočívá v tom, že tato síť není invariantní k posunu objektu. Pokud je síť natrénovaná na obrázcích krajiny, kde se slunce nachází v levém horním rohu, má poté problém detektovat slunce v pravém horním rohu. Pixely, které reprezentují slunce se totiž nachází na jiném místě ve vstupní vrstvě a jejich hodnoty využívají jiné váhy.

Řešení tohoto problému nabízí konvoluční síť. Ty využívají konvoluční filtry, které jsou specifické pro každou vrstvu. Takový filtr má danou velikost, ze které bere vstupní hodnoty a vypočítá hodnotu výslednou. Jedná se o jakési okno, které se aplikuje na každou část vstupní vrstvy a určí hodnotu konvoluce v této části. Hodnota neuronu vrstvy tedy závisí jen na blízkém okolí v předchozí vrstvě. Jelikož se hodnoty filtru při zpracování dané vrstvy nemění, je výsledná hodnota po aplikaci filtru na části obsahující slunce v levém horním rohu snímku stejná, jako výsledná hodnota po aplikaci na část obsahující slunce v pravém horním rohu snímku. Více konvolučních vrstev dokáže popsat větší část snímku, jejíž velikost závisí na velikostech použitých filtrů.

Pro využití detekce kloubů těla se nejčastěji používají právě konvoluční sítě, jejich výstupem je ve většině případů mapa pravděpodobnosti, znázorňující pravděpodobnost výskytu hledaného kloubu různých místech vstupního snímku. Příklad takových map lze vidět na obrázku 1.2. Pravděpodobnostní mapy využívají i některé trackery (Lukežič a kol., 2018).



Obrázek 1.2: Pravděpodobnostní mapy detekce různých kloubů těla.

## 2. Návrh řešení

V této kapitole stručně popíšu návrh mého programu, který má za úkol extrakci modelu reprezentujícího pohyb atleta při skoku o tyči. Skok je programu předán ve formě videa, po extrakci modelu program detekuje důležité momenty skoku a analyzuje vhodné biomechanické parametry. Hodnoty těchto parametrů následně zobrazí uživateli v grafické podobě a také zobrazí snímky zachycující důležité momenty skoku.

Program lze spustit ve dvou módech - analýza videa a analýza modelu.

Při analýze videa dostane program na vstupu video, ve videu najde atleta a s využitím jeho pozice detekuje jeho kostru v jednotlivých snímcích. Z detekované kostry vytvoří 3D model atletova těla, který reprezentuje atletův pohyb.

Analýza pohybu modelu probíhá stejně při zpracování videa i samotného modelu. Podle pohybu jednotlivých částí těla atleta se vypočítají hodnoty biomechanických parametrů, které se uloží a zobrazí uživateli.

### 2.1 Souřadné systémy

Pro reprezentaci pozic objektů je potřeba zvolit vhodné souřadné systémy, aby nezkreslovaly hodnoty parametrů.

#### 2.1.1 Snímek

Souřadný systém ve snímku má počátek v levém horním rohu. Bod  $(x,y)$  je ve vzdálenosti  $x$  pixelů od levé strany snímku a  $y$  pixelů od horní strany snímku.

#### 2.1.2 Video

Kamera typicky není při natáčení skoků statická, tudíž je potřeba reprezentovat pohyb objektů jinak než pouze jejich pozici ve snímcích videa.

Rozhodl jsem se pro určení pohybu podle vzájemné pozice objektu a pozadí videa v souřadném systému snímku. Polohu pozadí považuji za statickou. Jedná se o jistou formu projekce 3D prostoru do 2D, která zkresluje realitu, tento systém lze reprezentovat jako souřadnice v panoramatické fotce.

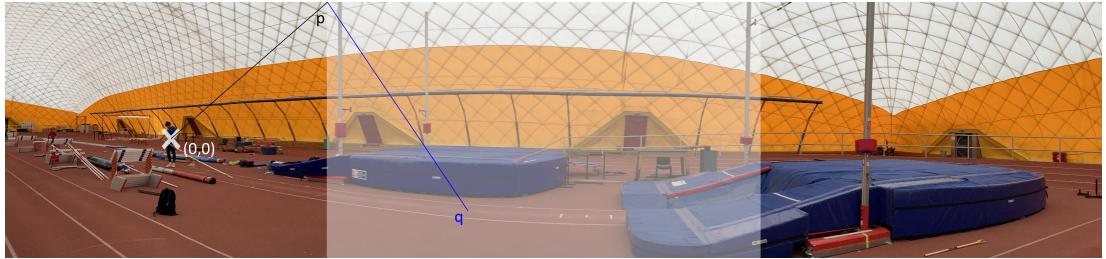
Pozice objektu v prvním snímku, v němž ho detekuji, je počátkem souřadného systému. Nejprve určím pozici levého horního rohu snímku, v němž se nachází objekt, jehož pozici chci určit. Tuto pozici určím vzhledem k pozici pozadí. Společně s pozicí objektu v tomto snímku získám pozici objektu vůči jeho pozici ve snímku, v němž jsem ho detekoval poprvé.

Souřadný systém videa znázorňuje obrázek ??.

#### 2.1.3 Prostor

Model atleta umístím do 3D prostoru, který se podobá reálnému světu, ale je založen na souřadném systému videa, tudíž realitu do jisté míry zkresluje.

Odhad reálných vzdáleností z videa je poměrně komplikovaný a pro základní parametry skoku bude tento systém dostačující. Pro případnou konverzi na metry



Obrázek 2.1: Vizualizace souřadného systému videa. Světlé pole reprezentuje snímek videa,  $p$  posun jeho horního rohu vůči původní pozici atleta,  $q$  bod, jehož pozici ve videu získám pomocí posunu snímku a pozice bodu  $q$  ve snímku.

lze systém vhodně přeškálovat v budoucnu, zatím vzdálenosti odpovídají vzdálenostem ve snímku videa, které závisí na jeho rozlišení. Musí se ovšem brát v potaz vzdálenost atleta od kamery, jelikož se jeho velikost v průběhu videa mění. Pro přesnější konverzi bych tedy musel zvolit jiný způsob určení pohybu objektů ve videu.

Kloub výsledného 3D modelu reprezentuji jako uspořádanou trojici  $(x,y,z)$ , počátek této soustavy umístím do místa kotníku odrazové nohy při odrazu atleta. Ideální by bylo počátek umístit na zadní hranu zasouvací skřínky. Zasovací skřínka je schovaná za doskočištěm, tudíž bych jeho pozici musel odhadovat na základě detekce tyče, což by mohlo být nepřesné.

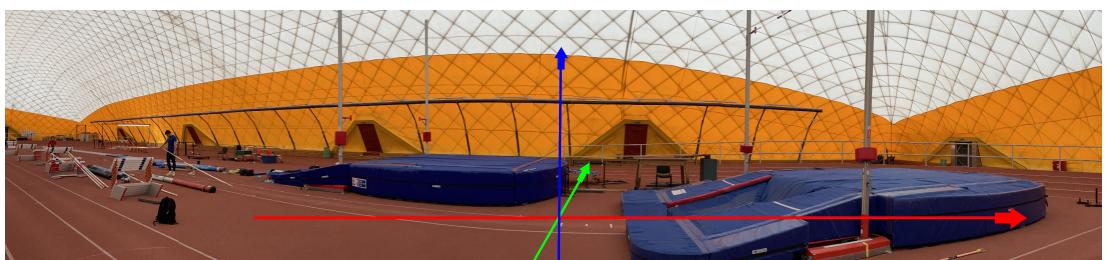
Hodnota první složky určuje horizontální vzdálenost bodu od počátku ve směru rovnoběžném s osou rozběžiště. Tato hodnota stoupá ve směru rozběhu.

Hodnota druhé složky určuje horizontální vzdálenost bodu od počátku ve směru kolmém na osu rozběžiště. Tato hodnota stoupá ve směru doprava z pohledu atleta při rozběhu.

Hodnota třetí složky určuje vertikální vzdálenost bodu od počátku. Tato hodnota stoupá ve směru vzhůru.

Program zatím využívá dvoudimenzionální podprostor 3D prostoru z důvodu komplikovaného odhadu hodnoty druhé složky pozice zkoumaných bodů. 3D prostor jsem implementoval s výhledem budoucího rozšíření aplikace.

Souřadný systém videa je znázorněný na obrázku 2.2.



Obrázek 2.2: Vizualizace souřadného systému 3D prostoru. Červená reprezentuje osu  $x$ , zelená  $y$  a modrá  $z$ . Směr šipek odpovídá růstu hodnot bodů v prostoru.

## 2.2 Vstup programu

Vstupem programu je video skoku o tyči pořízené běžnými prostředky, jakými jsou mobilní telefon nebo tablet.

Videa skoku o tyči se nejčastěji natáčí ze strany. Kamera se standardně nachází na kolmici k rozběžišti, která prochází místem odrazu. Můj program očekává na vstupu video ze strany, pro přesnost hodnot výsledných parametrů je vhodné, aby byla kamera na úrovni odrazu. Pro potřebu přesnější analýzy běhu atleta je lepší umístit kameru na úroveň poloviny rozběhu.

Při trénincích se kamera nachází ve vzdálenosti okolo 10 metrů od rozběžiště, závodní skoky jsou ve většině případů natáčeny z tribuny, tudíž je vzdálenost větší (v řádu desítek metrů). Velikost atleta ve videu nemá na funkcionality programu vliv, ale detekce menší postavy může vyústit v horší přesnost.

## 2.3 Analýza videa

Vstupem programu je video skoku o tyči. Toto video program nejprve zpracuje do vhodné reprezentace, následně ve videu nalezne atleta. Na základě pozice atleta v jednotlivých snímcích detekuje kostru atleta, kterou převede do 3D modelu. Podle pohybu atletova těla, který je reprezentovaný získaným modelem, analyzuje parametry skoku a určí jejich hodnoty. Výstupem programu je model atletova těla a hodnoty definovaných parametrů. Tento výstup následně uloží a zobrazí uživateli.

Program lze spustit na počítači s Unixovým operačním systémem.

### 2.3.1 Zpracování videa

Prvním krokem k provedení analýzy je zpracování videa. Aby video nebylo otevřené po celou dobu analýzy, rozhodl jsem se při analýze pracovat s jednotlivými snímky, které si ukládám při jediném průchodu videem. Rozlišení videa může být libovolné, ale z důvodu časové náročnosti rozlišení snímků videa před analýzou zmenším.

### 2.3.2 Nalezení atleta

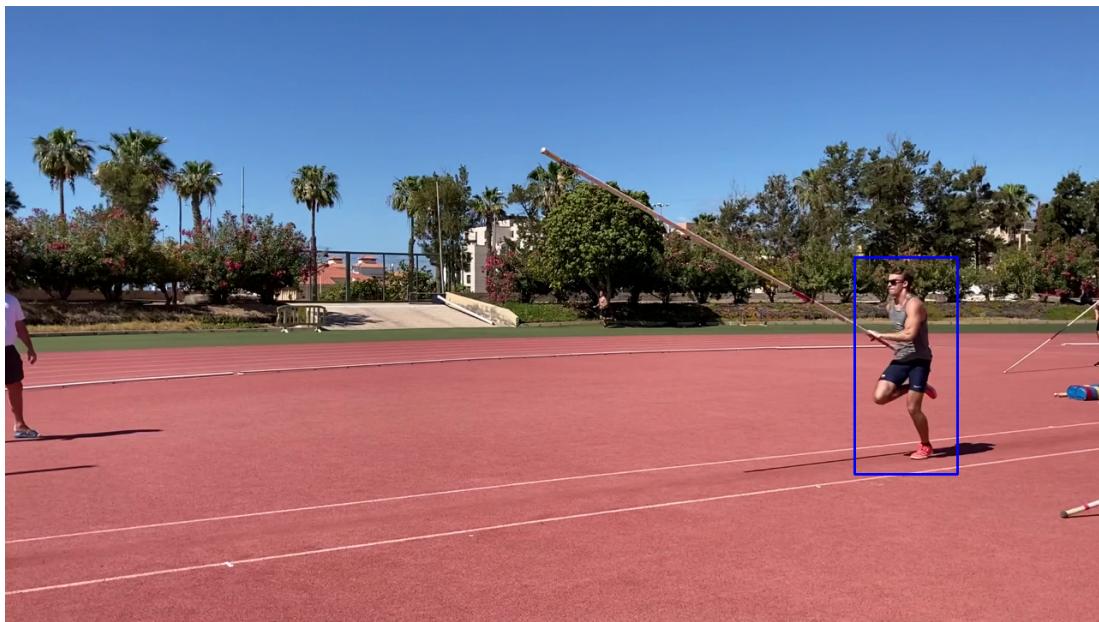
Pro nalezení atleta stačí určit snímek, ve kterém je atlet dobře vidět, a jeho pozici v něm. Pozici atleta bude reprezentovat jeho ohraničující rámeček, který je vyznačený na obrázku 2.3.

Pro nalezení atleta jsem zvolil dva přístupy.

Prvním z nich je manuální. Uživatel po spuštění analýzy vybere snímek, ve kterém je atlet dobře vidět. Následně vyznačí do videa ohraničující rámeček, do něhož se vejde postava atleta.

Druhým přístupem je automatické nalezení atleta. Při automatické detekci atleta je potřeba detektovat postavy ve snímku videa. Výsledkem této detekce je množina ohraničujících rámečků reprezentujících postavy ve snímku. Z těchto postav je potřeba vybrat tu, která reprezentuje atleta.

Atlet provádí na videu specifický pohyb. Nejprve se rozeběhne a poté provede skok, tedy ohraničující rámeček postavy atleta by se ve videu měl pohybovat



Obrázek 2.3: Ohraničující rámeček atletova těla ve snímku videa 1.MOV (viz tabulka 4.1).

horizontálně bez výrazných výkyvů ve vertikálním směru, následně se pohybovat směrem vzhůru a nakonec směrem dolů.

Pomocí trasování objektů mohu ve videu zkoumat pohyb postav a na základě jejich reálného pohybu filtrovat postavy, které mohou reprezentovat atleta.

V prvním snímku detekuju postavy a uložím si je do seznamu, následně budu zkoumat jejich pohyb. Jejich pohyb budu zkoumat pomocí trasování objektů v souřadném systému videa.

Pokud se postava v horizontálním směru nepohně za určitou dobu alespoň o danou vzdálenost, smažu ji ze seznamu, jelikož nereprezentuje atleta. Jakmile je seznam postav prázdný, spustím detekci postav znova v právě zpracovávaném snímku.

V opačném případě považuji postavu za atleta. Tímto okamžikem hledání atleta ve videu končí, jelikož vím, ve kterém snímku jsem atleta detekoval poprvé a jaký byl jeho ohraničující rámeček v daném snímku.

### Trasování postav

Trasování postav ve videu používám k odlišení atleta od ostatních postav na základě analýzy jejich pohybu. Trasování atleta provádím také při detekci kloubů kostry, ale modifikovaným způsobem, který je popsáný v sekci 2.3.3.

### 2.3.3 Detekce kostry

#### Trasování atleta

Trasování atleta při rozběhu nedělá většině trackerů problém. Nepřesné trasování nastává při skoku. Důvodem této nepřesnosti je rotace atletova těla. Tato rotace je při skoku výrazná a při vodorovné pozici trupu nemusí atletovo tělo zabírat věšinu ohraničujícího rámečku, což způsobí, že tracker začne trasovat

pozadí videa a ztratí atleta. Tato vlastnost a její napravení je vidět na obrázku 2.4.

Modifikací vedoucích ke zlepšení přesnosti jsem zkusil několik.

Nejprve jsem otáčel snímky videa podle naklonění trupu v předešlém snímku. K určení tohoto naklonění jsem použil detekovanou kostru atletova těla.

Následně jsem aktualizoval tracker, aby trasoval pouze trup. Jeho pozici jsem opět získával z detekované kostry. Tyto metody selhávaly při chybné detekci kostry, ačkoliv jsem se snažil implementovat kontrolní mechanismus. Například jsem neaktualizoval tracker na kostře, jejíž trup byl příliš daleko od právě trasovaného trupu. Problém detekce kostry špatné postavy je vidět na obrázku 2.6.

Nakonec jsem implementoval metodu, která nezávisí na detekci kostry atleta.

Atletovo tělo netrasuji jako celek, ale po částech. Původní ohraničující rámeček rozdělím na mřížku menších rámečků. Každý rámeček trasuji zvlášť a průběžně aktualizuji ty, které atleta ztratí.

Po každých několika snímcích kontroluji pohyb jednotlivých rámečků ve videu od předešlé kontroly. Určím rámeček, který se pohnul nejvíce. Pohyb rámečku musí být horizontálně ve stejném směru, jako je rozběh atleta. Následně posunu původní mřížku tak, aby odpovídala pozici nejvíce se pohybujícího rámečku. Do mřížky poté přesunu rámečky, jejichž trasování selhalo, na své původní místo. Průběh aktualizace je zobrazen na obrázku 2.5.

## Detekce kloubů těla

Pro určení atletovy kostry stačí detekovat klouby, které následně spojím úsečkami. Klouby detekují pomocí konvoluční sítě. Jelikož má implementace očekává, že na snímku je jediná postava, bylo pro dostatečnou kvalitu detekce kloubů potřeba vyříznout ze snímku okno, v němž se nachází atlet a příslušně okno rotovat, aby byl atlet v co nejvzpřímenější pozici. Sít, kterou používám, je totiž natrénovaná na databázi MPII Human Pose dataset (Andriluka a kol., 2014), v níž je většina postav ve vzpřímené poloze.

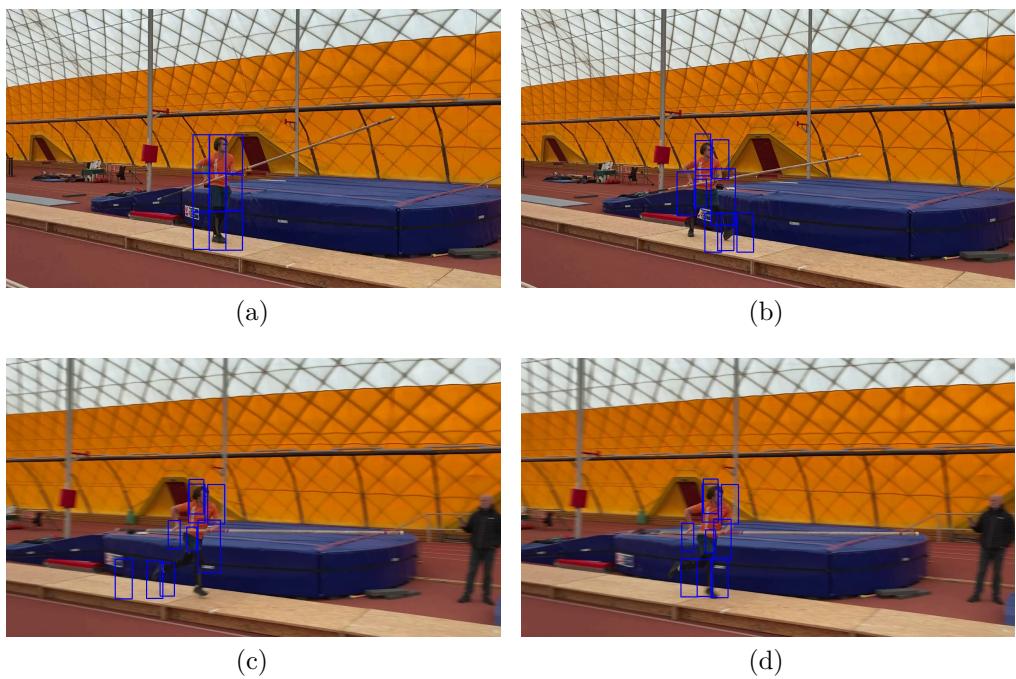
Určení pozice atletova okna provádím s použitím rámečků na těle atleta, kterého trasuji. Rámečky uzavřu do co nejmenšího rámečku a v něm naleznu střed. To je střed prvního okna pro detekci kostry. Střed okna v následujících snímcích posouvám směrem ke kyčlím atleta. Tento posun získám z pozice rámečku obsahujícího trackery atletova těla a pozice kyčlí kostry detekované v předchozím snímku. Aby se okno neposunulo příliš daleko od těla atleta při chybné detekci kostry, posouvám střed okna jen uvnitř rámečku obsahujícího trackery atletova těla. Tím napravím detekci kostry chybné postavy po několika snímcích, viz obrázek 2.6

Rotaci okna provádím podle náklonu trupu poslední validně detekované kostry. Validně detekovanou kostrou rozumím kostru, které jsem detekoval všechny klouby. Abych zamezil rotacím podle chybné detekce, aktualizuji poslední známý úhel náklonu trupu jen v případě, že se od předchozího příliš neliší. Pokud tedy při skoku detekuju kostru postavy, která stojí u místa odrazu, neaktualizuji úhel rotace a okno rotuji podle posledního známého náklonění trupu atleta. [OBRÁZEK]

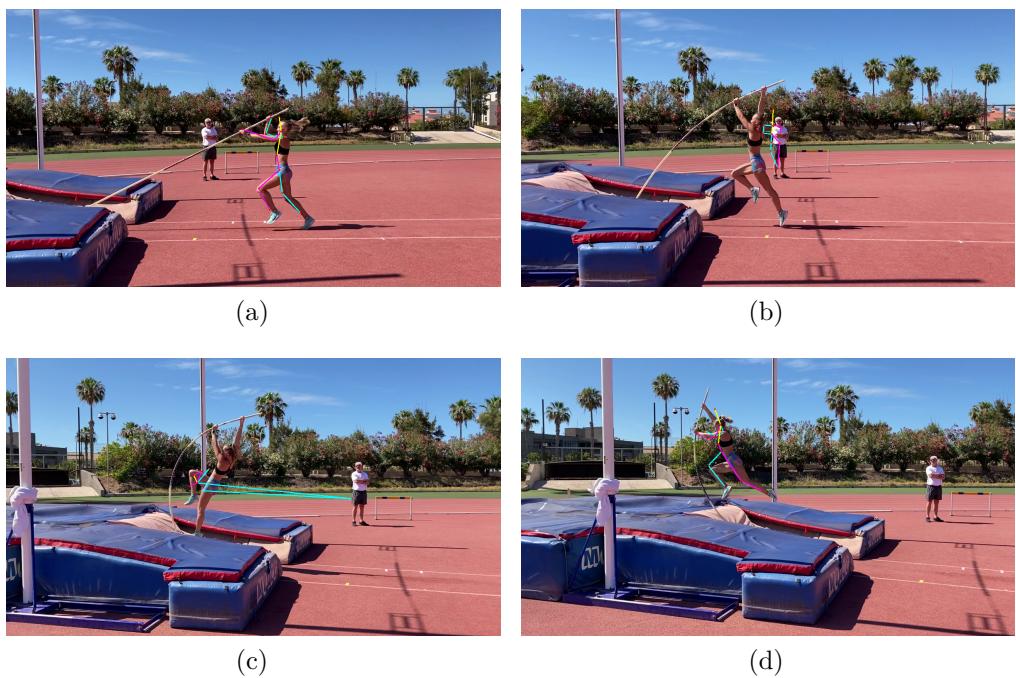
Pokud se mi nepodaří detekovat celou kostru, zkusím okno rotovat o určitý úhel oběma směry a vyberu ten s nejúspěšnější detekcí, co se počtu detekovaných kloubů kostry týče.



Obrázek 2.4: Použití jednoho trackeru (vlevo) a mřížky trackerů, která je implementovaná v programu (vpravo). Použité snímky jsou vybrané z videa 33.MOV (viz tabulka 4.1).



Obrázek 2.5: Pohyb mřížky v průběhu její první aktualizace. (a) - úvodní pozice mřížky, (b) - aktualizace stále neproběhla, protože není znám směr rozběhu, (c) - poslední snímek před aktualizací mřížky, (d) - první snímek po aktualizaci mřížky



Obrázek 2.6: Detekce špatné kostry a následné napravení. (a) - správná detekce, (b) - špatná detekce, (c) - průběh nápravy, (d) - správná detekce

Jelikož atlet mění v průběhu videa velikost - standardně se zvětšuje, protože se přibližuje ke kameře - je potřeba okno postupně zvětšovat. První okno je dané původním ohraničujícím rámečkem, který zvětší konstantou. Velikosti následujících oken určují podle dosud největší validně detekované kostry. Velikost se rovná vzdálenosti dvou nejvzdálenějších kloubů kostry. Okno pro detekci kostry je čtvercové, jehož strana má délku velikosti dosud největší validně detekované kostry, jejíž velikost zvětší konstantou. Okno pro první detekci nemusí být čtvercové, odvíjí se od výsledku nalezení atleta ve videu.

### 2.3.4 Konverze kostry do 3D modelu

Konverzi detekované kostry do 3D modelu provedu pomocí algoritmu pro převod mezi souřadnými systémy videa a 3D modelu. Tyto souřadné systémy jsou popsány v sekci 2.1 a implementace algoritmu v sekci 3.2.

### 2.3.5 Analýza parametrů

Vzhledem k vlastnostem mnou zvoleného modelu nebude vhodné analyzovat některé parametry skoku. Rozeberu proveditelnost analýzy biomechanických parametrů popsaných v sekci 1.1.2.

#### Délka jednotlivých kroků rozběhu

Délka jednotlivých kroků rozběhu je velice citlivá na úhel, pod kterým atleta snímám, a jeho vzdálenost od kamery. Vliv vzdálenosti by bylo možné vyřešit reprezentací délky kroku relativně vůči výšce postavy. Ovšem úhel osy kamery vůči rozběžnosti v programu nijak neřeší, tudíž se této nepřesnosti nelze jednoduše zbavit.

Proto jsem se rozhodl tento parametr neanalyzovat.

Místo délky kroku jsem se rozhodl implementovat zjištování hodnoty doby trvání jednotlivých kroků. K zisku této hodnoty stačí najít snímky, ve kterých se nižší noha začíná pohybovat směrem vzhůru. Tím získám (po jednoduché filtrace) momenty dokončení odrazů jednotlivých kroků. Doba, která uplyne mezi následujícími kroky, je výsledná hodnota.

#### Doba oporové fáze kroků

Dobu oporové fáze kroku lze analyzovat mírným rozšířením analýzy doby trvání jednotlivých kroků. Při spuštění stejného algoritmu od konce videa získám momenty došlapů. Spolu s momenty odrazů lze určit dobu oporové fáze jednotlivých kroků.

Vzhledem k tomu, že jeho přesnost závisí na frekvenci snímků videa, jsem se analýzu tohoto parametru rozhodl neanalyzovat.

#### Náběhová rychlosť

Náběhovou rychlosť nelze analyzovat v průběhu celého rozběhu příliš přesně z důvodu měnícího se úhlu, pod kterým kamera snímá atletův rozběh. V momentu odrazu lze rychlosť reprezentovat poměrně přesně, ale jen v jednotkách závislých

na pixelech. Pro převod na jednotky s větší vypovídající hodnotou by bylo možné využít délku tyče nebo výšku atleta. Pro implementaci této metody by bylo potřeba, aby uživatel zadal příslušný parametr - výšku atleta nebo délku tyče - a výsledná přesnost by nebyla příliš dobrá, jelikož se náběhová rychlosť u různých skoků liší nepatrně. Lepším způsobem pro analýzu tohoto parametru tak zůstává využití fotobuněk nebo radaru.

Místo analýzy tohoto parametru jsem zkoumal ztrátu horizontální rychlosti ramen a boků při odrazu. Tato změna nezávisí na použitých jednotkách, takže není potřeba odhadovat vzdálenosti. Pro zisk těchto hodnot porovnávám změnu pozice ramen a boků daný časový úsek před odrazem, při odrazu a stejný časový úsek po něm.

### Výška boků v průběhu rozběhu

Výška boků bez jakéhokoliv škálování v závislosti na velikosti atleta ve videu není příliš vypovídající pro porovnání začátku a konce rozběhu. Vliv na výsledný skok mají spíše lokální výkyvy, především ty, které nastávají v konci rozběhu. Tím pádem je hodnota tohoto parametru užitečná i bez škálování naměřených hodnot.

Výšku boků jsem tedy mezi implementované parametry zahrnul, pro zisk jeho hodnoty průměruji výšku levé a pravé kyčle.

### Místo odrazu

Přesnost tohoto parametru závisí také na zvoleném modelu, kterým reprezentuji tělo atleta, tedy kolik bodů na těle detekuji. Nejčastěji se udává jako vzdálenost špičky chodidla od zadní hrany zasouvací skříňky. Pozici špičky chodidla je složité odhadnout, pokud znám jen pozici kotníku. Přesnost odhadu hrany zasouvací skříňky by na hodnotu tohoto parametru měla také značný vliv.

Vzdálenost místa odrazu se zkoumá kvůli jejímu vlivu na horizontální vzdálenost úchopu horní ruky od odrazové nohy. Tuto vzdálenost jsem schopen změřit poměrně přesně, bohužel pouze v nedefinovaných jednotkách, případný odhad konkrétní vzdálenosti by nedosáhl větší přesnosti než odhad pouhým okem.

Pozice odrazu se jednoduše a přesně určí pouhým okem, především pokud jsou poblíž místa odrazu na zemi značky, proto jsem tento parametr neimplementoval.

### Výška úchopu

Podobně jako v předchozím případě lze hodnotu tohoto parametru poměrně přesně určit pouhým okem při znalosti délky tyče. Proto nebylo potřeba analýzu tohoto parametru zatím implementovat.

V budoucnu bych aplikaci rád rozšířil o detekci tyče, která by možnosti analýzy skoku značně rozšířila.

### Úhel odrazu

Jelikož je osa kamery při odrazu většinou kolmá na směr rozběhu, je možnost získání přesné hodnoty tohoto parametru solidní. Úhel určím podobně jako ztrátu rychlosti. Porovnám pozici boků daný časový úsek před odrazem, při odrazu a stejný časový úsek po něm. Na základě rozdílů těchto pozic určím úhel rozběhu

(jeho konce) a skoku (jeho počátku) vůči horizontální ose. Po odečtení úhlu rozběhu od úhlu skoku získám úhel odrazu.

Úhel odrazu jsem se tedy rozhodl analyzovat.

### Úhly v kloubech při odrazu

Úhly v kloubech kostry lze z modelu získat snadno, ale lze je odhadnout poměrně přesně i bez analýzy programem - alespoň pro účely porovnání skoků. Nejedná se o tak důležitý parametr jako úhel odrazu, tedy implementaci analýzy úhlů kloubů těla ponechám jako možné rozšíření programu do budoucna.

### Doba trvání skoku

Pro zisk hodnoty tohoto parametru by bylo užitečné detekovat tyč. Tuto funkcionality jsem neimplementoval, tudíž tento parametr není mezi analyzovanými.

### Převýšení

Pro určení míry převýšení je také vhodné detekovat tyč. Druhou možností je převod souřadného systému na metry a zadání výšky úchopu. Poté by bylo možné převýšení určit. V budoucích verzích programu by se tento parametr mohl objevit. Zatím jsem ho neimplementoval.

### Další implementované parametry

Nad rámec popsaných parametrů jsem se rozhodl implementovat analýzu následujících parametrů:

**Úhel došlapu jednotlivých kroků** Parametrem souvisejícím s dobou oporové fáze kroku je úhel došlapu. Jedná se o úhel mezi kotníkem, kyčlí a vertikálou. Aproximuje vzdálenost došlapu před těžiště atleta. Pro zisk hodnoty tohoto parametru použiji metodu získávání snímků, ve kterých dochází k došlapu.

**Náklon trupu** Jedná se o náklon trupu v rovině určené vertikálou a osou rozběžiště. Hodnota tohoto parametru není příliš přesná na začátku rozběhu, ale s postupem času se přesnost zvětšuje díky lepšímu úhlu záběru těla atleta.

### Důležité momenty skoku

Kromě parametrů ve videu detekuji podstatné momenty skoku, za něž považuji začátek rozběhu, odraz a moment kulminace boků nad laťkou.

## 2.4 Analýza modelu

Aby uživatel nemusel video analyzovat znovu, je mu umožněno načtení modelu ze souboru, který je popsaný v sekci 2.5.

Funkcionalita programu je velice podobná, na rozdíl od analýzy videa se na začátku načte také soubor s uloženým modelem. Neprovádí se hledání atleta ve videu, detekce kostry, ani konverze kostry do 3D modelu. Analýza parametrů

probíhá stejně jako při analýze videa, model se neukládá, ukládají se jen parametry, aby se jejich hodnoty daly využít pro případnou analýzu.

Kostru z 3D modelu získá program s využitím konverzí souřadných systémů, kterou popisují v sekci 3.2.

## 2.5 Výstup programu

Výstupem programu je model reprezentující pohyb atleta v průběhu rozběhu i skoku a hodnoty definovaných parametrů, které jsou

- doba trvání jednotlivých kroků,
- ztráta rychlosti ramen a boků,
- výška boků,
- úhel odrazu,
- úhel došlapu jednotlivých kroků,
- náklon trupu a
- důležité momenty skoku.

### 2.5.1 Uložení výstupu

**Parametry** Hodnoty parametrů vypíšu do souboru pro případnou hlubší biomechanickou analýzu. Jedná se o `.csv` soubor, jehož formát je následující.

Parametry ukládám po sloupcích. První sloupec obsahuje relativní čas vůči momentu odrazu v každém snímku videa, následující sloupce obsahují parametry.

První řádek souboru obsahuje název analyzovaného videa, druhý názvy parametrů, počínaje od druhého sloupce. Ve třetím řádku jsou vyspané jednotky, ve kterých jsou hodnoty parametrů určeny.

Parametry, které reprezentuje jedna hodnota, zabírají jen první řádek hodnot.

Parametry, které neodpovídají snímkům, ale obsahují více hodnot (například doba trvání jednotlivých kroků), jsou uloženy postupně od prvního řádku, dokud jejich hodnoty znám.

Parametry, pro které znám hodnotu v každém snímku, jsou uloženy v řádcích odpovídajících příslušným snímkům videa.

Takto vypadá prvních pár řádků souboru s hodnotami parametrů po analýze videa `17.MOV` (viz tabulka 4.1):

```
data/17.MOV
Time,Hips height,Hips velocity loss,Left ankle height,Right ankle height,....
s,px,%,px,px,%,°,s,°
-3.85,75.21,106.04,5.62,11.18,27.79,-7.38,0.13,248.43,-2.30
-3.83,79.86,,11.89,3.87,, -0.25,0.20,,1.84
-3.82,75.98,,6.12,7.39,, -14.09,0.03,, -6.30
-3.80,81.02,,8.38,2.88,,21.95,0.03,, -2.21
-3.78,77.05,,6.95,15.49,,29.16,0.08,, -6.30
-3.77,80.58,,18.09,11.37,, -24.36,0.12,,1.84
-3.75,75.86,,4.51,5.81,, -11.59,0.03,, -6.30
```

**Model** Po analýze videa se výsledný model uloží do textového souboru. První řádek obsahuje název analyzovaného videa.

Následně se v souboru opakují části reprezentující detekce jednotlivých snímků. Na prvním řádku této části je číslo snímku videa, následuje pozice levého horního rohu snímku v souřadnicích videa převedených do 3D - hodnota  $x$ -ové osy roste směrem doprava. Na dalších řádcích jsou vypsány klouby kostry těla v souřadnicích snímku převedených do 3D. Tyto klouby jsou uloženy postupně, podle zvoleného modelu reprezentace těla, ten je popsán v sekci 3.1.

Takto vypadá prvních pár řádků popisu modelu, který program extrahoval z videa 17.MOV (viz tabulka 4.1):

```
data/17.MOV
0
-719,0,-421
716.211,0,340.27
716.211,0,368.108
710.632,0,373.676
705.053,0,390.378
699.474,0,390.378
727.368,0,373.676
721.789,0,395.946
705.053,0,395.946
710.632,0,407.081
721.789,0,446.054
732.947,0,473.892
716.211,0,412.649
710.632,0,446.054
693.895,0,479.459
716.211,0,390.378
```

Po analýze modelu se model do souboru neukládá.

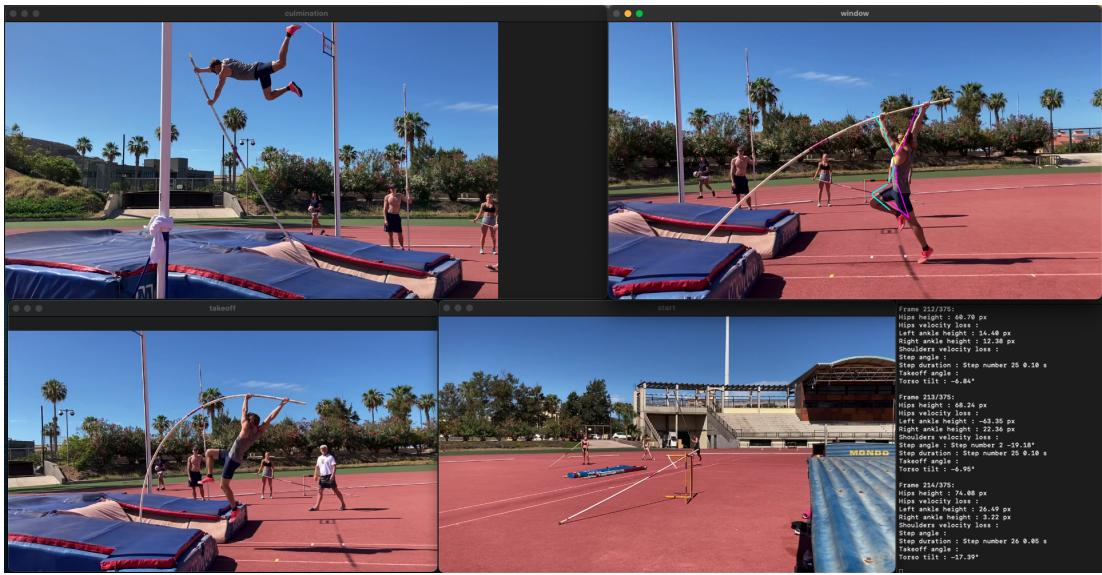
### 2.5.2 Zobrazení výstupu uživateli

Výstup programu je vhodné uživateli přehledně zobrazit. Rozhodl jsem se pro vlastní prohlížeč snímků videa. Mezi jednotlivými snímky se lze pohybovat dopředu i dozadu. Ve výchozím režimu se zobrazí video se zakreslenou kostrou, jejíž zakreslení do snímku lze vypnout a znova zapnout.

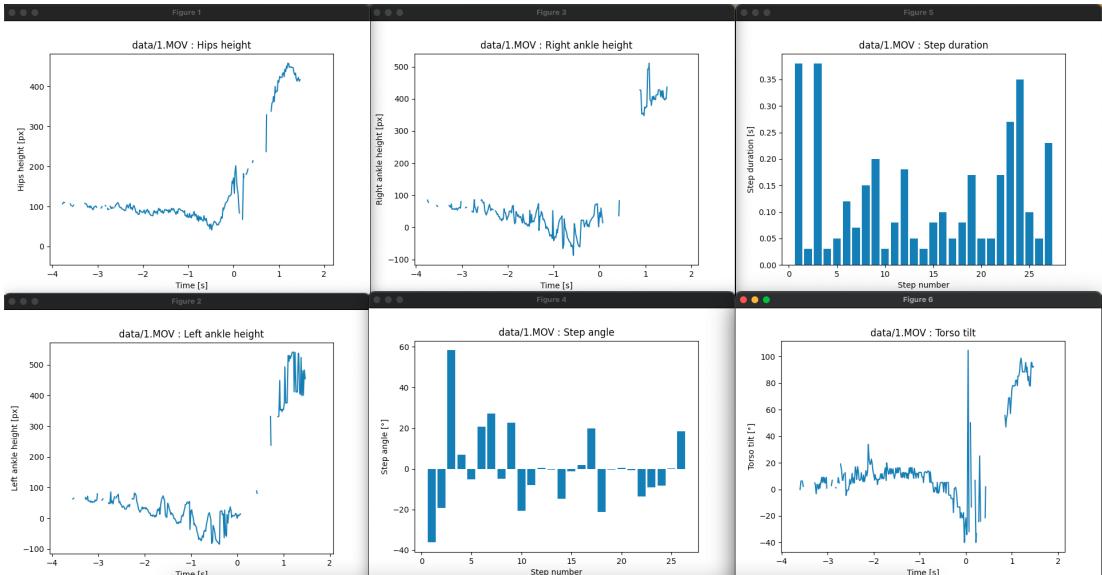
Při zobrazení prohlížeče dojde také k zobrazení snímků, které reprezentují začátek rozběhu, moment odrazu a moment kulminace nad laťkou.

Při prohlížení daného snímku vypíše program do konzole hodnoty příslušných parametrů. Parametry jsem rozdělil do kategorií podle části skoku, při které jsou podstatné. Například úhel došlapu je irelevantní při skoku, ale je důležitý při rozběhu. Tedy při snímku rozběhu se vypíše například úhel naklonění trupu v daném snímku a doba trvání právě probíhajícího kroku. Parametry související s odrazem vypíše program při zobrazení snímku, který reprezentuje moment odrazu. Způsob zobrazení výstupu je vidět na obrázku 2.7.

Po ukončení prohlížeče snímků zobrazím uživateli hodnoty parametrů zanesených do grafů, které se načtou z vygenerovaného souboru.



(a)



(b)

Obrázek 2.7: Způsob zobrazení výstupu uživateli. (a) - snímky zachycující začátek rozběhu, odraz a kulminaci boků, prohlížeč videa a výpis příslušných hodnot analyzovaných parametrů, (b) - grafy znázorňující hodnoty parametrů v průběhu skoku. Pro zobrazení tohoto výstupu byl použit model získaný z videa 1.MOV (viz tabulka 4.1).

## 2.6 Programové vybavení

### 2.6.1 Počítačové vidění a zpracování obrazu

Pro potřeby počítačového vidění a zpracování obrazu lze využít několik knihoven.

**OpenCV** OpenCV (Bradski, 2000) je jednou z nejrozšířenějších open-source knihoven. Mezi její výhody patří všeobecnost. Je vhodnou volbou pro zpracování obrazu, videí a vytváření modelů počítačového vidění a strojového učení. OpenCV vznikla již v roce 2000, což je jedním z důvodů, proč ji využívá a udržuje mnoho vývojářů. Na internetu lze tedy najít mnoho informací o naprosté většině funkcionality, kterou tato knihovna disponuje.

OpenCV lze spustit na různých platformách a je vhodnou volbou i pro mobilní zařízení, což může být užitečné pro budoucí rozšíření aplikace. Tuto knihovnu lze použít v programu psaném v jazycích C, C++, Python nebo Octave.

**OpenVINO** OpenVINO je knihovna z dílny společnosti Intel. Existuje ve dvou verzích, jako open-source a jako distribuce spravovaná právě společností Intel. Knihovna disponuje solidní zásobou modelů hlubokého učení pro potřeby počítačového vidění a jejich optimalizací pro procesory společnosti Intel.

Ačkoliv se vývojáři snaží podporovat i jiné procesory, může být spuštění této knihovny na procesorech s jinou architekturou problematické. Možnosti knihovny pro zpracování obrazu jsou omezené, takže se často používá v kombinaci s OpenCV.

API této knihovny je určené pro jazyky C, C++ a Python.

**VisionWorks** Společnost NVidia vyvinula knihovnu VisionWorks, která využívá rychlosti grafických karet. Tato knihovna slouží především k detekci a trasování objektů. Má výborné výsledky v oblasti autonomního řízení, což není příliš použitelné v mému programu.

**Vision Workbench** NASA spravuje vlastní knihovnu Vision Workbench, zabývající se počítačovým viděním a zpracováním obrazu. Hlavním cílem této knihovny je zpracování obrazu a počítačové vidění pro vesmírné roboty.

### 2.6.2 Grafická reprezentace výstupu

Užitečným prvkem při analýze videozáznamu je přehrávač videa. Pro mé potřeby by bylo vhodné, aby se pro daný snímek zobrazovaly hodnoty parametrů, takže přehrávač videa nahradím prohlížečem snímků videa.

Pro analýzu parametrů a jejich případné porovnání je vhodné jejich hodnoty zanést do grafů.

**gnuplot** Jedním z nejrozšířenějších programů pro tvorbu grafů je gnuplot (Williams a kol., 2013). Program se spouští z příkazové řádky a běží na několika platformách. Jedná se o program, jehož první verze vznikla už v roce 1986, takže má, podobně jako OpenCV, velké množství uživatelů, kteří s ním mají bohaté zkušenosti.

**Gnuplot** zpracovává vstup z příkazové řádky nebo skriptů, které popisují, jaká data se mají vykreslit. Příkazy nejsou psané v žádném programovacím jazyce, ale mají vlastní syntaxi.

**MATLAB** Velké množství vědců používá pro práci s daty MATLAB (MATLAB, 2010). Jedná se o programovací jazyk, který se specializuje například na numerické výpočty, práci s maticemi nebo zanášení dat do grafů.

MATLAB je možné provázat s jinými programovacími jazyky, mezi něž patří C, Java nebo Python.

**Matplotlib** Stále větší oblibě se těší programovací jazyk Python, který disponuje knihovnou pro generování grafů. Tato knihovna se jmenuje Matplotlib (Hunter, 2007), která se používá pouze pomocí jazyka Python. Standardním způsobem práce s touto knihovnou je přes modul Pyplot, jehož funkcionalita je podobná MATLABu. Pyplot disponuje množinou funkcí, které umí vykreslit různé typy grafů. Výhodou použití Pythonu je také snadné spouštění skriptů, které není potřeba manuálně kompilovat před jejich spuštěním.

Skripty psané v Pythonu lze spouštět přímo z kódu jazyka C++, což umožňuje automatické zobrazení grafů po doběhnutí analýzy videa skoku o tyči.

# 3. Realizace

## 3.1 Zvolené prostředky

Pro potřeby analýzy videa jsem použil jazyk C++ a knihovnu OpenCV. Jelikož je analýza výpočetně náročná, dal jsem z rychlostních důvodů přednost jazyku C++ před jazykem Python. Také mám s jazykem C++ více zkušeností.

Tracker, který jsem zvolil pro trasování objektů je `cv::TrackerCSRT`, jeho implementace je založena na využití Discriminative Correlation Filter with Channel and Spatial Reliability (Lukežič a kol., 2018). Tento tracker jsem zvolil na základě jeho vysoké úspěšnosti v soutěži The Visual Object Tracking VOT2017 challenge (Kristan a kol., 2017) a jeho implementaci v OpenCV. O přesnosti trasování objektů, které ve videu rotují jsem bohužel nenašel dostatek informací, abych tento parametr při výbírání trackeru uvažoval.

Trackery nejsou součástí samotné knihovny OpenCV, jejich implementace je obsažena v modulu OpenCV-contrib.

Pro zobrazení parametrů v grafické podobě jsem se rozhodl využít Matplotlib. Pro využití této knihovny není potřeba znát speciální syntaxi, základní znalosti Pythonu jsou dostačující. Jelikož je knihovna Matplotlib určena pro Python, je snadné napsat skript, který nejen vygeneruje grafy, ale také zpracuje data, která se mají zobrazit.

Program lze spustit na počítači s Unixovým operačním systémem.

## 3.2 Konverze souřadných systémů

Souřadné systémy snímku a videa jsou jen posunuté o vzájemnou pozici levého horního rohu snímku a pozadí v prvním snímku. Tudíž jejich konverzi reprezentuje operace posunutí.

Konverze mezi souřadným systémem videa a prostoru je složitější. Jednotka systémů je stejná, tudíž škálování není třeba provádět. První složky bodů v obou systémech spolu korespondují. Druhá složka bodu v souřadném systému videa odpovídá třetí složce bodu v prostoru. Tuto konverzi používám jen pro určení pozice kloubů těla atleta.

Je třeba zohlednit směr rozběhu atleta. Pokud běží doleva, musím převrátit hodnotu první složky. Pokud běží doprava, hodnotu první složky nechám stejnou.

Hodnotu druhé složky nastavím na 0. Odhad pozice kloubu v tomto směru je problematický, jelikož se atlet při skoku otáčí podél své osy, tudíž její implementaci ponechám do budoucích verzí programu. Pro její správné určení bude také potřeba zohlednit stranu, ze které je video natočeno.

Hodnotu třetí složky v prostoru získám převrácením druhé složky bodu v souřadném systému videa.

Výsledkem tohoto postupu je posunutý prostor, musím ho tedy posunout s využitím pozice kotníku odrazové nohy v momentu odrazu.

Konverzi bodu kostry  $(x',y')$  v souřadném systému videa do prostoru lze

reprezentovat takto:

$$\begin{aligned} \text{Při běhu doprava: } & (x,y,z) = (x', 0, -y') - d. \\ \text{Při běhu doleva: } & (x,y,z) = (-x', 0, -y') - d. \end{aligned}$$

$d$  značí pozici kotníku odrazové nohy při odrazu v prostoru.

### 3.3 Struktura programu

Program se skládá z několika tříd, UML diagram popisující strukturu je na obrázku 3.1. Kompletní funkcionality je přístupná skrz třídu `video_processor` a její metody `process_video` a `process_model`. Tyto metody komunikují se zbylými částmi programu.

O nalezení atleta se stará třída `athlete_finder`, ta po nalezení atleta vrátí jeho pozici ve videu. K určení atletovy pozice využívá instance třídy `athlete_candidate`, které reprezentují postavy ve videu. Analýzu pohybu jednotlivých postav provádí jednotlivé instance třídy `movement_analyzer` a část zabývající se detekcí pozadí provádí pro každou postavu instance třídy `background_tracker`.

Funkcionality týkající se detekcí kostry atleta zajišťuje třída `person`. Pohyb atleta zkoumá instance třídy `movement_analyzer` ve spolupráci s instancí třídy `background_tracker`. O detekci jednotlivých částí těla se stará instance třídy `parts_detector`.

Po detekování kostry těla kovertuje detekované části těla do 3D modelu atletova těla třída `model`, výsledek této detekce využívá pro určení parametrů instance třídy `vault_analyzer`, která určí hodnoty biomechanických parametrů. Parametry reprezentují specializované struktury, které dědí vlastnosti struktury `parameter`. Struktura zobrazující hierarchii parametrů je na obrázku ???. `vault_analyzer` také zajišťuje ukládání parametrů do souboru.

Zobrazování výstupu má za úkol třída `viewer`.

Konstanty, definice vlastních typů a pomocné funkce jsou uloženy v souborech `forward.cpp` a `forward.hpp`.

Jelikož nemusím vždy detekovat validní pozice všech kloubů těla ve snímku, zabalím body do struktury `std::optional`.

### 3.4 Vstup programu

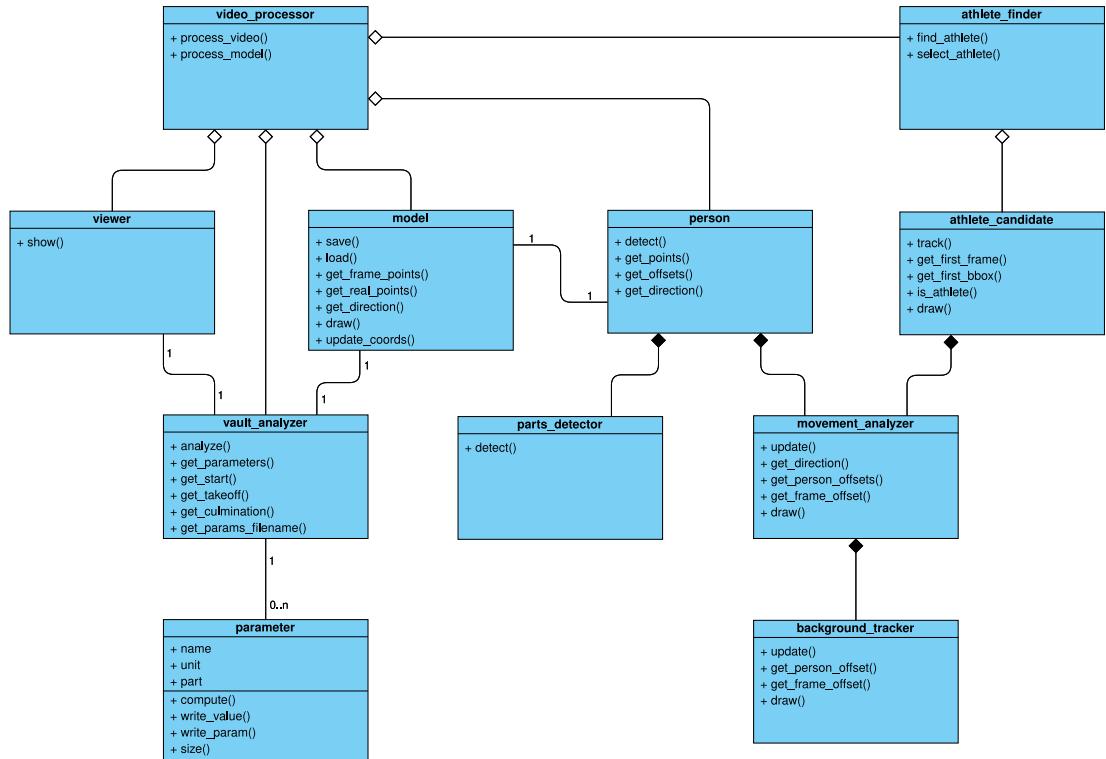
Programu jsou při jeho spuštění předány parametry. Parametry specifikují mód, ve kterém si uživatel přeje program spustit, a obsahují cesty k souborům, které má program zpracovat.

O zpracování parametrů se stará soubor `main.cpp`, který definuje vstupní bod aplikace a volá metody třídy `video_processor`, která zpracuje video, případně model, a výsledek předá k analýze.

### 3.5 Analýza videa

#### 3.5.1 Zpracování videa

Zpracování videa má na starosti třída `video_processor`.



Obrázek 3.1: UML diagram zobrazující strukturu programu.

Video načtu s použitím třídy `cv::VideoCapture`. Z videa získám hodnotu frekvence snímků a jednotlivé snímky videa uložím při průchodu do vektoru 3-rozměrných matic `cv::Mat`, které reprezentují snímky videa.

Z důvodu časové náročnosti videí s velkým rozlišením zmenším každý snímek metodou `cv::resize`. Výsledný snímek videa má rozlišení  $1280 \times 720$  px. Při zpracování videa na šířku je výška videa 720 px, šířka se změní, aby byl poměr stran snímku zachován. Při zpracování videa na výšku je šířka výsledného snímku 720 px, výška se změní v odpovídajícím poměru.

Vzorce pro změnu velikosti snímku videa s rozlišením  $x' \times y'$  na výsledné rozlišení  $x \times y$ , reprezentující 720p jsou následující:

$$\text{Na výšku: } (x, y) = (720 \cdot x' / y', \quad 720)$$

$$\text{Na šířku: } (x, y) = (720, \quad 720 \cdot y' / x')$$

První složka udává šířku snímku, druhá výšku.

### 3.5.2 Nalezení atleta

Nalezení atleta jsem implementoval dvěma způsoby - manuálním a automatickým. O nalezení atleta se stará třída `athlete_finder`.

#### Manuální způsob

Tímto způsobem určí uživatel pozici atleta ve snímku. Vybere snímek, ve kterém je atlet v záběru, a následně ho uzavře do ohraničujícího rámečku. Tuto funkcionality v grafickém rozhraní nabízí metoda `cv::selectROI`.

## Automatický způsob

Abych nalezl atleta automaticky, procházím snímky videa postupně, udržuji si seznam postav ve videu a atleta ve videu. Postavy reprezentuje `std::list`, jehož prvky jsou instance třídy `athlete_finder::athlete_candidate`. Pro `std::list` jsem se rozhodl, jelikož se počet postav ve videu často mění a při této operaci není potřeba `std::list` realokovat. Atleta ve videu reprezentuje instance `std::optional<person>`, její hodnota není validní, dokud nenaleznu atleta.

Dokud nenaleznu atleta, opakuji při zpracování každého snímku následující postup.

Vyfiltruji seznam postav ve videu, abych v seznamu nechal jen postavy, které mohou reprezentovat atleta. Mezi témoto postavami se pokusím najít atleta. Pokud ho najdu, vytvořím instanci třídy `person` na základě snímku, ve kterém jsem ho poprvé detekoval, a jeho pozice v něm. Pokud je seznam postav prázdný a zatím se mi nepodařilo najít atleta, spustím detekci postav v právě zpracovávaném snímku a detekované postavy uložím do seznamu postav.

Postavy, jejichž trasování neselhalo, jsou celé uvnitř snímku a pohybují se, mohou reprezentovat atleta, proto je ze seznamu postav nesmažu. Pokud neuplynulo 0.3 s od první detekce postavy, považuji ji za pohybující se. Po uplynutí této doby se musela od momentu první detekce pohnout v horizontálním směru alespoň o šířku jejího ohraničujícího rámečku, abych ji považoval za pohybující se a nechal ji mezi kandidáty na atleta. Pozici postavy určuje střed jejího ohraničujícího rámečku. Pohyb postav zkoumá třída `movement_analyzer`.

Za atleta zvolím postavu, pro kterou je určen směr pohybu. Směr pohybu určí třída `movement_analyzer` jakmile se postava pohně alespoň o šířku jejího ohraničujícího rámečku v horizontálním směru.

Ohraničující rámečky postav vykreslím do snímku a tyto snímky vrátím pro následné vytvoření videa.

**Detekce postav ve snímku** Detekci postav ve snímku provádí instance třídy `cv::HOGDescriptor` pomocí knihovního SVM detektoru, který vrátí metoda `cv::HOGDescriptor::getDefaultValueDetector`. Ohraničující rámečky postav ve videu získám metodou `cv::HOGDescriptor::detectMultiScale`. Parametry této metody jsou

- právě zpracovávaný snímek,
- detekované ohraničující rámečky,
- práh detekce,
- posun okna,
- obal okna,
- škálování,
- konečný práh a
- použití seskupovacího algoritmu.

*Práh detekce* určuje vzdálenost klasifikační roviny SVM detektoru a vlastností právě zpracovávaného okna. Hodnotu tohoto parametru jsem nechal výchozí - tedy 0.

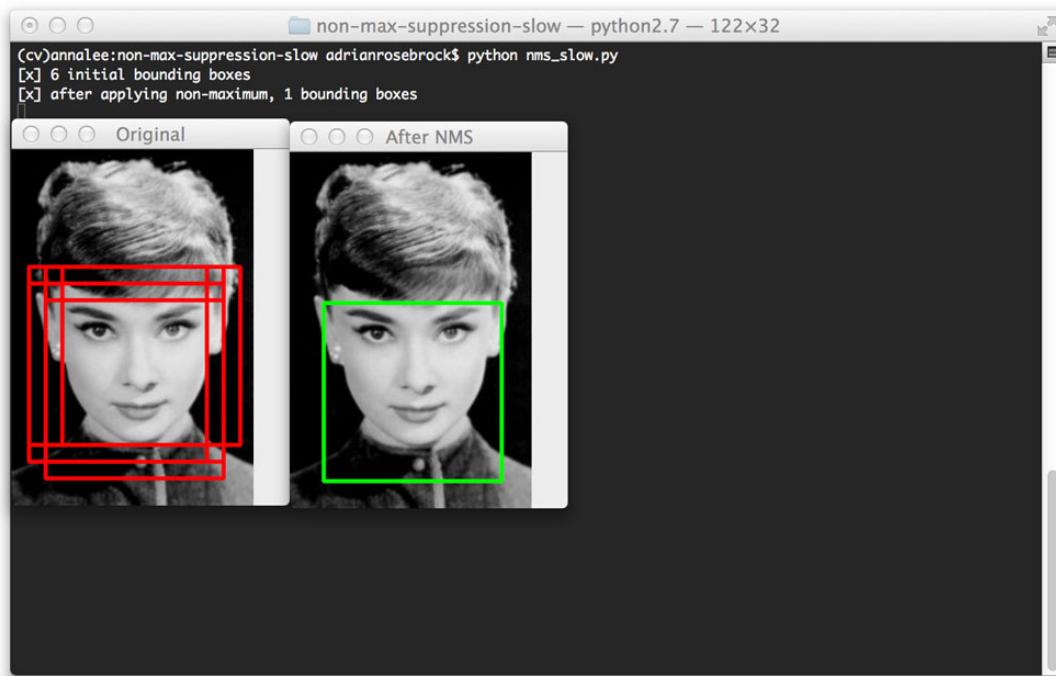
*Posun okna* určuje velikost kroku mezi okny pro detekci postav ve snímku. Velikost tohoto kroku jsem nastavil na (4,4) pixely.

*Obal okna* je počet pixelů okolo okna v obou směrech, které detektor zahrnuje do detekce. Velikost obalu jsem zvolil (16,16).

*Škálování* udává koeficient, kterým se zmenšuje vstupní snímek. Jelikož detekuj různě veliké postavy a okno pro detekci má pevnou velikost, musím měnit velikost snímku a detekci spouštět na různých velikostech. Snímek tedy po každé detekci zmenším daným koeficientem, jehož hodnotu jsem nastavil na 1.05.

*Konečný práh* není v knihovně dostatečně komentován, proto jsem ponechal výchozí hodnotu 2.

*Použití seskupovacího algoritmu* udává, zda se před vrácením ohraničujících rámečků postav má použít algoritmus pro jejich seskupení. Tento algoritmus nefungoval příliš dobře, proto seskupení rámečků implementuji sám. Používám modifikaci algoritmu non-maximum suppression (Rosebrock, 2015). Algoritmus jsem implementoval stejně, jen jsem se rozhodl seřadit rámečky podle velikosti, nikoliv podle pozice jejich spodních stran. Důvodem pro tuto změnu byla skutečnost, že metoda detekovala postavu jen na části atletova těla, tudíž pro určení výsledné detekce dávám přednost větším rámečkům. Využití algoritmu non-maximum suppression je vidět na obrázku 3.2.



Obrázek 3.2: Výsledek aplikace algoritmu non-maximum suppression při detekování tváře. Implementace podle Rosebrock (2015) zvolila rámeček, který se na snímku nachází nejníž. Má implementace by dala přednost největšímu snímku. Převzato z <https://www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/> dne 26. 5. 2021.

**Analýza pohybu postav** O analýze pohybu postav se stará třída `movement_analyzer`, stejná třída se stará i o pohyb atleta, jehož trasování je modifikované, tudíž je analýza pohybu blíže popsaná v sekci 3.5.3.

Při analýze atletova pohybu předávám třídě `movement_analyzer` vektor všech rámečků mřížky. Postavy při hledání atleta mřížku nemají, tudíž předám jedno-prvkový vektor obsahující ohraničující rámeček postavy.

### 3.5.3 Detekce kostry

Jelikož může být postav ve videu více, bylo potřeba pro přesnost detekce kostry atleta ze snímku vyříznout okno, ve kterém se atlet nachází. Pozici tohoto okna určuji trasováním atleta. Detekci kostry atleta zajišťuje instance třídy `person`, která reprezentuje atleta.

Video se zpracovává postupným procházením snímků videa. V každém snímku určím pozici atletova těla a následně detekuju klouby kostry. Do snímku zakreslím detekce a po průchodu celého videa snímky se zakreslenými detekcemi vrátím.

Pozice detekovaných kloubů atletova těla v jednotlivých snímcích udržuje instance třídy `person`. Posun snímků pro převod těchto pozic do souřadného systému videa obstarává třída `movement_analyzer`.

## Trasování atleta

Jak jsem již popisoval v sekci 2.3.3, bylo potřeba trasování atleta modifikovat, aby byla kvalita trasování při skoku dostatečná. Rozhodl jsem se pro rozdelení původního ohraničujícího rámečku atleta na mřížku  $3 \times 3$  menších rámečků. Chtěl jsem jedním z menších rámečků pokrýt střed ohraničujícího rámečku, proto jsem se rozhodl pro tento počet. Každý rámeček v mřížce trasuji samostatně, trackery jednotlivých rámečků jsou uloženy v 1D vektoru. Kvůli analýze pohybu rámečků při aktualizaci mřížky ukládám pozice rámečků do vektoru, který pro každý snímek obsahuje vektor pozic rámečků v daném snímku.

V prvním snímku, kde jsem detekoval atleta, inicializuji tracker pro každý rámeček mřížky.

Ihnad po inicializaci a v následujících snímcích určím pozici jednotlivých rámečků. Pokud trasování všech rámečků mřížky selže, ukončím trasování atleta a v následujících snímcích kostru nedetekuju.

Vzhledem k tomu, že atleta trasuji po částech, se může stát, že některé trackery začnou trasovat pozadí videa. Abych je přesunul na správné místo, zkoumám pohyb jednotlivých trackerů za poslední 2 snímky.

Vyberu rámeček, jehož pozice se za poslední 2 snímky nejvíce změnila. Kontroluji, aby horizontální směr odpovídal směru rozběhu atleta. K tomuto rámečku přesunu mřížku, aby v ní byl nejvíce se pohybující rámeček na původním místě. Musím také zajistit, aby byla mřížka uvnitř snímku. Mřížku opět rozdělím na menší rámečky a inicializuji v ní trackery, které atleta ztratily.

Tyto trackery určím tak, že posun jejich rámečků je menší než  $1/4$  posunu nejvíce se pohybujícího rámečku. Při skoku atlet švihá nohou směrem dopředu, pohyb této nohy vůči pozadí je přibližně dvojnásobný vůči pohybu trupu, tudíž jsem musel zvolit menší hodnotu než  $1/2$ , abych zamezil přesunu validních trackerů z trupu na nohu.

Pohyb aktualizovaných rámečků 2 snímky neanalyzuji, aby do jejich pohybu nebylo započteno jejich posunutí při aktualizaci.

Atletova pozice se v průběhu videa mění, snažil jsem se implementovat i změnu velikosti mřížky podle velikosti těla atleta, ale přesnost trasování se zhoršila, proto škálování mřížky neprovádím.

## Analýza pohybu atleta

Pro určení pozice atleta v souřadném systému videa využívám třídu `movement_analyzer` a `background_tracker`. Tato třída analyzuje také pohyb postav, každá instance analyzuje pohyb jediné postavy.

Atlet je ve snímku reprezentován několika rámečky, postava jedním, ale ten zabalím do vektoru a následný postup je pro oba případy stejný. Pozici postavy ve snímku rozumím střed nejmenšího obdélníku, do něhož lze rámečky uzavřít. Rámečky ve snímku, snímek a číslo snímku dostane analyzátor pohybu na vstupu.

V prvním snímku, ve kterém chci určit pozici atleta, inicializuju tracker pozadí. Část pozadí, kterou budu trasovat, uzavřu do ohraničujícího rámečku, který má velikost  $x/4 \times y/2$ , kde  $x$  je šířka snímku,  $y$  výška. Tento rámeček umístím k levé nebo pravé straně snímku, podle toho, na jaké straně je průnik atleta a pozadí menší. Přednost má levá strana. Rámeček umístím tak, aby nad ním i pod ním bylo stejně místa. Tím se vyvaruji trasování jednobarevného nebe, případně tartanu, pokud bych pozadí umístil nahoru, případně dolů.

Při zpracování každého snímku zjistím pozici levého horního rohu snímku v souřadném systému videa, díky čemuž snadno určím pozici libovolného objektu ve snímku.

Pozice pozadí ve videu je  $d_0 = b_0 - p_0$ ,  $b_0$  udává pozici středu pozadí v tomtéž snímku,  $p_0$  je pozice atleta v prvním snímku, ve kterém ho detekuji. Pozice  $t_0$  levého horního rohu prvního snímku je rovna  $-p_0 = d_0 - b_0$ .

Při analýze dalšího snímku znám pozici atleta  $p_1$  ve snímku. S využitím trackeru určím pozici pozadí  $b_1$  ve snímku a jeho pozici  $d_1 = d_0$  ve videu. Ze znalosti pozice pozadí získám pozici levého horního rohu snímku  $t_1 = d_1 - b_1$ . Takto získám pozici levého horního rohu každého snímku.

Jelikož se záběr kamery pohybuje, je potřeba pozadí aktualizovat, jakmile se dostane mimo záběr. S touto aktualizací musím aktualizovat pozici nového pozadí ve videu. Mám tedy pozadí, které se dostalo mimo záběr. Jeho pozice ve videu je  $d_i$ , pozice ve snímku  $b_i$ . Inicializuju nové pozadí s pozicí  $n_i$  ve snímku. Pozice nového pozadí ve videu se určí jako  $d_i + n_i - b_i$ . Dále pracuji s touto hodnotou místo  $d_i$  a novým pozadím videa.

Tracker pozadí znovu inicializuju, jakmile do oblasti, která ho reprezentuje, vběhne atlet. Pokud pozadí bez průniku s atletem nelze vytvořit, nezacyklím se, jen budu v každém snímku vytvářet nové pozadí, aby byl průnik minimální.

Podle pohybu atleta ve videu také určím směr jeho rozběhu pro následný převod kostry do 3D modelu. Směr rozběhu určím, jakmile se pozice atleta změní alespoň o šířku jeho těla od první pozice, kde jsem atleta detekoval.

## Detekce kloubů těla

Klouby těla jsou určeny databází MPII Human Pose dataset (Andriluka a kol., 2014). Jedná se o 16 bodů těla, ale posledním je pozadí, které mohu ignorovat.

Používám tedy jen prvních 15 bodů těla. Jedná se o

0. hlavu,
1. krk,
2. pravé rameno,
3. pravý loket,
4. pravé zápěstí,
5. levé rameno,
6. levý loket,
7. levé zápěstí,
8. pravou kyčel,
9. pravé koleno,
10. pravý kotník,
11. levou kyčel,
12. levé koleno,
13. levý kotník a
14. hrudník.

O detekci kloubů těla se stará třída `parts_detector`. Ta dostane pro detekci snímek, rámečky atletova těla v něm a maximální vzdálenost, o kterou může posunout střed okna od středu atletova těla, což je střed nejmenšího rámečku, do něhož se vejdou rámečky atletova těla. Maximální vzdálenost je určena průměrnou vzdáleností dvojic středů rámečků atleta.

Detektor si pamatuje poslední velikost kostry, posun středu okna a úhel naklonění trupu. Detekci zkouší na natočení proti poslednímu známému úhlu trupu a následnému natočení o 20 stupňů oběma směry. Tyto detekce probíhají postupně a vybere se ta, která detekuje nejvíce kloubů těla. Pokud nějaká detekce uspěje na 100 %, další se nezkouší.

Úhel naklonění trupu určuji jako úhel mezi úsečkou spojující střed kyčlí a hlavu a vertikálou, která prochází středem kyčlí. Úhel reprezentuje míru natočení trupu oproti vzpřímené pozici po směru hodinových ručiček, na rozdíl od analýzy parametru naklonění trupu, kde předklon znamená kladný úhel náklonu trupu. Hodnoty jsou v rozmezí  $[-180, 180]$  stupňů. Metoda `cv::rotate` otáčí snímek o úhel proti směru hodinových ručiček, tudíž bude v otočeném snímku atletův trup vzpřímený, když mu předám nezměněnou hodnotu úhlu naklonění trupu.

Výpočet úhlu naklonění trupu provádím následující metodou:

```

std::optional<double> get_vertical_tilt_angle(
    const frame_point &a,
    const frame_point &b) noexcept {
    if (a && b) {
        cv::Point2d v1(0, -1); // Point straight up.
        cv::Point2d v2 = *b - *a;
        double dot = v1.x * v2.x + v1.y * v2.y;
        double val = dot / (cv::norm(v1) * cv::norm(v2));
        double mult = 1;
        if (v2.x != 0)
            mult = v2.x / std::abs(v2.x);
        return mult * std::acos(val) * 180.0 / M_PI;
    }
    return std::nullopt;
}

```

Prvním argumentem je pozice středu kyčlí, druhým pozice hlavy atleta. Pro výpočet využívám vlastnosti skalárního součinu, díky kterému lze určit úhel mezi vektory. Při výpočtu vycházím ze vzorce  $\langle u, v \rangle = \|u\| \cdot \|v\| \cdot \cos \alpha$ , kde  $\alpha$  je úhel, který vektory svírají.

Pro detekci kloubů těla nejprve vyberu rámeček okna pro detekci. Jeho střed získám ze středu rámečků atleta a posunu středu z minulého snímku. Rámečky atleta aktualizují každé 2 snímky, tudíž jejich posunutí při aktualizaci není velké. [OBRÁZEK] Velikost jeho strany určím z velikosti dosud největší detekované kostry, kterou zvětším koeficientem 1.3. Pokud je velikost kostry dosud neznámá, zvětším 1.3 krát nejmenší rámeček obsahující rámečky atletova těla. Škálování rámečků provádí tak, abych zachoval jejich střed a aby se vešly do snímku.

Následně kolem středu zvoleného rámečku okna otočím snímek proti poslednímu známému naklonění trupu, k tomuto úhlu případně přičtu úhel dalšího naklonění, pokud se detekce nepodaří napoprvé. K natočení snímku využiji metodu `cv::rotate`. Z natočeného snímku vyříznou část, která je určena oknem. Střed okna zůstal po rotaci na stejném místě, takže získám jen natočené okno. [OBRÁZEK] Toto okno zmenší na rozlišení, jehož menší strana má 150 px, aby detekce probíhala rychleji.

Zmenšené okno předám jako vstup konvoluční sítě a spustím detekci. Z výsledku detekce extrahuji klouby těla do nezmenšeného okna, ty podle jeho pozice a naklonění posunu na správnou pozici do snímku.

Nakonec aktualizují velikost dosud největší detekované kostry, pokud detekují všechny části těla a tato kostra je největší. Také aktualizují posun středu okna oproti středu rámečku ohraničujícího atleta. Tento posun zmenším, pokud je větší než maximální vzdálenost. Jako poslední aktualizují úhel naklonění trupu, pokud se od posledního známého liší o méně než 30 stupňů. Tento úhel určuje střed kyčlí a hlava atleta vůči vertikále, pokud byly tyto části těla v předchozím snímku detekovány, jinak úhel neaktualizují.

**Sítě** K detekci kloubů těla používám konvoluční síť, která je implementovaná ve frameworku Caffe (Jia a kol., 2014). Tuto síť načtu ze souborů metodou `cv::dnn::readNet`. Váhy sítě a její parametry jsem použil z projektu OpenPose (Cao a kol., 2021). Vstupem této sítě může být soubor libovolně velkých obrázků. Výstupem sítě je 4-dimenzionální matice `cv::Mat`, dimenze popisují

- číslo snímku,
- číslo výstupní vlastnosti,
- výšku výstupu a
- šířku výstupu.

**Extrakce kloubů z výstupu** Číslo snímku je irrelevantní, jelikož předávám síti vždy jen jeden snímek. Výstupních vlastností je pro MPII model 44, ale používám jen prvních 15, které reprezentují klouby těla. Z výstupu dostanu matici pravděpodobností kloubu  $n$  konstruktorem `cv::Mat`, kterému dám výšku, šířku, typ prvků a ukazatel na výslednou matice, který dostanu z výstupu sítě. Ze získané matice vyberu pozici prvku, který má maximální pravděpodobnost. Pokud je tato pravděpodobnost větší než 0.1, převedu pozici tohoto bodu ve výstupu na odpovídající pozici ve vstupu. Pokud jsou pravděpodobnosti ve všech místech výstupu nanejvýš 0.1, nastavím detekovanému bodu nedefinovanou pozici. Výslednou pozici určím výpočtem  $(x,y) = (x_{out} \cdot w_{in}/w_{out}, y_{out} \cdot h_{in}/h_{out})$ .  $(x_{out},y_{out})$  je pozice na výstupu,  $(w_{in},h_{in})$  velikost vstupu a  $(w_{out},h_{out})$  velikost výstupu.

Podle pozice a natočení okna ve snímku určím pozici výsledných bodů ve snímku. Bod posunu o levý horní roh okna, tím získám pozici bodu v otočeném snímku. Otočením bodu podle středu okna o úhel natočení okna zpět získám pozici bodu v nerotovaném snímku. Pro otočení bodu využiji metodu `cv::transform`.

[OBRÁZEK]

Pozice kloubů reprezentuji jako instance `cv::optional<cv::Point2d>`, tedy pozice kloubů, které se mi nepodaří nadetkovat, nejsou definované.

### 3.5.4 Konverze kostry do 3D modelu

O převod kostry do 3D modelu se stará konstruktor třídy `model`. Konstruktor dostane instanci třídy `person`, která reprezentuje atleta a cestu ke zpracovanému videu.

Nejprve se převedou klouby kostry do 3D, aby se daly body sčítat. Výsledkem konverze bodu  $(x,y)$  je bod  $(x,0,y)$ . Stejná operace se provede s pozicemi levého horního rohu jednotlivých snímků.

Následně se díky témtu výsledkům určí reálná pozice bodů. Mám tedy kloub kostry  $(x,y,z)$  a posun snímku  $(x',y',z')$ . Výsledná pozice je  $(x+x',y+y',z+z')$ , pokud atlet běží doprava,  $(-x-x',y+y',z+z')$  pokud atlet běží doleva.

Posunutí počátku systému do pozice kotníku odrazové nohy v momentu odrazu se provádí po analýze parametrů odečtením této hodnoty.

### 3.5.5 Analýza parametrů

Pro analýzu parametrů skoku používám pozice kloubů těla v každém snímku. Tyto hodnoty mám uložené ve vektoru vektorů, jehož prvky jsou instance `std::optional<cv::Point3d>`. První dimenze určuje číslo snímku, druhá číslo kloubu. I když atleta nedetekuji v celém videu, mám pro každý snímek uloženy pozice kloubů. Pokud jsem v daném snímku kloub nedetkoval, je jeho hodnota nedefinovaná (`std::nullopt`).

Detekce kloubů těla není úplně přesná, často při detekci prohodí levá a pravá noha, tuto chybu je potřeba brát při analýze parametrů v potaz. [GRAF levého kotníku, pravého kotníku a nižšího kotníku]

O analýzu parametrů se stará třída `vault_analyzer` a struktury reprezentující jednotlivé parametry, pro analýzu používám souřadný systém 3D prostoru.

## Důležité momenty skoku

**Start** Start rozběhu je snímek, ve kterém se začne pohybovat atletův kotník. Postupně zkoumám pozice kotníků ve všech snímcích videa a udržuji si předešlou pozici levého a pravého kotníku. Jakmile se pozice levého nebo pravého kotníku změní o více než 1 pixel, ukončím procházení a vrátím číslo snímku, ve kterém pohyb začal - jedná se předposlední snímek, který jsem zkoumal. Změna pozice kotníků, kterou určuji při zpracování prvního snímku není definovaná, tudíž zpracuji vždy alespoň 2 snímky a výsledné číslo snímku je validní.

O určení momentu startu rozběhu se stará metoda `vault_analyzer::find_start`:

```
std::optional<std::size_t> find_start(const model_video_points &points)
noexcept {
    std::size_t index = 0;
    model_point left = std::nullopt;
    model_point right = std::nullopt;
    for (const auto &body : points) {
        std::optional<double> dist =
            distance(body[body_part::l_ankle], left);
        if (dist && *dist > 1) break;
        dist = distance(body[body_part::r_ankle], right);
        if (dist && *dist > 1) break;
        left = body[body_part::l_ankle];
        right = body[body_part::r_ankle];
        ++index;
    }
    if (index && index != points.size())
        return index - 1;
    return std::nullopt;
}
```

Funkce `distance` funguje stejně, jako metoda `cv::norm` aplikovaná na rozdíl argumentů, pokud jsou oba argumenty validní, jinak vrátí `std::nullopt`.

**Odraž** Moment odrazu je snímek, ve kterém dochází k odrazu od země při posledním detekovaném kroku. Popis detekce kroků popisují v následující sekci.

**Kulminace** Moment kulminace nad lafkou je určen snímkem, ve kterém je pozice středu kyčlí nejvýš v celém videu.

## Doba trvání jednotlivých kroků

Pro určení doby trvání kroků je potřeba určit specifický moment každého kroku. Zvolil jsem určení momentu odražení od země jednotlivých kroků, tato implementace mi pomohla s detekcí momentu odrazu.

Jelikož si nemohu být jistý pozicí levé a pravé nohy, musím zkoumat pozici nižšího kotníku. Postupně procházím pozice nižšího kotníku v jednotlivých snímcích videa. Určím momenty, ve kterých nižší kotník opouští lokální minimum.

Jelikož detekce nemusí být vždy přesná, musím určit práh, jehož překonání znamená vertikální posunutí kloubu kostry. Velikost tohoto prahu musí záviset na velikosti těla atleta a frekvenci snímků videa. Velikost prahu určuji takto:  $x \cdot s/fps$ , kde  $s$  je vzdálenost dvou nejvzdálenějších kloubů těla atleta,  $fps$  frekvence snímků videa a  $x$  vhodná hodnota. Po analýze vertikálních pohybů nižšího kotníku v momentech odrazů jednotlivých kroků několika videí, jejichž frekvence snímků byla 60 fps, jsem hodnotu  $x$  nastavil na  $0.023 \cdot 60 = 1.38$ . Zvolil jsem nejmenší vertikální posun v momentech odrazů jednotlivých kroků v analyzovaných videích. Jednalo se o videa 1.MOV, 8.MOV a 12.MOV (viz tabulka 4.1).

Možným vylepšením by bylo zkoumání levé a pravé nohy a jejich vzájemné pozice - která je vepředu a která vzadu - ale jelikož není detekce levé a pravé nohy občas správná, musel jsem se rozhodnout pro výše popsanou implementaci.

Pro přesnost určení kroků je potřeba vyfiltrovat snímky, ve kterých nedochází ke kroku, příkladem je chvíle po odrazu, při níž atlet švihá odrazovou nohou dopředu. Odrazová noha se tedy po odrazu pohybuje vzhůru a následně dolů (při švihu), což může být z hlediska algoritmu vnímáno jako krok. [OBRÁZEK]

Budu postupně procházet snímky, ve kterých nižší kotník opouští lokální minimum, označím je jako snímky odrazů. Mezi následujícími snímky odrazů najdu nejvyšší pozici nižšího kotníku. Následující snímek odrazu označím za krok jen v případě, že se v něm nižší kotník nachází pod průměrem výšky nalezeného maxima nižšího kotníku a výšky nižšího kotníku v momentu předešlého odrazu. První snímek odrazu ve výsledném seznamu nechám. [OBRÁZEK]

Abych nedetekoval kroky po provedení skoku, ukončím filtrace (a další kroky do výsledku nepřidám) jakmile narazím na snímek, který reprezentuje odraz při kroku a nižší kotník se v něm nachází výš, než je nejvyšší pozice nižšího kotníku mezi prvním a druhým krokem. [OBRÁZEK]

Podle frekvence snímků videa a vyfiltrovaných čísel snímků, v nichž dochází k odrazu při krocích, určím dobu trvání kroků. Doba trvání  $i$ -tého kroku je tedy  $d_i = (k_{i+1} - k_i) \cdot fps$ ,  $k_i$  je číslo snímků  $i$ -tého odrazu kroku a  $fps$  je frekvence snímků videa.

## Ztráta rychlosti ramen a boků

Pro výpočet ztráty horizontální rychlosti ramen a boků využívám střed ramen a kyčlí atletova těla. Nejprve určím pozici středu boků 0.1 s před momentem odrazu, při odrazu a 0.1 s po momentu odrazu.

Pokud byly úspěšné detekce ve všech popsaných momentech a mám jistotu, že nebudu dělit nulou, spočítám výslednou ztrátu rychlosti vzorcem  $l = (1 - (a - d)/(d - b)) \cdot 100$ , kde  $a$  je hodnota první složky pozice zkoumané části těla 0.1 s po odrazu,  $d$  určuje stejnou hodnotu v momentu odrazu a  $b$  před odrazem. Výslednou ztrátu určuji v procentech.

Pokud hodnotu výrazu nelze spočítat, má tento parametr nedefinovanou hodnotu.

## Výška boků

V průběhu celého videa určuji výšku boků. Pro každý snímek uložím hodnotu průměru výšky levé a pravé kyčle. Pokud se ve snímku podařilo detektovat jen jednu, uložím výšku detekované, pokud se nepodaří detektovat ani jednu kyčel ve snímku, je hodnota výšky boků v tomto snímku nedefinovaná.

## Úhel odrazu

Úhel odrazu určuji jako rozdíl úhlu pohybu na začátku skoku a úhlu pohybu na konci rozběhu. Úhly pohybu udávají úhel mezi  $x$ -ovou osou a směrem pohybu středu kyčlí atleta.

Pro výpočet úhlu odrazu získám pozici středu kyčlí 0.1 s před momentem odrazu, při odrazu a 0.1 s po odrazu. Z pozice kyčlí při odrazu a po něm určím úhel, pod kterým se pohybuje atlet při skoku. Na základě pozice kyčlí při odrazu a před ním vypočítám úhel pohybu před odrazem.

Kladná hodnota úhlu odrazu znamená odraz vzhůru, záporná odraz dolů. Hodnoty jsou v intervalu  $[-180, 180]$  stupňů.

## Úhel došlapu jednotlivých kroků

Algoritmem, kterým získávám momenty došlapů pro výpočet oporové fáze kroků, získám momenty došlapů všech kroků. V nich určím úhel mezi vertikálou a úsečkou, kterou určuje kotník a kyčel došlapující nohy.

Jedná se o míru došlapu nohy před kyčel, při došlapu před kyčel je tedy hodnota kladná, při došlapu za kyčel je hodnota záporná. Hodnoty se pohybují mezi  $-180$  a  $180$  stupni.

## Náklon trupu

Náklon trupu je úhel mezi vertikálou a úsečkou, kterou určuje pozice středu kyčlí a pozice hlavy. Tento úhel vypočítám pro každý snímek videa.

Úhel určuji stejně, jako v případě detekce kostry (sekce 3.5.3), jen se jedná o úhel v trojrozměrném prostoru. Jedná se o míru naklonění, jejíž hodnota je kladná, pokud je atlet v předklonu a záporná, pokud je v záklonu. Směr rozběhu je ve směru rostoucí  $x$ -ové souřadnice.

Hodnota naklonění trupu se pohybuje v rozmezí  $-180$  až  $180$  stupňů.

## 3.6 Analýza modelu

O načtení modelu ze souboru se stará třída `model`, podle uložených pozic kloubů kostry vůči levému hornímu rohu snímku a posunu snímku ve videu se klouby převedou do 3D modelu. Tento postup je stejný jako při převodu detekované kostry do 3D modelu. Tento model je na rozdíl od konverze z detekované kostry posunut, aby byl počátek systému v místě kotníku odrazové nohy v momentu odrazu.

Pro následnou analýzu parametrů se využívá model ve 3D souřadném systému stejným způsobem, jako při analýze videa.

## 3.7 Výstup programu

### 3.7.1 Uložení výstupu

Výstup ukládám do textových souborů, které jsou uloženy ve složce `outputs/name/`, kde `name` je název zpracovávaného videa (bez cesty a přípony). Parametry ukládám do souboru `parameters.csv`, model do souboru `model.txt`.

Uložení parametrů provádí třída `vault_analyzer`, uložení modelu má na starosti třída `model`. Pro vypisování hodnot do souboru používám `std::ofstream`.

Struktura těchto souborů je popsána v sekci 2.5.1.

Kromě těchto textových souborů generuje program trojici videí, kterou uloží do stejné složky jako textové soubory.

Prvním z nich je video bez detekcí, obsahuje původní snímky videa, která program zmenšíl před analýzou. Název tohoto videa je `raw.avi`.

Druhé video se generuje jen v případě, že uživatel zvolí automatickou detekci atleta, obsahuje postavy, které program v každém videu bral v potaz při hledání atleta. Jedná se o video `found.avi`.

Třetí video obsahuje zakreslenou kostru, rámečky těla atleta a pozici trasovaného pozadí. Toto video se zapíše do souboru `detections.avi`.

### 3.7.2 Zobrazení výstupu uživateli

O funkcionalitu zobrazení výstupu uživateli se stará třída `viewer`. Ta otevře prohlížeč snímků videa, zobrazí důležité momenty skoku a následně zobrazí grafy obsahující hodnoty parametrů.

#### Prohlížeč snímků

Prohlížeč snímků spustím zavoláním metody `viewer::show`. Jejími parametry jsou snímky videa s detekcemi, bez detekcí a instance třídy `vault_analyzer`, která analyzovala atletův pohyb.

Prohlížeč si udržuje číslo právě zobrazovaného snímku, při jeho změně kontroluje, aby se jeho hodnota nedostala mimo hodnoty, kterými lze přistupovat k jednotlivým snímkům.

V nekonečné smyčce provádí program následující kroky:

Snímek zobrazím metodou `cv::imshow`, která vytvoří okno a v něm zobrazí snímek podle čísla právě zobrazovaného snímku.

Spolu se zobrazením snímku vypíše program do konzole číslo snímku, dobu snímku vůči odrazu a hodnoty parametrů, které s daným snímkem souvisí. [TABULKA]

Následně se čeká na vstup uživatele s využitím metody `cv::waitKey`. Tato metoda vrací kód klávesy, kterou uživatel stiskl, a čeká na vstup uživatele neomezeně dlouhou dobu.

Levou a pravou šipkou se uživatel pohybuje mezi snímky videa dopředu, případně dozadu, mezerníkem uživatel zobrazí a opět skryje zakreslené detekce kostry atletova těla. Stiskem klávesy `esc` se zobrazí grafy s hodnotami parametrů a ukončí se prohlížení snímků videa přerušením smyčky.

## Grafická reprezentace parametrů

Zobrazení parametrů v grafické podobě zajišťuje skript `plot_parameters.py`, který načte hodnoty parametrů ze souboru, který se vygeneroval při analýze parametrů.

Skript se spouští přímo z kódu jazyka C++, což zajišťuje API pro použití Pythonu prostředky, kterými disponuje jazyk C (Foundation, 2021). O spouštění se stará třída `viewer`.

Kód, který spouští skript pro zanesení parametrů do grafů vypadá následovně:

```
void show_parameters() const noexcept {
    Py_Initialize();

    FILE *file = fopen("plot_parameters.py", "r");
    if(file) {
        wchar_t *argv[3] = {
            Py_DecodeLocale("plot_parameters.py", NULL),
            Py_DecodeLocale("--file", NULL),
            Py_DecodeLocale(params_filename.c_str(), NULL)
        };
        PySys_SetArgv(3, argv);
        PyRun_SimpleFile(file, "plot_parameters.py");
        fclose(file);
    }

    Py_Finalize();
}
```

Kromě vygenerování grafů programem je možné skript pro zanesení parametrů do grafu spustit zvlášt. Skript očekává jediný argument `-file`, který následuje cesta k souboru obsahujícímu detekované hodnoty parametrů.

Pro podrobnější analýzu je možné využít pouze funkci `read_file` ve vlastním skriptu. Tato funkce dostane na vstupu parametr určující cestu k souboru s detekovanými parametry a vrací název analyzovaného videa, názvy parametrů, jednotky parametrů a hodnoty parametrů.

Názvy a jednotky parametrů jsou uloženy jako seznamy řetězců, hodnoty jako seznam seznamů čísel. První rozměr hodnot určuje parametr, druhý pořadové číslo hodnoty příslušného parametru (například číslo snímku nebo číslo kroku). Prvním parametrem, který funkce vrátí je relativní čas vůči odrazu v sekundách.

# 4. Experimenty

## 4.1 Sběr dat

Funkčnost programu jsem průběžně ladil na několika videích z mého archivu. Pro potřeby korektního otestování funkčnosti a specifických vlastností programu jsem natočil další videa.

Testovací videa obsahují 46 videí, které zachycují závodní a tréninkové skoky a tréninkové nácviky skoků, videa jsou natočena z různých úhlů, stran a vzdáleností. Na videích je zachyceno 5 atletů (3 muži a 2 ženy). K natočení videí byly využity mobilní telefony Apple iPhone SE (2020) a Samsung Galaxy S10+. Většina videí je natočena v rozlišení Full HD ( $1920 \times 1080$  px), jedno z videí je po komprimaci, která nastane při poslání videa přes aplikaci WhatsApp. Videá natočená telefonem Apple iPhone SE (2020) mají frekvenci snímků 60 fps, frekvence snímků videí natočených Samsungem Galaxy S10+ je 30 fps. Vliv frekvence snímků videa při analýze je popsán v sekci 4.8.

Seznam videí s příslušnými vlastnostmi je popsán v tabulce 4.1.

Hlavní část testovacích dat jsem pořídil na soustředění na ostrově Tenerife. Počasí bylo po celou dobu pobytu příznivé pro skok o tyči, jelikož svítilo slunce. Z hlediska testovacích dat to byla škoda, protože tato část testovacích dat neobsahuje videa v různých světelných podmínkách.

Jedná se o 32 videí, která zachycují skoky tří atletů a dvou atletek. Kromě samotných skoků jsou na videích zachyceny i jiné tréninkové prostředky, mezi které patří rozběh s naznačením odrazu (např. video 2.MOV), průchod (pouhý odraz na tyč, tělo zůstává ve víceméně vzpřímené pozici) (např. video 1.MOV) nebo náskok do ohybu (průchod, kdy se atlet vrací zpět na rozběh s pomocí trenéra) (např. video 3.MOV). Skoky jsem natáčel z různých stran, úhlů a vzdáleností.

Jak jsem popisoval v sekci 2.2, závodní skoky jsou typicky natáčeny z mnohem větší vzdálenosti. Mobilní telefony nedisponují kvalitním přiblížením při natáčení videa, tudíž je atlet na videu mnohem menší. Navíc se na závodech pohybuje více atletů, než na tréninku, i z tohoto důvodu jsem mezi testovací videa zahrnul mě závodní skoky - 1 soutěžní skok z Mistrovství České republiky mužů a žen v hale 2021 a 4 rozsvičovací a 5 soutěžních skoků ze Zlaté tretry 2021.

Mezi testovací videa jsem dále zahrnul 4 tréninkové skoky z atletické haly na pražském Strahově, mezi kterými jsou 2 skoky, které zachycují zvrat na rovné tyči (otočení atleta vzhůru nohama bez ohnutí tyče) (např. video 33.MOV).

Úvodní pozice atleta v testovacích videích, které jsem pro experimenty používal, jsem volil manuálně. Jejich popis je uveden v tabulce ??.

## 4.2 Hledání atleta ve videu

Při spuštění automatického hledání atleta ve všech testovacích videích se atletovo tělo nepodařilo nalézt ve dvou videích (5.MOV a 40.mp4). V jednom videu označil program za atleta osvětlení stadionu (video 11.MOV) kvůli chybné analýze pohybu pozadí, které trasovalo postavu.

Velikosti nalezených atletů se lišily. V některých videích našel program jen část atletova těla - úvodní rámeček se nacházel jen na trupu, případně nohách

	A	T	V	DT	VÚ	O	DR	KR	KV	D1	D2	VK	S	R	FPS	K
1.MOV	M3	1		490	445	370	29	12	11	7	350	160	L	1920x1080	60	1
2.MOV	M3	7		490	445	380	29	12	11	7	350	160	L	1920x1080	60	1
3.MOV	Ž2	6		400	360	300	8	4	4	7	250	160	L	1920x1080	60	1
4.MOV	M1	2		460	320	220	9	4	4	7	350	160	L	1920x1080	60	1
5.MOV	Ž1	1		370	325	270	14	8	8	7	350	160	L	1920x1080	60	1
6.MOV	M1	2		460	320	220	9	4	4	7	350	160	L	1920x1080	60	1
7.MOV	Ž2	7		400	340	260	18	10	10	7	350	160	L	1920x1080	60	1
8.MOV	Ž1	1		370	325	260	14	8	6	7	-80	170	L	1920x1080	60	1
9.MOV	M1	2		460	320	220	9	4	4	7	250	170	L	1920x1080	60	1
10.MOV	Ž2	1		400	340	260	18	10	10	7	850	170	P	1920x1080	60	1
11.MOV	Ž1	1		370	325	250	14	8	8	7	300	170	P	1080x1920	60	1
12.MOV	M1	2		460	330	230	9	4	4	7	300	170	P	1920x1080	60	1
13.MOV	Ž1	3		370	325	250	14	8	8	5	250	170	P	1080x1920	60	1
14.MOV	M1	4		460	330	230	9	4	4	7	300	170	P	1920x1080	60	1
15.MOV	Ž1	3		370	325	250	14	8	8	7	350	170	L	1920x1080	60	1
16.MOV	M1	4		460	330	240	9	4	4	7	350	170	L	1920x1080	60	1
17.MOV	Ž1	3		370	325	245	14	8	8	7	350	170	L	1920x1080	60	1
18.MOV	Ž1	3		370	325	250	14	8	8	7	350	170	L	1920x1080	60	1
19.MOV	Ž1	3		370	325	245	14	8	8	7	350	170	L	1920x1080	60	1
20.MOV	Ž2	5	350	400	340	275	18	10	5	7	-50	170	L	1920x1080	60	1
21.MOV	M1	1		460	380	315	14	6	6	7	300	170	L	1920x1080	60	1
22.MOV	M1	1		460	380	315	14	6	6	7	300	170	L	1920x1080	60	1
23.MOV	M1	5	380	460	380	330	14	6	6	7	300	170	L	1920x1080	60	1
24.MOV	M3	1		460	405	330	15	6	6	7	300	170	P	1920x1080	60	1
25.MOV	M3	1		490	450	370	29	12	12	10	350	170	P	1920x1080	60	1
26.MOV	M3	0		490	450	350	29	12	12	7	350	170	L	1920x1080	60	1
27_L.mp4	M3	5	490	490	450	370	29	12	12	7	300	170	L	1920x1080	30	2
27_P.MOV	M3	5	490	490	450	370	29	12	12	7	350	170	P	1920x1080	60	1
28_odraz.mp4	M3	5	490	490	450	365	29	12	12	7	300	170	P	1920x1080	30	2
28_stajony.MOV	M3	5	490	490	450	365	29	12	12	7	-80	170	P	1920x1080	60	1
29.MOV	M3	5	490	490	450	360	29	12	12	7	-80	170	L	1920x1080	60	1
30.mp4	M3	8	532	500	465	390	36	16	16	14	200	250	P	1920x1080	30	2
31.MP4	M3	1		460	405	330	15	6	6	7	300	170	P	1280x704	30	2
32.MOV	M2	5	450	460	420	300	13	8	8	5	230	170	P	1920x1080	60	1
33.MOV	M2	4		445	290	220	7	4	4	5	250	170	P	1920x1080	60	1
34.MOV	M2	4		445	290	230	7	4	4	4	350	170	P	1920x1080	30	1
35.MOV	M2	5	430	445	390	270	13	8	8	5	250	170	P	1920x1080	60	1
36.mp4	M3	9	480	490	455	370	33	14	14	25	600	270	L	1920x1080	30	2
37.mp4	M3	9	470	500	455	390	33	14	14	25	300	350	L	1920x1080	30	2
38.mp4	M3	9	550	500	455	370	33	14	11	25	550	230	L	1920x1080	30	2
39.mp4	M3	9	530	500	455	400	33	14	14	25	600	300	L	1920x1080	30	2
40.mp4	M3	8	500	500	455	390	33	14	13	25	600	270	L	1920x1080	30	2
41.mp4	M3	8	540	500	460	390	33	14	14	25	250	350	L	1920x1080	30	2
42.mp4	M3	8	540	500	460	400	33	14	14	25	100	350	L	1920x1080	30	2
43.mp4	M3	8	540	500	460	380	33	14	14	25	900	350	L	1920x1080	30	2
44.mp4	M3	8	520	500	460	390	33	14	14	25	250	350	L	1920x1080	30	2

Obrázek 4.1: Testovací videa.

A - atlet ( $M_i$  : muž  $i$ ,  $Ž_i$  : žena  $i$ ), T - typ skoku (0 : rozběh, 1 : průchod, 2 : průchod na rovné tyči, 3 : zvrat, 4 : zvrat na rovné tyči, 5 : skok, 6 : náskok do ohybu, 7 : proběhnutí, 8 : závodní skok, 9 : závodní skok rozvážení), V - výška latky [cm], DT - délka tyče [cm], VÚ - výška úchopu [cm], O - vzdálenost místa odrazu od zadní hrany zasouvací skříňky [cm], DR - délka rozběhu od zadní hrany zasouvací skříňky [m], KR - počet kroků, KV - počet kroků na videu, D1 - vzdálenost kamery od osy rozběžiště [m], D2 - vzdálenost kamery od zadní hrany zasouvací skříňky [cm], VK - výška kamery nad zemí [cm], S - strana, ze které je video natočeno (P : pravá, L : levá), R - rozlišení videa [px×px], FPS - frekvence snímků videa, K - kamera (1 : Apple iPhone SE (2020), 2 : Samsung Galaxy S10+)

atleta. V některých videích se atletovo tělo detekovalo větší, než bylo třeba. Vliv velikosti úvodních rámečků rozebírám v sekci 4.3.2.

Atletovo tělo program po nalezení ztratil jen v nepatrém množství případů. Dokonce ve videích, ve kterých kvůli nevhodnému pozadí selhávalo trasování atleta vybraného manuální cestou. Jelikož se pohyb nalezeného atleta trasuje, je značná šance, že při inicializaci postavy v prvním snímku dojde k chybě trasování a začnou se hledat postavy nové. Problémem tohoto přístupu může být to, že atlet nebude trasován po celou dobu rozběhu, ale jen v jeho konci.

## 4.3 Volba úvodní pozice atleta

### 4.3.1 Pozice ohraničujícího rámečku

Pozice úvodního ohraničujícího rámečku nemá na detekci příliš velký vliv. Část mých testovacích dat je specifická tím, že na druhé straně rozběžiště stojí trenér a při posunutí úvodního rámečku atleta směrem dozadu se okno pro detekci kostry posouvá více za atleta. Po odrazu, kdy se atletovy nohy pohybují dopředu, obsahuje okno z větší části trenéra, čímž dochází k nežádoucí detekci jeho těla. U ostatních posunutí k témtoto nežádoucím detekcím dochází méně.

Rámečky na těle atleta většinou neobsahují celé tělo, jelikož se atlet přibližuje a jejich velikost se nemění. Nejčastěji jsou umístěny na atletově trupu a při zvratu je jejich část pod trupem. Tento problém je možné vyřešit mírným posunutím úvodního rámečku směrem nad nebo před atletovo tělo. Tím se při zvratu atleta rámečky inicializují na části těla, která se pohybuje dopředu a nahoru. Toto posunuté může ovšem zvýšit míru nežádoucích detekcí, které jsem popsal v předchozím odstavci.

Pohyb některých částí těla při posunutí úvodního rámečku je vidět na obrázku ??.

### 4.3.2 Velikost ohraničujícího rámečku

Velikost úvodního ohraničujícího rámečku má na detekci značný vliv. Pokud uživatel zvolí příliš malý rámeček, nemusí se do okna určeného pro detekci kostry vejít celé tělo atleta. Aktualizace velikosti detekčního okna probíhá pouze v případě, že se detekuje celé tělo atleta. Snažím se tak omezit zbytečné zvětšování okna v případech, kdy se detekuje více postav, které nejsou v okně celé a atlet není vidět příliš dobře.

Atlet se v průběhu videa typicky přibližuje, čímž se jeho pozice ve videu zvětšuje, ale velikost mřížky zůstává stejná jako na začátku. To může zapříčinit jen částečné detekce atletova těla a tedy se okno pro detekci nikdy nezvětší.

Pokud uživatel zvolí naopak větší rámeček, může se stát, že se trackery mřížky chytí ostatních objektů. Pokud je video pořízené dostatečný čas před začátkem atletova rozběhu, mohou se trackery pohybovat s postavami v pozadí a ztratit tak atleta hned na začátku.

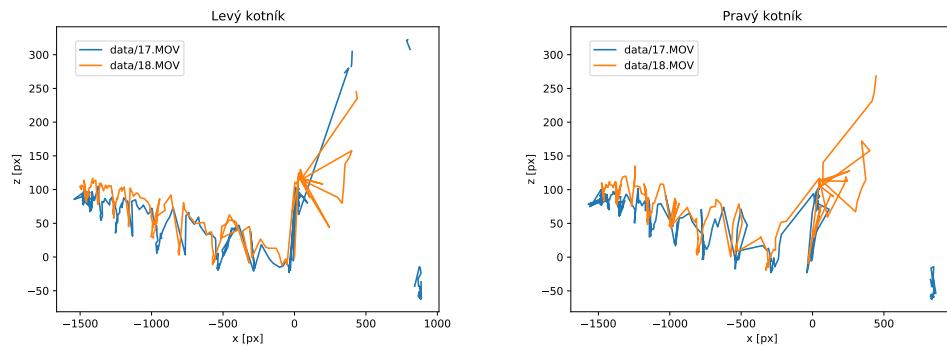
Příliš velký úvodní rámeček může způsobit, že detektor kostry detekuje kostru jiné postavy nebo smíchá klouby různých postav v jednu kostru. [OBRÁZEK] Tyto detekce neovlivní chování trackeru, ale značně znepřesňují analýzu skoků. Toto chování blíže popisují v sekci 4.6.

## 4.4 Pohyb kamery

### 4.4.1 Vertikální pohyb kamery

Vertikální pohyb kamery nemá na výsledné hodnoty parametrů téměř žádný vliv.

Pohyb atleta ve videu určuje podle pozice detekované části pozadí ve videu, tudíž při se při pohybu kamerou ve snímku pohybuje pozadí opačným směrem. Tato vlastnost stabilizuje souřadný systém videa při pohybech kamery. Stabilizace je vidět v grafech na obrázku 4.2. V grafech je znázorněna pozice levého a pravého kotníku ve videích 17.MOV a 18.MOV. Kamera se při pořizování videa 17.MOV pohybovala v průběhu první poloviny rozběhu směrem vzhůru, v průběhu druhé poloviny rozběhu směrem dolů. Záběr tedy tvořil oblouk, jelikož zároveň trasoval atleta. Při pořizování druhého videa k vertikálním pohybům kamery nedocházelo.



Obrázek 4.2: Vliv horizontálního pohybu kamery na souřadný systém. Grafy znázorňují pozici levého a pravého kotníku ve videích 17.MOV a 18.MOV.

Pokud bych souřadný systém nestabilizoval trasováním pozadí, vytvořil by graf pozic částí atletova těla při analýze videa 17.MOV oblouk opačným směrem, než kterým se pohybovala kamera. Takovýto oblouk se v grafu nenachází a výsledné parametry nejsou nijak ovlivněny.

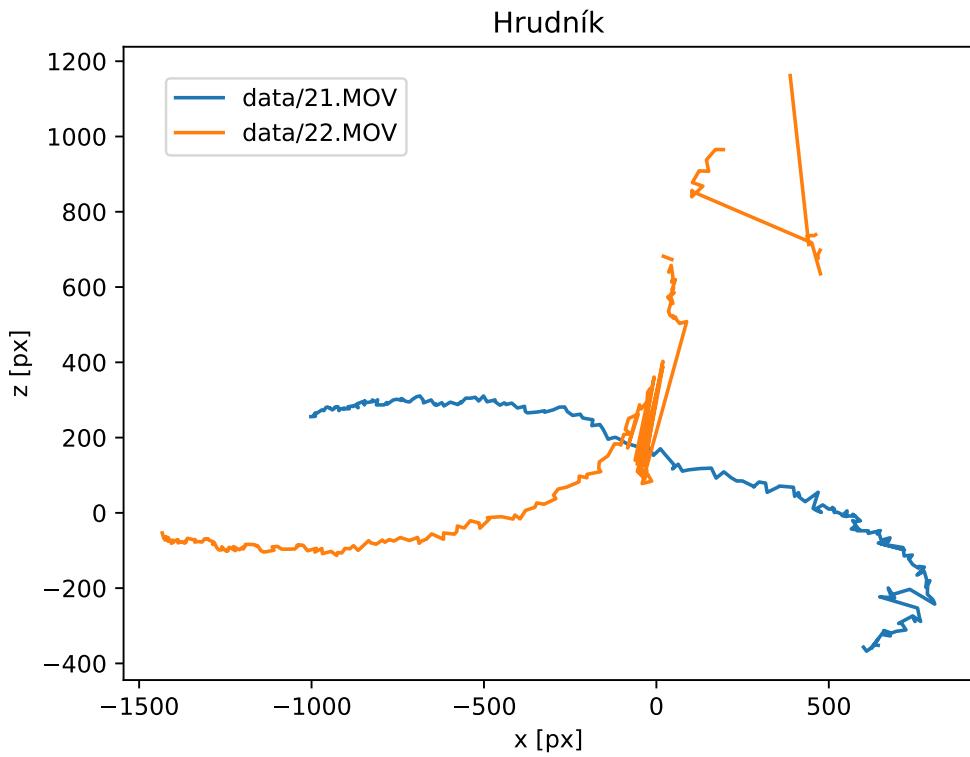
### 4.4.2 Rotace kamery

Zajímavým experimentem je analýza videí, při jejichž natáčení jsem rotoval s kamerou o  $90^\circ$  po (21.MOV) a proti (22.MOV) směru hodinových ručiček. Videa jsou pořízena z levé strany rozběžiště.

Na trasování atleta ani detekci kostry rotace vliv nemá. Je to dáno také charakteristikou vlastnosti skoku o tyči a to rotací atletova těla. S tou je nutné při analýze videozáznamu skoku o tyči počítat.

Rotace kamery znamená také rotaci souřadného systému videa, na kterou není připraven. Z pohybu horizontálního se tedy postupně stává pohyb vertikální. Tuto rotaci souřadného systému zobrazuje graf na obrázku 4.3. Detekce částí těla proběhla správně na téměř všech snímcích videa. Problematická byla jen při dopadu do doskočiště, kdy byl atlet schovaný za stojany.

Jelikož se v obou videích trasovala část pozadí, která byla před atletem (na snímcích videa se tedy nacházela vlevo), měnila se s rotací také pozice atleta



Obrázek 4.3: Vliv rotace na pozici modelu v prostoru.  
Graf znázorňuje pozici hrudníku při analýze videí 21.MOV a 22.MOV.

ve vertikálním směru. Při rotaci po směru hodinových ručiček (video 21.MOV) posouvala rotace atleta pod trasovanou část pozadí, což mělo za důsledek klesání atleta v souřadném systému. Odraz se projevil horizontálním pohybem v původním směru rozběhu. Při opačné rotaci (video 22.MOV) se atlet začal postupně pohybovat nad trasované pozadí, tedy začal stoupat a odraz se projevil horizontálním pohybem proti původnímu směru rozběhu. Míra vertikálního posunu atleta při rotaci je dána vzájemnou vzdáleností atletova těla a trasované části pozadí.

Parametry získané analýzou těchto videí tedy nemají vysokou vypovídající hodnotu.

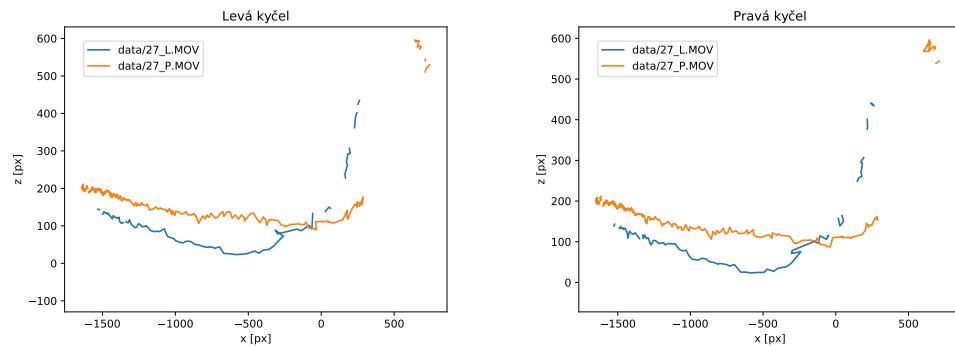
## 4.5 Pozice kamery

### 4.5.1 Směr rozběhu

Právě kvůli porovnání skoku z různých stran jsem pořídil videa 27\_L.mp4 a 27\_P.MOV. Jedná se o tentýž skok, který je natočený ze stejné vzdálenosti od rozběžiště a velice podobné vzdálenosti od zadní hrany zasouvací skřínky. Videa mají rozdílnou frekvenci snímků, ale stejně rozlišení, tudíž je atlet na obou videích v odpovídajících fázích skoku stejně veliký.

Do grafů na obrázku 4.4 jsem zakreslil pozice levé a pravé kyčle. Pro lepší porovnání jsem posunul souřadné systémy podle přesného momentu odrazu. Ten nastává pro video 27\_L.mp4 na snímku číslo 101 a ve videu 27\_P.MOV na snímku číslo 284. Program určil odraz ve snímcích číslo 109 a 1 (počítáno od 0). Chybná

detekce místa odrazu byla způsobena nepřesnou implementací filtrování kroků, vylepšení chybovosti navrhoji v sekci ??.



Obrázek 4.4: Vliv horizontálního pohybu kamery na souřadný systém. Grafy znázorňují pozici levé a pravé kyčle ve videích 27\_L.mp4 a 27\_P.MOV.

Z grafů je vidět, že jsou pozice vyznačených částí těla velice podobné.

Ve fázi rozběhu jednotlivých kroků se kyčle pohybovaly pod jiným úhlem. Vliv na tento úhel má například mírná rotace kamery, jelikož nebyly kamery na stativu, nelze vliv rotace ignorovat.

Z grafů je patrné, že detekce rozběhu proběhla lépe na videu 27\_P.MOV, ale tělo ve fázi skoku se podařilo lépe detektovat na videu 27\_L.mp4.

Tělo se při rozběhu na videu natočeném z pravé strany podařilo detektovat lépe díky lepšímu kontrastu s pozadím, vliv kontrastu probírám více v sekci 4.7. Při fázi skoku byla pro detekci výhodnější levá strana, jelikož je z této strany lépe vidět levá paže (která je z pravé strany schovaná za tělem) a při obratu, který atlet na tyči provádí, je atlet natočen čelem k levé kameře.

#### 4.5.2 Úhel záběru

[Porovnání skoku natočeného z jiných úhlů]

### 4.6 Další postavy ve videu

Další postavy ve videu mají na detekci vliv, zvlášť když se nachází blízko atleta. Tato problematika je vidět na obrázku 2.6.

Výsledné hodnoty parametrů detekce špatné kostry značně znehodnocuje.

### 4.7 Splynutí s pozadím

Pro trasování atleta a detekci kostry je vhodné, aby byl atlet od pozadí dobře rozpoznatelný. Pokud atlet s pozadím splývá, je pro tracker i detektor částí těla složité správně určit jeho pozici. Tento problém je výborně vidět na obrázku ??.

V případě trackeru se jeho ohraňující rámeček v některých případech chytí na pozadí videa a tedy ztratí atleta.



Snímky znázorňují ztrátu atleta při jeho splynutí s pozadím (vlevo) a úspěšné trasování, pokud je atlet od pozadí dobře rozeznatelný (videa 36.mp4 a 40.mp4).

Detekce kostry má větší problém s určením správné pozice kloubů těla, pokud je pozadí blízko atleta členité. Na obrázku ?? [doplním] je vidět několik příkladů tohoto chování detektora.

## 4.8 Frekvence snímků

Frekvence snímků videa (pokud se pohybuje v rozumných hodnotách) má na výsledek detekce zanedbatelný vliv. Značný vliv může mít na přesnost výpočtu hodnot biomechanických parametrů skoku, jelikož na ní do jisté míry závisí.

Z hlediska časové náročnosti analýzy videa je zajímavé sledovat vliv frekvence snímků videa na přesnost detekcí. Vynecháním poloviny snímků videa trvá analýza přibližně polovinu času, což může být výhodné pro analýzu videí, která obsahují spoustu snímků - ať už z důvodu vysoké frekvence či délky videa.

Jelikož jsou mezi testovacími daty i videa s frekvencí snímků 30 fps, nepředpokládal jsem, že trasování selže.

Problematičtější mohla být detekce kostry. S některými fázemi skoku má detektor značné problémy, ty popisuji v sekci 4.9. Některé snímky se detektoru podaří detektovat, ale úspěšnost není vysoká. Může se tedy stát, že by korektní detekce proběhly ve vynechaných snímcích. Detekce kostry spoléhá na natočení okna, které určuje detekce předchozí kostry, což může být z hlediska vynechání snímků, v nichž by k detekci došlo, velice problematické. Okno pro detekci se v takovém případě neotočí, a tedy nedetekuje postavu atleta správně.

Video, které jsem zvolil pro porovnání, je 23.MOV. Toto video je pořízeno frekvencí 60 fps, analýzu jsem spustil bez úprav a poté jsem z videa vynechal každý druhý snímek. Nižší frekvenci jsem netestoval, jelikož 30 snímků za vteřinu je na spodní hranici přípustné frekvence pro analýzu videozáznamu skoku o tyči, jelikož jsou pohyby při skoku velice rychlé.

Pozici atleta ve videu jsem vybral ručně v prvním snímkpu, který byl zahrnutý v obou videích, tedy inicializace atleta proběhla v obou videích stejně.

Jak je vidět v grafu naklonění trupu získaného analýzou videa 30.mp4 4.5, tento problém není příliš podstatný a vliv na detekci má spíše póza, kterou atletovo tělo zaujímá, než jeho samotná rotace.

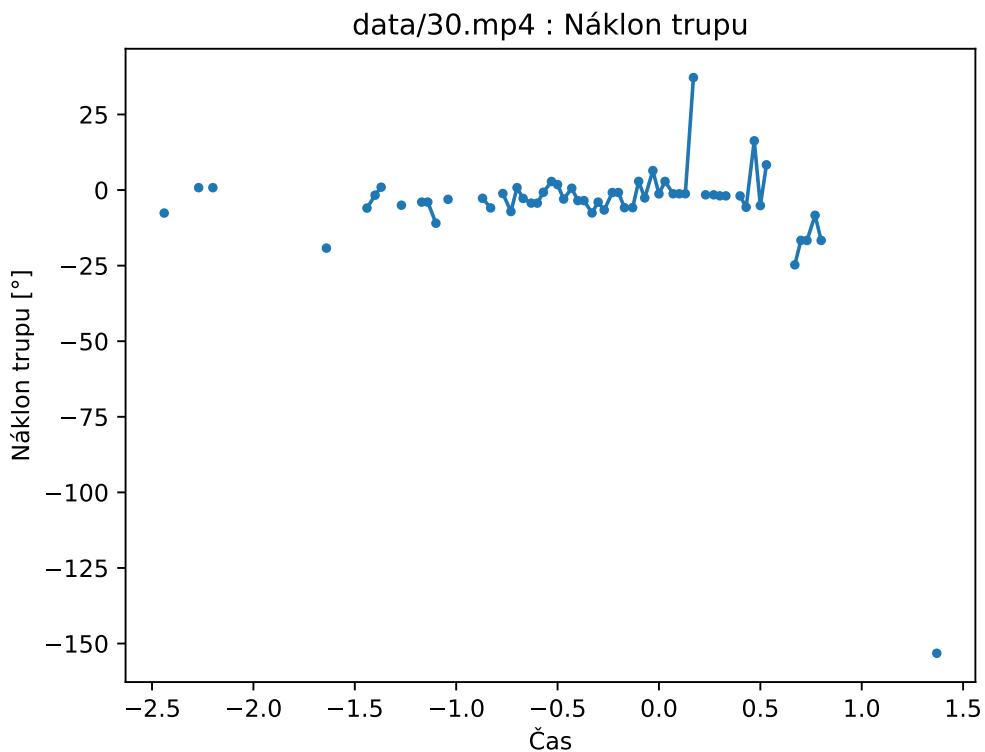
Výsledky experimentu s vynecháním snímků jsou zobrazené na obrázku 4.6. Na grafech je vidět, že detekce selhávala ve velice podobných místech a frekvence videa na výslednou detekci značný vliv neměla. Také nepřesnosti detekcí, které jsou vidět na grafu pohybu pravého zápěstí, jsou velice podobné.

Jelikož se jednalo o stejné video se stejnou úvodní pozicí atleta, rozhodl jsem se počátek souřadného systému pro porovnání pozic jednotlivých částí těla nepřesouvat do pozice odrazového kotníku v momentu odrazu, ale nechal jsem jej ve středu úvodního ohraňujícího rámečku. Posun by mohl být různý, pokud by se detekce lišily a výsledné grafy by nemusely být tak přehledné.

[PARAMETRY]

## 4.9 Shrnutí

Shrnutí analýzy testovacích videí.



Obrázek 4.5: Náklon trupu.

Při analýze videa 30.mp4 dokázal program správně detektovat atleta, jehož trup byl oproti vzpřímené pozici natočen o  $-150^\circ$ . Program v té chvíli předpokládal poslední známý náklon trupu, tedy přibližně  $-20^\circ$  (záporná hodnota značí záklon).

#### 4.9.1 Navrhovaná vylepšení

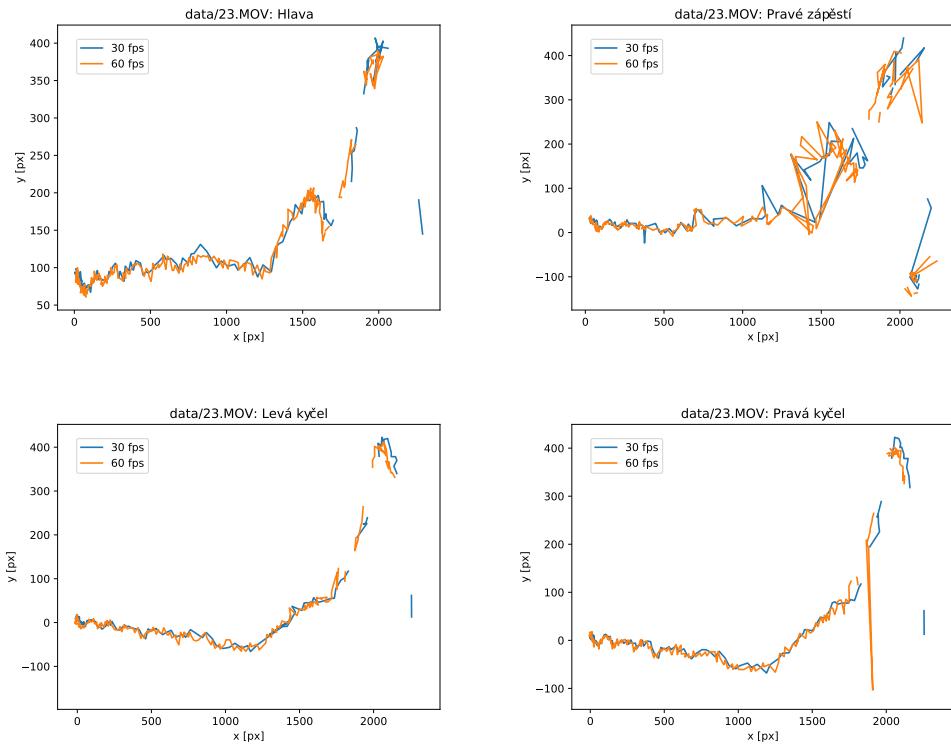
Na závěr bych rád popsal, jakými způsoby by bylo možné funkcionality programu vylepšit.

##### Trasování těla

Úspěšnosti trasování těla atleta při skoku by mohlo pomoci její natáčení podle fáze, ve které se skokan nachází. Přesnějším způsobem trasování může být kombinace implementovaného přístupu s využitím mřížky a metody otáčení snímku, kterou program používal předtím. Mřížku by bylo možné otáčet podle úhlu naklonění trupu.

##### Detekce kostry

Ideálním přístupem k detekci pozice těla pro takto specifické využití je natrénování detektoru na vhodné množině dat. K detekci postavy pouze při skoku o tyči by většinu takové množiny dat měly tvořit právě snímky zachycující skok o tyči. Databáze MPII Human Pose dataset (Andriluka a kol., 2014), na které je natrénovaný model, který aplikace nyní využívá obsahuje naprosté minimum dat týkajících se skoku o tyči, aby dobře generalizoval. Pro účely mé aplikace není vysoká míra generalizace nutná, při detekci těla atleta, když se ve snímku poblíž



Obrázek 4.6: Vliv frekvence snímků na detekci.

Na grafech jsou znázorněny pozice hlavy, pravého zápěstí, levé a pravé kyčle ve videu 23.MOV bez úpravy frekvence snímků a s poloviční frekvencí snímků (každý druhý snímek je vynechaný).

nachází jiná postava přesnosti spíše škodí.

Použití vhodné trénovací databáze by mohlo omezit potřebu určení pozice atleta ve videu pomocí trackerů.

Kromě trénování specializovaného modelu by mohlo být užitečné zkoumat celý výstup sítě, která detekci zajišťuje. Tento způsob by mohl omezit případy, kdy se detekuje více postav najednou (viz sekce 4.6), ale nepomohla by s neúspěšnými detekcemi ve fázi skoku.

Při popisu experimentu v sekci 4.3.2 jsem popsal problém s volbou příliš malého úvodního rámečku atletova těla. Menší rámeček občas zvolí automatické hledání atleta, což může mít za důsledek neúplnou detekci kostry. Tento problém by se dal snadno vyřešit aktualizací velikosti detekčního okna bez ohledu na úspěšnost detekce. Při implementaci jsem dal přednost ošetření případu, který nastává velmi vzácně, na rozdíl od volby příliš malého ohraničujícího rámečku.

Detekce kostry dosahuje horších výsledků z důvodu zmenšení rozlišení okna použitého pro její detekci. Toto zmenšení má obrovský vliv na rychlosť zpracování videa, tudíž jsem k němu přistoupil. Ponechání původního rozlišení by tedy mělo přesnost detekce mírně zlepšit.

## Časová náročnost

Rozdělení videa do několika částí a jejich následné paralelní zpracování by časovou náročnost analýzy značně snížilo. Dalším přístupem by mohlo být rozdelení

detekce kostry do více částí a jejich paralelní zpracování.

Kromě využití paralelismu by mohl pomoci specializovaný model pro detekci částí těla, jelikož nyní při neúspěšné detekci zkouší další natočení snímku, která v určitých případech nepomohou. Tento model by žádné další natočení nepotřeboval, a tudíž by se detekce spouštěla na menším množství vstupů.

Časovou náročnost při detekci postav zbytečně zvyšuje množství instancí třídy `background_tracker`, díky nimž určuji pohyb postav ve videu. Tyto trackery často trasují stejnou část pozadí, ale aktualizují se pro každou instanci. Původně jsem tracker pozadí implementoval v závislosti na poloze příslušné postavy a objektový návrh jsem poté nestihl změnit.

## Analýza parametrů

Přesnost analýzy parametrů, ale také zanesení modelu atleta do souřadného systému, ovlivňuje rotace kamery. Stabilizaci této rotace by zajistila dvojice trackerů pozadí, jejichž vzdájemná pozice by dokázala určit míru rotace jednotlivých snímků. Díky této hodnotě bych snadno určil přesnou pozici atleta v prostoru i při rotaci kamery.

# Závěr

Jednotlivé kroky analýzy videozáznamu skoku o tyči, mezi které patří extrakce modelu atleta, detekce důležitých momentů skoku, vyhodnocení a zobrazení parametrů se mi podařilo implementovat. Ačkoliv není přesnost analýzy vždy dostatečná, jedná se o úvodní krok k realizaci aplikace, která by videozáznamy dokázala spolehlivě a přesně analyzovat a na základě výsledků těchto analýz skoky porovnávat, což byl můj prvotní cíl.

Původně jsem měl v úmyslu vytvořit aplikaci pro mobilní zařízení, která by videa analyzovala v průběhu tréninků a soutěží. Převod aplikace z počítače na platformu iOS byl složitější, než jsem očekával a raději jsem svou pozornost zaměřil na vylepšení funkcionality počítačové aplikace. Posléze se ukázalo, že časová náročnost analýzy by bez modifikací implementovaného algoritmu nezvládla videa analyzovat včas.

Značnou pozornost jsem při psaní práce věnoval automatickému nalezení atleta ve videu. Parametry pro vyhledávání postav ve videu se mi bohužel nepodařilo nastavit tak, abych byl s úspěšností vyhledávání spokojen. Proto je manuální nalezení atleta ve videu výchozím nastavením programu.

Způsob trasování atleta jsem v průběhu psaní práce několikrát měnil, úspěšnost se stále zlepšovala, ale ani aktuální metoda není 100% úspěšná ve fázi skoku.

Detekce částí atleta má problémy s malými postavami a při skoku, především ve fázi skoku, kdy má atlet nohy blízko trupu. Tyto nepřesnosti jsou důsledkem zvoleného modelu, který jednotlivé části těla detekuje a jeho trénovací databází. Ani pokusy o natočení snímku detekci příliš nepomáhají.

Na přesnostech detekce atletova těla závisí hodnoty zkoumaných parametrů (a také důležitých momentů skoku). Například detekce momentu odrazu spoléhá na kvalitní detekci atletových nohou, která ne vždy proběhne. Kvalitní analýza parametrů si zakládá na důkladné znalosti biomechanických principů a její implementace vyžaduje mezioborovou spolupráci.

Věřím, že jsem svou prací ukázal, že je možné vytvořit program schopný kvalitní analýzy parametrů skoku o tyči, kterou by bylo možné využívat jak v tréninkovém, tak závodním prostředí.

# Seznam použité literatury

- AGGARWAL, J. a CAI, Q. (1997). Human motion analysis: a review. In *Proceedings IEEE Nonrigid and Articulated Motion Workshop*, pages 90–102. doi: 10.1109/NAMW.1997.609859.
- ANDRILUKA, M., PISHCHULIN, L., GEHLER, P. a SCHIELE, B. (2014). 2D Human Pose Estimation: New Benchmark and State of the Art Analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- ATHLETICS, W. (2021). Research centre. [online]. URL <https://www.worldathletics.org/about-iaaf/documents/research-centre>. Naposledy navštívěno 17. 5. 2021.
- BRADSKI, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.
- CAO, Z., HIDALGO, G., SIMON, T., WEI, S.-E. a SHEIKH, Y. (2021). OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **43**(1), 172–186. doi: 10.1109/TPAMI.2019.2929257.
- CHANG, I.-C. a HUANG, C.-L. (2000). The model-based human body motion analysis system. *Image and Vision Computing*, **18**(14), 1067–1083. ISSN 0262-8856. doi: [https://doi.org/10.1016/S0262-8856\(00\)00046-9](https://doi.org/10.1016/S0262-8856(00)00046-9). URL <https://www.sciencedirect.com/science/article/pii/S0262885600000469>.
- CORTES, C. a VAPNIK, V. (1995). Support-vector networks. *Machine learning*, **20**(3), 273–297.
- DALAL, N. a TRIGGS, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1. doi: 10.1109/CVPR.2005.177.
- FANG, H.-S., XIE, S., TAI, Y.-W. a LU, C. (2018). RMPE: Regional Multi-person Pose Estimation.
- FIAZ, M., MAHMOOD, A. a JUNG, S. K. (2018). Tracking Noisy Targets: A Review of Recent Object Tracking Approaches.
- FOUNDATION, P. S. (2021). 1. Embedding Python in Another Application. [online]. URL <https://docs.python.org/3.8/extending/embedding.html>. Naposledy navštívěno 23. 5. 2021.
- GRAVESTOCK, H., BISSAS, A. a MERLINO, S. (2017). Biomechanical report for the IAAF World Championships London 2017: Pole Vault Men's. URL <https://www.worldathletics.org/download/download?filename=e2903e25-0ee3-43d2-b031-1af5a3b84fb8.pdf&urlslug=Men%27s%20pole%20vault%20-%202017%20IAAF%20World%20Championships%20Biomechanical%20report>.

HUNTER, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, **9**(3), 90–95. doi: 10.1109/MCSE.2007.55.

JIA, Y., SHELHAMER, E., DONAHUE, J., KARAYEV, S., LONG, J., GIRSHICK, R., GUADARRAMA, S. a DARRELL, T. (2014). Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*.

KRISTAN, M., LEONARDIS, A., MATAS, J., FELSBERG, M., PFLUGFELDER, R., ZAJC, L. C., VOJÍŘ, T., HÄGER, G., LUKEŽIC, A., ELDESOKEY, A., FERNÁNDEZ, G., GARCIA-MARTIN, A., MUHIC, A., PETROSINO, A., MEMARMOGHADAM, A., VEDALDI, A., MANZANERA, A., TRAN, A., ALATAN, A., MOCANU, B., CHEN, B., HUANG, C., XU, C., SUN, C., DU, D., ZHANG, D., DU, D., MISHRA, D., GUNDOGDU, E., VELASCO-SALIDO, E., KHAN, F. S., BATTISTONE, F., SUBRAHMANYAM, G. R. K. S., BHAT, G., HUANG, G., BASTOS, G., SEETHARAMAN, G., ZHANG, H., LI, H., LU, H., DRUMMOND, I., VALMADRE, J., JEONG, J.-C., CHO, J.-I., LEE, J.-Y., NOSKOVA, J., ZHU, J., GAO, J., LIU, J., KIM, J.-W., HENRIQUES, J. F., MARTÍNEZ, J. M., ZHUANG, J., XING, J., GAO, J., CHEN, K., PALANIAPPAN, K., LEBEDA, K., GAO, K., KITANI, K. M., ZHANG, L., WANG, L., YANG, L., WEN, L., BERTINETTO, L., POOSTCHI, M., DANELLJAN, M., MUELLER, M., ZHANG, M., YANG, M.-H., XIE, N., WANG, N., MIKSIK, O., MOALLEM, P., VENUGOPAL, P. M., SENNA, P., TORR, P. H. S., WANG, Q., YU, Q., HUANG, Q., MARTÍN-NIETO, R., BOWDEN, R., LIU, R., TAPU, R., HADFIELD, S., LYU, S., GOLODETZ, S., CHOI, S., ZHANG, T., ZAHARIA, T., SANTOPIETRO, V., ZOU, W., HU, W., TAO, W., LI, W., ZHOU, W., YU, X., BIAN, X., LI, Y., XING, Y., FAN, Y., ZHU, Z., ZHANG, Z. a HE, Z. (2017). The Visual Object Tracking VOT2017 Challenge Results. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 1949–1972. doi: 10.1109/ICCVW.2017.230.

LINDEBERG, T. (2012). *Scale Invariant Feature Transform*, volume 7. doi: 10.4249/scholarpedia.10491.

LUKEŽIČ, A., VOJÍŘ, T., ČEHOVIN ZAJC, L., MATAS, J. a KRISTAN, M. (2018). Discriminative Correlation Filter Tracker with Channel and Spatial Reliability. *International Journal of Computer Vision*, **126**(7), 671–688. ISSN 1573-1405. doi: 10.1007/s11263-017-1061-3. URL <http://dx.doi.org/10.1007/s11263-017-1061-3>.

MATLAB (2010). *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts.

PISHCHULIN, L., INSAFUTDINOV, E., TANG, S., ANDRES, B., ANDRILUKA, M., GEHLER, P. a SCHIELE, B. (2016). DeepCut: Joint Subset Partition and Labeling for Multi Person Pose Estimation.

ROSEBROCK, A. (2015). (Faster) Non-Maximum Suppression in Python. [online]. URL <https://www.pyimagesearch.com/2015/02/16/faster-non-maximum-suppression-python/>. Naposledy navštíveno 21. 5. 2021.

SIMI REALITY MOTION SYSTEMS GMBH (2014). Simi Motion 2D/3D - Motion Capture and complex, biomechanical Movement Analysis - syncable, video-based system. [online]. URL <http://www.simi.com/en/products/movement-analysis/simi-motion-2d3d.html?type=rss%C3%83%C2%82%C3%82%C2%B4a%3D0%27a%3D0>. Naposledy navštíveno 24. 5. 2021.

SOLEIMANITALEB, Z., KEYVANRAD, M. A. a JAFARI, A. (2019). Object Tracking Methods:A Review. In *2019 9th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 282–288. doi: 10.1109/ICCKE48569.2019.8964761.

SUN, K., XIAO, B., LIU, D. a WANG, J. (2019). Deep High-Resolution Representation Learning for Human Pose Estimation.

SZANDAŁA, T. (2020). Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks.

TATARIANTS, M. (2021). Human Pose Estimation Technology 2021 Guide. [online]. URL <https://mobidev.biz/blog/human-pose-estimation-ai-personal-fitness-coach>. Naposledy navštíveno 22. 5. 2021.

TOSHEV, A. a SZEGEDY, C. (2014). DeepPose: Human Pose Estimation via Deep Neural Networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. doi: 10.1109/cvpr.2014.214. URL <http://dx.doi.org/10.1109/CVPR.2014.214>.

WILLIAMS, T., KELLEY, C. a MANY OTHERS (2013). Gnuplot 4.6: an interactive plotting program. [online]. URL <http://gnuplot.sourceforge.net/>.

# Seznam obrázků

1.1	Jednoduchá a hluboká neuronová síť. Jednotlivá kolečka reprezentují neurony, úsečky mezi nimi přechody mezi neurony (váhy). Převzato z <a href="https://becominghuman.ai/deep-learning-made-easy-with-deep-cognition-403fbe445351">https://becominghuman.ai/deep-learning-made-easy-with-deep-cognition-403fbe445351</a> dne 26. 5. 2021.	11
1.2	Pravděpodobnostní mapy detekce různých kloubů těla.	12
2.1	Vizualizace souřadného systému videa. Světlé pole reprezentuje snímek videa, $p$ posun jeho horního rohu vůči původní pozici atleta, $q$ bod, jehož pozici ve videu získám pomocí posunu snímku a pozice bodu $q$ ve snímku.	14
2.2	Vizualizace souřadného systému 3D prostoru. Červená reprezentuje osu $x$ , zelená $y$ a modrá $z$ . Směr šipek odpovídá růstu hodnot bodů v prostoru.	14
2.3	Ohraničující rámeček atletova těla ve snímku videa 1.MOV (viz tabulka 4.1).	16
2.4	Použití jednoho trackeru (vlevo) a mřížky trackerů, která je implementovaná v programu (vpravo). Použité snímky jsou vybrané z videa 33.MOV (viz tabulka 4.1).	18
2.5	Pohyb mřížky v průběhu její první aktualizace. (a) - úvodní pozice mřížky, (b) - aktualizace stále neproběhla, protože není znám směr rozbehru, (c) - poslední snímek před aktualizací mřížky, (d) - první snímek po aktualizaci mřížky	19
2.6	Detekce špatné kostry a následné napravení. (a) - správná detekce, (b) - špatná detekce, (c) - průběh nápravy, (d) - správná detekce	19
2.7	Způsob zobrazení výstupu uživateli. (a) - snímky zachycující začátek rozbehru, odraz a kulminaci boků, prohlížeč videa a výpis příslušných hodnot analyzovaných parametrů, (b) - grafy znázorňující hodnoty parametrů v průběhu skoku. Pro zobrazení tohoto výstupu byl použit model získaný z videa 1.MOV (viz tabulka 4.1).	25
3.1	UML diagram zobrazující strukturu programu.	30
3.2	Výsledek aplikace algoritmu non-maximum suppression při detekování tváře. Implementace podle Rosebrock (2015) zvolila rámeček, který se na snímku nachází nejníž. Má implementace by dala přednost největšímu snímku. Převzato z <a href="https://www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/">https://www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/</a> dne 26. 5. 2021.	32
4.1	Testovací videa.	44
4.2	Vliv horizontálního pohybu kamery na souřadný systém.	46
4.3	Vliv rotace na pozici modelu v prostoru.	47
4.4	Vliv horizontálního pohybu kamery na souřadný systém.	48
4.5	Náklon trupu.	51
4.6	Vliv frekvence snímků na detekci.	52

# **A. Přílohy**

## **A.1 První příloha**