SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2398

PREPOZNAVANJE POZE ČOVJEKA I STVARANJE 3D MODELA IZ VIDEOZAPISA

Marin Hrkec

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2398

PREPOZNAVANJE POZE ČOVJEKA I STVARANJE 3D MODELA IZ VIDEOZAPISA

Marin Hrkec

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

DIPLOMSKI ZADATAK br. 2398

Pristupnik:	**Marin Hrkec (0036498864)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Željka Mihajlović

Zadatak: **Prepoznavanje poze čovjeka i stvaranje 3D modela iz videozapisa**

Opis zadatka:

Proučiti sustav animacije čovjeka koji prikazuje model čovjeka s raznim animacijama pokreta čovjeka. Proučiti metode prepoznavanja poze čovjeka iz videozapisa, posebice proučiti sustav VIBE (Video Inference for Body Pose and Shape Estimation). Razraditi mogućnosti animiranih pokreta modela ljudskog tijela. Razraditi mogućnosti stvaranja 3D modela iz videozapisa. Analizirati i ocijeniti ostvarene rezultate. Diskutirati upotrebljivost ostvarenih rezultata kao i moguća proširenja. Izraditi odgovarajući programski proizvod. Koristiti programski jezik Python, programsku biblioteku PyTorch, te skup podataka AMASS. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 28. lipnja 2021.

SADRŽAJ

1. Uvod

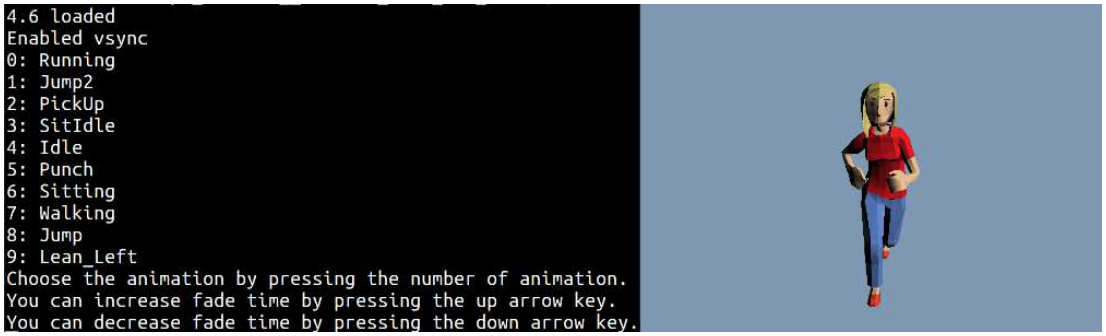
Prepoznavanje poze čovjeka iz slike, a posebice iz videozapisa je dugotrajan problem s mnogim primjenama. Postignut je veliki napredak u prepoznavanju poze i oblika čovjeka iz jedne slike. Ipak, iz videozapisa se saznaje više informacija o čovjeku, a u tom segmentu još nema većeg napretka. Većina metoda ne uspijeva kreirati precizne i prirodne pokrete čovjeka. Središte ovog rada je proučiti sustav animacije čovjeka koji prikazuje model čovjeka s raznim animacijama pokreta čovjeka, kao i metode prepoznavanja poze čovjeka iz videozapisa, posebice sustav VIBE (engl. *Video Inference for Human Body Pose and Shape Estimation*). VIBE je jedan od novih sustava za prepoznavanje poze čovjeka iz videozapisa koji koristi skup podataka `AMASS`. `AMASS` je skup podataka dobiven uzorkovanjem pokreta čovjeka.

2. Sustav animacije čovjeka

Sustav animacije čovjeka izrađen je u *OpenGL*-u koristeći programski jezik C++. Pri tome je pokrivena izrada raznih elemenata potrebnih za razvoj samog sustava. Izrađen je funkcionalni sustav koji prikazuje model čovjeka s raznim animacijama - trčanje, hodanje, sjedenje, skakanje. Ostvarena je podrška za učitavanje modela čovjeka, implementiran je animacijski isječak (engl. *animation clip*), postupci *mesh skinning* i *animation blending*, tj. prijelaz iz jedne animacije u drugu.

Profesionalni sustavi animacije zahtijevaju mnogo vremena i znanja za izradu. Izrada takvih sustava je zahtjevan posao pa je iz tog razloga u sklopu diplomskog rada fokus na izradi jednostavnijeg sustava. Grafički stil je jednostavan, a model čovjeka za prikaz animacija koristi manji broj poligona u odnosu na profesionalne modele. Slični sustavi se izvode unutar studija koji razvijaju grafičke pogone za igre (engl. *game engine*). Primjer takvog sustava je sustav animacije *Mecanim* grafičkog pogona *Unity*.[1] Sustav animacije je jedan od sustava koji čine grafički pogon i omogućuje lakše upravljanje animacijama. Sustav razvijen u sklopu ovog diplomskog rada je opsegom mnogo manja verzija takvih projekata. [7] Na slici 2.1 je prikazana aplikacija tijekom izvođenja.

[1] `https://docs.unity3d.com/460/Documentation/Manual/MecanimAnimationSystem.html`



Slika 2.1: Prikaz aplikacije u nekom trenutku izvođenja

2.1. Korišteni alati

Za izradu ovog sustava koristi se grafička biblioteka *OpenGL* i programski jezik C++. Model čovjeka i animacije se učitavaju iz *glTF* datoteke uz korištenje biblioteke *cgltf*. Za učitavanje teksture koristi se biblioteka *stb_image*. Implementiran je postupak *mesh skinning* koji se izvodi na grafičkoj kartici (engl. *Graphics processing unit, GPU*). Postoji mogućnost odabira animacije i podešavanja vremena prijelaza iz jedne animacije u drugu koristeći tipkovnicu. [7],[9]

2.2. Izrada pomoćnih elemenata

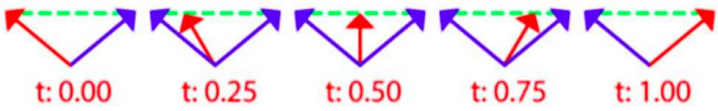
Prikaz animacija je matematički zahtjevan posao. Zato je za prikaz na ekran i animiranje potrebno izraditi pomoćne matematičke elemente – vektore, matrice i kvaternione.

2.2.1. Vektori

Vektori su veličine koje imaju iznos, smjer i orijentaciju. Koriste se za prikaz pomaka. U sklopu sustava animacije podržani su vektori s dvije, tri i četiri dimenzije, kao i matematičke operacije: zbrajanje i oduzimanje vektora, množenje vektora skalarom, skalarni i vektorski umnožak vektora. Također, moguće je izračunati normu vektora, normalizirani vektor, kut između vektora, odbijanje (refleksiju) i interpolaciju vektora. Podržane su tri vrste interpolacije: linearna interpolacija (*lerp*), sferična linearna interpolacija (linearna interpolacija po površini sfere, *slerp*) i normalizirana linearna interpolacija (*nlerp*) prikazane na slici 2.2. Normalizirana linearna je dovoljno točna aproksimacija sferične linearne interpolacije i zato se često koristi.[9]

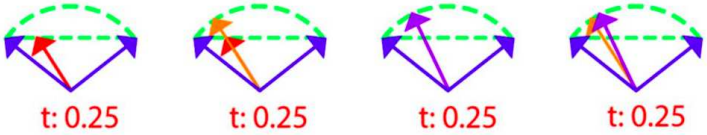
2.2.2. Matrice

U kontekstu animacije, matrica predstavlja afinu transformaciju. Linearno preslikava točke iz jednog prostora u drugi. Afina transformacija ili preslikavanje u geometriji predstavlja funkciju između afinih prostora koja preslikava točke u točke, pravce u pravce i ravnine u ravnine. Također, kod afinih preslikavanja, par paralelnih pravaca ostaje paralelan nakon preslikavanja, ali preslikavanje ne čuva kutove između pravaca ili udaljenosti između točaka. Mreža poligona (engl. *mesh*) je predstavljena vrhovima, koji su samo točke u prostoru. Ti vrhovi se pomiču množenjem matricama. Matrice su također potrebne za transformaciju pogleda i projekcije. U sklopu sustava animacije podržane su matrice dimenzija 2x2, 3x3 i 4x4, kao i matematičke operacije: zbrajanje



(a) LERP prikazana crvenom bojom

(b) SLERP prikazana narančastom bojom



(c) NLERP prikazana ljubičastom bojom

Slika 2.2: Usporedba interpolacija vektora

i oduzimanje matrica, množenje matrice skalarom, množenje matrica, transponiranje matrice, računanje inverza matrice i računanje determinante matrice.

Kreiranje matrica kamere

Matrice se koriste za transformaciju u koordinatni sustav kamere. Prva potrebna matrica kamere je *frustum*. *Frustum* izgleda kao krnja četverostrana piramida – ima šest strana i predstavlja sve što je vidljivo kameri. Izrada *frustuma* je neintuitivna pa se zato još dodaje pomoćna matrica perspektive (engl. *perspective matrix*) za prikladniju izradu *frustuma*. Matrica perspektive se koristi za većinu grafičkih prikaza na ekran. Uz nju, kreirana je i matrica ortografske projekcije. Ona linearno preslikava vrhove u prostor kamere i često se koristi u dvodimenzionalnim igrama. Matrica pogleda (engl. *view matrix*) je inverz matrice transformacije kamere (pozicije, rotacije i skale kamere). Često se koristi i umjesto da se kreira matrica kamere pa se računa njen inverz, implementirana je funkcija `lookAt` za generiranje te matrice. Kao parametri toj funkciji šalju se: pozicija kamere, točka u koju kamera gleda i vektor pogled-gore (engl. *view-up*) koji određuje okomicu na pogled kamere.

2.2.3. Kvaternioni

Kvaternioni su proširenje kompleksnih brojeva. Općeniti zapis kvaterniona je

$$a_1 + a_2 i + a_3 j + a_4 k = (a_1 \vec{a}).$$

Služe za zapis rotacije i dobro interpoliraju rotaciju. Vrlo su bitan element sustava animacije jer se mnogo animacija čovjeka može postići samo rotacijom, bez potrebe translacije i skaliranja. Primjer takvih animacija su animacije zglobova poput koljena ili lakta koji mogu samo rotirati. Kvaternioni se često stvaraju koristeći os i kut rotacije. Rotacija oko neke osi za kut θ može se prikazati na sferi kao usmjereni kružni luk čija je duljina $\frac{1}{2}\theta$ na ravnini koja je okomita na os rotacije. Kvaternion može provesti dvije pune rotacije, tj. rotaciju za 720 stupnjeva. To znači da je period kvaterniona 720 stupnjeva. Period funkcija sin i cos je 360 stupnjeva. Zato se dijeljenjem θ s 2 opseg rotacije kvaterniona izjednačuje s opsegom funkcija sin i cos. Tako se rotacija neke točke oko vektora $\vec{u}$ za kut θ može zapisati kao kvaternion q:

$$q = (s\vec{v}), s = cos\frac{\theta}{2}, \vec{v} = \hat{u}sin\frac{\theta}{2}$$

U sklopu sustava animacije podržane su matematičke operacije za rad s kvaternionima: zbrajanje i oduzimanje kvaterniona, množenje kvaterniona skalarom, množenje kvaterniona, računanje modula i jediničnog kvaterniona, kao i interpolacija kvaterniona. Interpoliraju se slično kao i vektori. Interpolacija kvaterniona se koristi za animiranje rotacija između dva ključna okvira, a budući da se mnogo animacija čovjeka ostvaruje rotacijom zglobova, interpolacija se često koristi. Kod interpolacije je potrebno očuvati jediničnu duljinu kvaterniona. Linearnom interpolacijom se ne ostvaruje konstantna brzina pa se linearna interpolacija ne koristi.[9]

2.2.4. Struktura `Transform`

Pozicija, rotacija i skaliranje nekog tijela mogu biti zapisani u jednoj matrici dimenzija 4x4. No interpolacija između matrica može biti zahtjevna jer su rotacija i skaliranje zapisani u iste komponente takve matrice. Zato se uvodi nova struktura `Transform` koja rješava taj problem tako da se pozicija, rotacija i skala spremaju zasebno. U isječku koda 2.1 je prikazana definicija te strukture.

```
1 struct Transform {
2   vec3 position;
3   quat rotation;
4   vec3 scale;
5 }
```

Isječak koda 2.1: Definicija strukture `Transform`

Strukture `Transform` se mogu kombinirati kao što se mogu matrice i kvaternioni. Efekti dviju struktura `Transform` se mogu kombinirati u jednu. Kombiniranje skaliranja i rotacije dvije struktura je jednostavno – potrebno ih je samo pomnožiti. Kombiniranje pozicije je ipak zahtjevnije. Na novu poziciju utječu i komponente skaliranja i rotacije. Kod određivanja kombinacija dvije strukture, redoslijed transformacija je: prvo skaliranje, zatim rotiranje i posljednje je translatiranje.

Pomoću te strukture se preslikava iz jednog koordinatnog sustava u drugi. Moguće je i obratno – kreirati strukturu pomoću koje se vraća iz ciljnog prostora u početni. Kao matricama i kvaternionima, i strukturama `Transform` je moguće izračunati inverz. Na primjer, neki lik u igri se kreće kroz prostor i nakon prolaska na sljedeću razinu, potrebno je vratiti tog lika na početno mjesto. To se jednostavno postiže množenjem strukture `Transform` lika s njezinim inverzom.

Postoje strukture `Transform` koje predstavljaju zglobove u dva određena trenutka u vremenu. Za animiranje nekog modela čovjeka, potrebno ih je interpolirati. Vektori i kvaternioni čine strukturu `Transform`. Budući da se vektori i kvaternioni mogu interpolirati, `Transform` se može također. Ta funkcija se zove miješanje (engl. *blend, mix*). Isječak izvornog koda 2.2 prikazuje funkciju miješanja. Ta mogućnost miješanja je bitna za stvaranje glatkih prijelaza između animacija. [9]

```
Transform mix(const Transform& a, const Transform& b, float t) {
  return Transform(
    lerp(a.position, b.position, t),
    nlerp(a.rotation, b.rotation, t),
    lerp(a.scale, b.scale, t)
  );
}
```

Isječak koda 2.2: Miješanje u strukturi `Transform`

2.2.5. Ostvarivanje prikaza

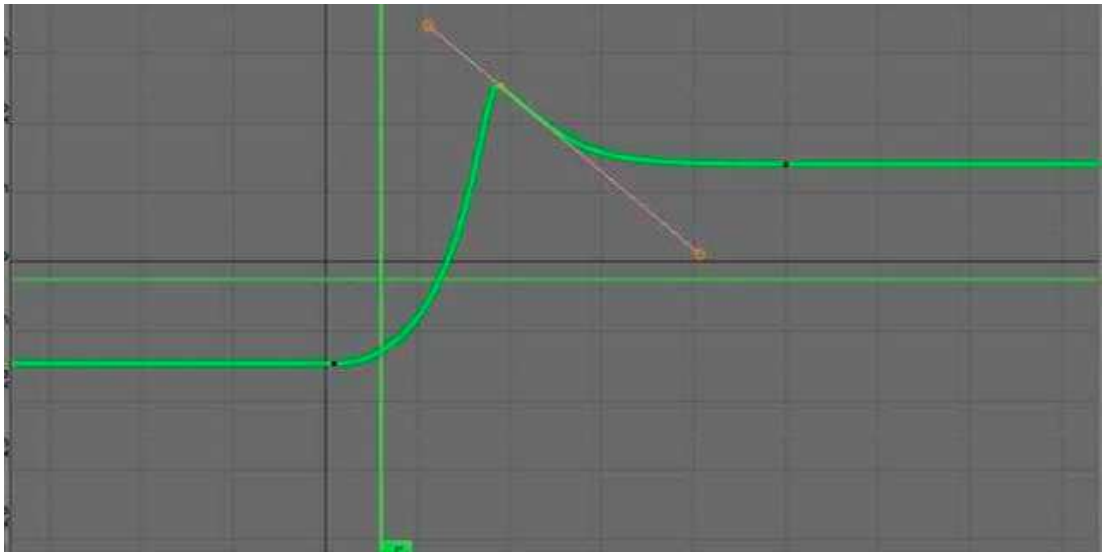
Za olakšavanje rada s *OpenGL*-om, dodaje se sloj apstrakcije *Renderer*, dovoljan za prikaz animiranog modela. Glavni dio ovog sloja je klasa `Shader`. `Shader` opisuje što se iscrtava i kako se taj objekt transformira i sjenča. Klasa sadrži funkcije za prevođenje i povezivanje (engl. *linking*) sjenčara vrhova (engl. *vertex shader*) i sjenčara fragmenata (engl. *fragment shader*). Za rad sa sjenčarima, pretpostavlja se da difuzna komponenta boje dolazi iz teksture. Teksture se učitavaju iz slika `.png` formata. Za učitavanje slike koristi se pomoćna biblioteka `stb_image`.[9]

2.2.6. Datoteke formata `glTF`

Datoteke formata `glTF` se mogu spremati na više načina. Prvi način je u obliku čistoga teksta s ekstenzijom `.gltf`. Drugi način je spremanje u kompaktnijem obliku u binarnoj datoteci s ekstenzijom `.glb`. U sklopu ovog diplomskog rada radi se s datotekama s ekstenzijom `.gltf`. Datoteke formata `glTF` se koriste za spremanje cijele trodimenzionalne scene, a ne pojedinog modela pa dio spremljenih podataka u datoteci nije potreban. Za učitavanje `glTF` datoteke koristi se pomoćna biblioteka `cgltf`.[7]

2.3. Animacijske krivulje i okviri

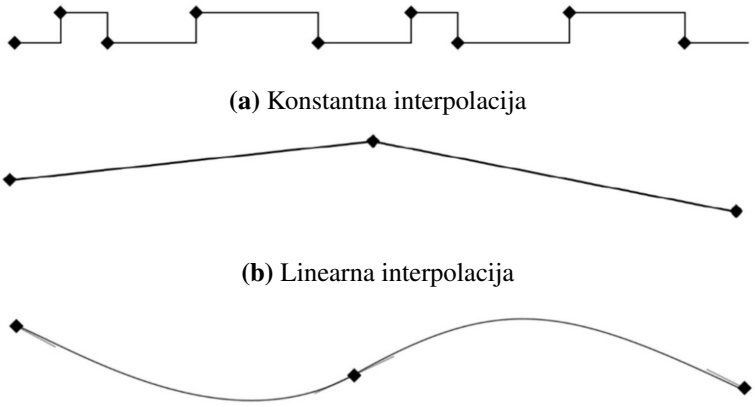
U početku razvoja animacije, bilo je uobičajeno da se za animacije u igrama uzorkuje svaki zglob u animaciji u određenim intervalima pa se za prikaz te animacije provodila linearna interpolacija tih uzoraka. No to nije precizan način za izvođenje animacija. U alatima za izradu 3D modela, poput *Blendera* i *Maye*, animacije se stvaraju korištenjem animacijskih krivulja, poput krivulje koja je prikazana na slici 2.3. Na osi apscisi je vrijeme, a na ordinati vrijednost u tom vremenu. Na primjer, vrijednost na ordinati može označavati vrijednost x koordinate animiranog modela kroz vrijeme pa bi ta krivulja označavala kako se taj model miče po x koordinati u vremenu. Koriste se Hermitove i Bezierove krivulje. Često korištena krivulja u animacijama je kubna Hermitova krivulja.[9]



Slika 2.3: Primjer animacijske krivulje

Kod definiranja animacijske krivulje, koristi se jedna od tri metode interpolacije – konstantna, linearna ili kubna. Sve tri vrste su prikazane na slici 2.4. Kod kons-

tantne interpolacije, krivulja ima konstantnu vrijednost do sljedećeg ključnog okvira (engl. *keyframe*). Linearna interpolacija između dva okvira je interpolacija pravcem. Kubna interpolacija se može prikazati Bezierovim ili Hermitovim krivuljama, a u sklopu ovog diplomskog rada se koristi Hermitova krivulja. Kod Hermitove kubne krivulje, svaki okvir uz svoju vrijednost, ima i vrijednost, tj. nagib ulazne i izlazne tangente iz točke.[9]



(a) Konstantna interpolacija

(b) Linearna interpolacija

(c) Kubna interpolacija

Slika 2.4: Interpolacije krivulja

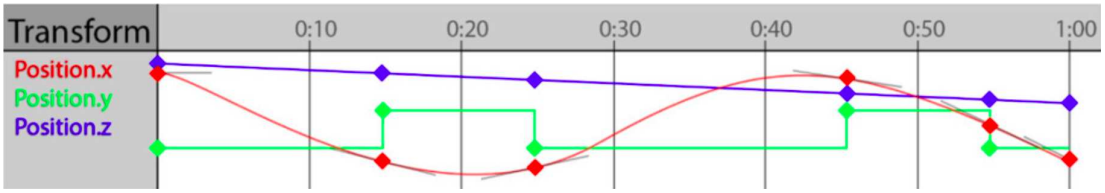
Za spremanje podataka pojedinog okvira, definira se klasa `Frame`. Podaci koji definiraju pojedini okvir ovise o metodi interpolacije koja se koristi. Ako je interpolacija konstantna ili linearna, onda su to samo vrijeme i vrijednost. Ako je interpolacija kubna, onda je potrebno spremati i tangente. Okvir (engl. *frame*) može biti proizvoljne veličine – skalar, vektor ili kvaternion. Definicija klase je prikazana u isječku koda 2.3.[9]

```
template<unsigned int N>
class Frame {
public:
  float mTime;
  float mValue[N];
  float mIn[N];
  float mOut[N]; }
```

Isječak koda 2.3: Klasa `Frame`

U klasu `Frame` se spremaju ključni okviri animacije. Kolekcija okvira se naziva animacijska traka (engl. *animation track*). Animacijska traka mora sadržavati barem dva okvira. Primjer jedne takve trake prikazan je na slici 2.5 – u okvirima su sprem-

ljene vrijednosti o položaju. Koriste se tri različite metode interpolacije za svaku koordinatu – konstantna za y, linearna za z i kubna za x koordinatu.



Slika 2.5: Animacijska traka

Za svaku animiranu strukturu `Transform` nije poželjno spremati odvojene vektore i kvaternione animacijskih traka. Zato se definira nova klasa `TransformTrack` koja enkapsulira tri trake: traku za poziciju, za rotaciju i za skaliranje. Definicija klase je prikazana u isječku koda 2.4. Kostur modela sadrži više kostiju, ali sve animacije ne animiraju sve kosti. Zato je još dodan atribut `mId` koji označava kost na koju se odnosi.[9]

```
class TransformTrack {
  unsigned int mId;
  VectorTrack mPosition;
  QuaternionTrack mRotation;
  VectorTrack mScale; }
```

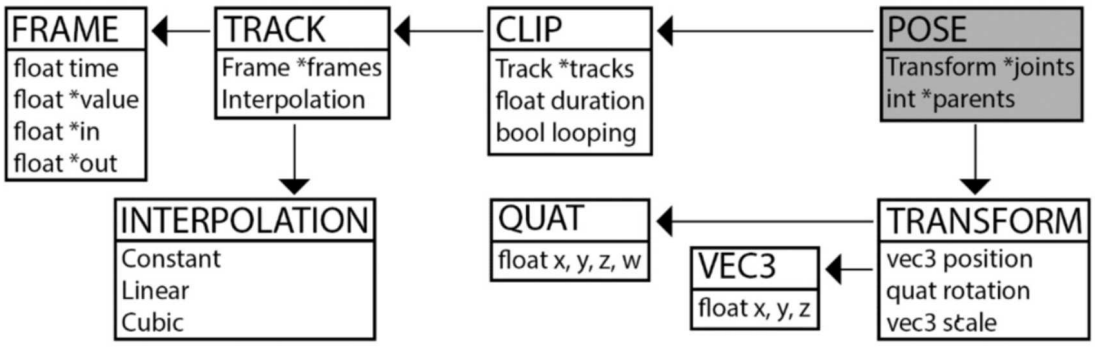
Isječak koda 2.4: Klasa `TransformTrack`

Animacijski isječak (engl. *animation clip*) je kolekcija `TransformTrack` objekata koji se sastoje od barem dva okvira. Isječak animira kolekciju tih objekata, a ta kolekcija koja se animira se naziva poza. Pozu se može zamisliti kao kostur animiranog lika u nekom određenom vremenskom trenutku. Poza je zapravo hijerarhija `Transform` objekata i vrijednosti svakog tog objekta utječu na njegovu djecu u hijerarhiji. Na slici 2.6 su prikazane ovisnosti svih objekata koji su potrebni za generiranje poze animiranog lika u jednoj sličici (engl. *frame*). Rezultat uzorkovanja animacijskog isječka je poza.

Definira se nova klasa za zapis poza. Poza je transformacija svih kostiju ili zglobova u hijerarhiji. Može se promatrati i kao jedna sličica u animaciji. Definicija je prikazana u isječku koda 2.5. Sadrži dva paralelna vektora – jedan za strukture `Transform` zglobova i drugi vektor za zapis roditelja svakog zgloba.

```
class Pose {
  std::vector<Transform> mJoints;
  std::vector<int> mParents; }
```

Isječak koda 2.5: Klasa `Pose`



Slika 2.6: Ovisnosti objekata pri generiranju poze

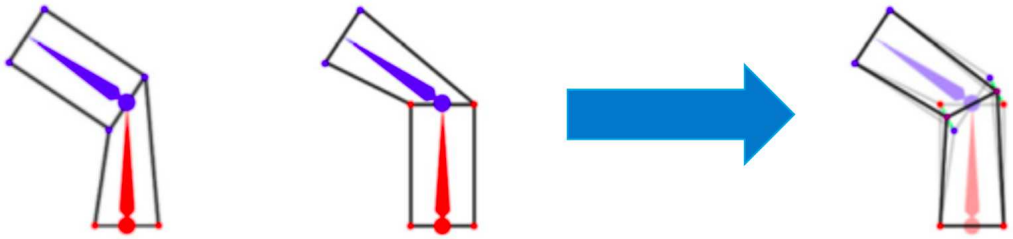
Nakon kreiranja svih pomoćnih struktura i klasa, potrebno je omogućiti prikaz animiranog modela. Za prikaz modela, potrebno je implementirati postupak *mesh skinning*. To je postupak deformiranja mreže poligona (engl. *mesh*) tako da odgovara animiranoj pozi. Može se izvoditi na procesoru pomoću C++ koda *CPU skinning* ili pomoću sjenčara na grafičkoj kartici *GPU skinning*. Mreža poligona je sastavljena od vrhova. Svaki vrh ima barem poziciju i normalu, a često i koordinate teksture. Kad se trodimenzionalni model modelira, on se modelira u određenoj pozi. Za ljudske likove, to je često tzv. T poza (engl. *T pose*) prikazana na slici 2.7. Jednom kad je modelirana mreža poligona, stvara se kostur. Svaki vrh mreže poligona je pridružen barem jednoj kosti kostura – taj proces se naziva *rigging* i taj kostur predstavlja *bind* pozu.

Mesh skinning je zapravo proces određivanja koji zglob ili kost utječe na deformaciju pojedinog vrha. Na jedan vrh može utjecati više zglobova. U postupku *rigid skinning* na svaki vrh utječe samo jedna kost. Za taj postupak, svaki vrh prolazi kroz niz operacija. Nakon učitavanja mreže vrhova, svi vrhovi su u koordinatnom sustavu modela. Vrh se prvo množi inverznom matricom *bind* poze zgloba kojem pripada. Zatim se množi matricom trenutne poze zgloba kojem pripada da se vrati u sustav modela. Nakon toga slijedi standardno množenje s *ModelView* matricom koja ga stavlja u sustav kamere i konačno množenje s matricom projekcije koja stavlja vrh u sustav projekcije za prikaz na ekranu. Pored tog postupka, postoji postupak *smooth skinning* u kojem na svaki vrh utječe više kostiju i taj postupak je danas standard u animacijama.

U većini igara, uzima se da četiri kosti utječu na svaki vrh u postupku *mesh skinning* i takav postupak je implementiran u sklopu ovog diplomskog rada. Da bi takav postupak radio, potrebno je pamtiti koje kosti i kojom težinom utječu na svaki vrh. U početnu strukturu vrha se zato dodaju dva vektora s četiri dimenzije – vektor koji pamti indekse kostiju koje utječu na njega i vektor težina kojom svaka od tih kosti utječe na vrh.[9]

Slika 2.7: T poza

Prijelaz iz jedne animacije u drugu može izgledati neskladno, posebice u slučaju kad lik u trenu promijeni animaciju. Taj problem se rješava generiranjem prijelaznih okvira animacije koji su prosjek dviju animacija. Na taj način prijelaz animacije iz jedne u drugu izgleda skladnije i prirodnije. Taj prijelaz je uglavnom kratak – traje oko četvrtine sekunde. Primjer generiranja takvog prijelaznog okvira prikazan je na slici 2.8.[7]



Slika 2.8: Miješanje dviju animacija

3. Metode prepoznavanje poze čovjeka iz videozapisa

Veliki napredak je postignut u prepoznavanju 3D poze i oblika čovjeka iz jedne slike. Ti postupci imaju široku primjenu, ali ipak ne nose dovoljno informacija o pokretu i ponašanju čovjeka. Više informacija se saznaje iz čovjeka u pokretu. Iz tog razloga, proučavaju se mnoge metode prepoznavanja poze čovjeka iz videozapisa što je i glavna tema ovog diplomskog rada. Pouzdane metode za tu primjenu još nisu dostupne zato što često ne uspijevaju proizvesti precizno prepoznavanje. Jedan od razloga je manjak točnih 3D oznaka ljudske poze, a njih nije jednostavno pribaviti ni za slike, a posebice za videozapise. Većina metoda za prepoznavanje poze čovjeka iz videozapisa se temelji na metodama za prepoznavanje poze iz slike pa će se prvo obraditi postupci za prepoznavanje poze iz slike.[5]

3.1. Prepoznavanje poze i oblika iz jedne slike

Kao rezultat metoda prepoznavanja poze i oblika čovjeka iz slike često se koriste parametrizirani 3D modeli tijela. Takvi modeli dobro odražavaju oblik čovjeka i stvaraju 3D mrežu točaka (engl. *mesh*) koju je moguće koristiti za razne primjene. Početne metode su bile krhke, zahtijevale su ručnu intervenciju čovjeka i uglavnom nisu dobro generalizirale. Jedna od prvih metoda koja implementira *end-to-end* pristup – prepoznavanje 3D poze i oblika čovjeka iz jedne slike je *SMPLify*.[1]

3.1.1. SMPLify

Metodom *SMPLify* se može automatski kreirati 3D poza tijela čovjeka, kao i njegov 3D oblik iz jedne slike. Time je pokazano da zglobovi s dvodimenzionalne slike nose mnogo informacija o obliku tijela. Taj problem je izazovan zbog kompleksnosti ljudskog tijela, izražajnosti, odjeće, osvjetljenja, kao i samog nejednoznačnog preslikava-

nja informacija iz 2D podataka u 3D. Kao rješenje, koristi se konvolucijska neuronska mreža (engl. *convolutional neural network, CNN*) `DeepCut` za prepoznavanje lokacija zglobova na slici.[6] Nakon toga, statistički model tijela `SMPL` (engl. *A Skinned Multi-Person Linear Model*) se prilagodi prema prepoznatim zglobovima tako da ciljna funkcija kažnjava grešku između projiciranih 3D zglobova i prepoznatih zglobova. `SMPL` model je realističan 3D model tijela čovjeka naučen na tisućama skeniranja ljudskih tijela. Ključna funkcionalnost `SMPL` modela je da može biti prilagođen maloj količini podataka zato što obuhvaća vrlo mnogo informacija o obliku i korelaciji raznih dijelova ljudskog tijela. `SMPL` je generativni model kojim se opisuje oblik tijela – proporcije tijela, visina, masa i poza tijela – deformacije 3D površina ovisno o pokretu čovjeka.

Poza i oblik se optimiziraju na način da su projicirani zglobovi 3D modela blizu 2D zglobovima sa slike koje je prepoznala konvolucijska neuronska mreža. Mnoge prijašnje metode u ovom području su kreirale 3D štapićaste figure iz 2D zglobova. Korištenjem takvih modela, bilo je lako uočiti poze koje su zapravo nemoguće jer bi se dijelovi ljudskog tijela presijecali u 3D. Takvi rezultati su česti kod preslikavanja podataka iz 2D u 3D zbog manjka podataka o dubini. Taj problem se rješava računanjem međusobnog prodiranja (engl. *interpenetration*) modela. No, taj postupak je prilično skup za kompleksan model poput ljudskog tijela. *SMPLify* metoda taj postupak pojednostavljuje na način da definira skup "kapsula" koje približno opisuju oblik tijela i zapravo djeluju kao sudarači (engl. *collider*). Dimenzije tih kapsula se određuju iz parametara oblika tijela. Tako je postupak računanja međusobnog prodiranja ubrzan i riješen problem nemogućih poza. `SMPL` model prepoznaje spol čovjeka sa slike i kreira model odgovarajućeg spola za točniji rezultat. Ako u nekoj situaciji spol nije moguće odrediti, kreira se model koji je neovisan o spolu.[1]

U nekim nedavnim metodama, duboke neuronske mreže se uče da prilagode parametre `SMPL` modela direktno s piksela bez prepoznavanja zglobova. Također, razvijene su metode koje stvaraju neparametarski model čovjeka. Jedna od takvih je metoda koja koristi vokselizirani model kao rezultat. Iako postoji mnogo različitih metoda za prepoznavanje čovjeka sa slike, sve takve metode proizvode nestabilne rezultate kad se primijene na videozapise.

3.2. Prepoznavanje poze i oblika iz videozapisa

Prve metode za prepoznavanje poze i oblika iz videozapisa su bile ograničene na jednostavne pokrete poput hodanja. Neke od novijih metoda koriste mehanizme dubokog

učenja koristeći samo pozicije zglobova. Postoje metode koje koriste pristup u dva koraka da bi preslikale početne 2D zglobove u 3D zglobove. No takve metode nisu davale zadovoljavajuće rezultate na novim, neviđenim podacima.

Također, neke metode koriste `SMPL` model tako da koriste metodu *SMPLify* kroz vrijeme trajanja videozapisa za stvaranje konzistentnog oblika i glatke pokrete tijela. Takvim metodama nastaju bolji rezultati. Za daljnje poboljšanje performansi, uvode se vremenski modeli temeljeni na samonadgledanim mrežama (engl. *Self-attention-based networks; transformers*).

3.3. Generativni suparnički modeli za modeliranje sekvenci

Generativni suparnički modeli (engl. *Generative adversial network, GAN*) su imali značajan utjecaj na modeliranje i sintezu slika. Generativni suparnički modeli se koriste u povratnim arhitekturama za modeliranje sekvencijskih zadataka poput strojnog prevođenja. Primarna namjena *GAN*-a je generiranje uzoraka. *GAN* se sastoji od dvije mreže: generatora i diskriminatora. Te dvije mreže su protivnici (engl. *adversaries*). Jedna pokušava prevariti drugu, dok druga pokušava prepoznati prevaru. Mreža generator generira umjetne uzorke, a mreža diskriminator pokušava otkriti umjetno generirane uzorke. Njihovo natjecanje ih tjera da budu sve bolji u svojim pokušajima. Na kraju, generator generira umjetne uzorke koje diskriminator ne može razlikovati od pravih uzoraka.[8] Sekvencijskim arhitekturama i suparničkim učenjem se može predvidjeti buduća sekvenca pokreta na temelju prošlih sekvenci ili generirati sekvenca pokreta čovjeka. Suparničko učenje se može koristiti i za poboljšanje predviđenih poza nad sekvencijskim ulaznim podacima. U tom kontekstu, kreira se mreža diskriminator koja pokušava razlikovati predviđene poze i poze dobivene uzorkovanjem pokreta (engl. *motion capture, MoCap*).

4. Sustav VIBE

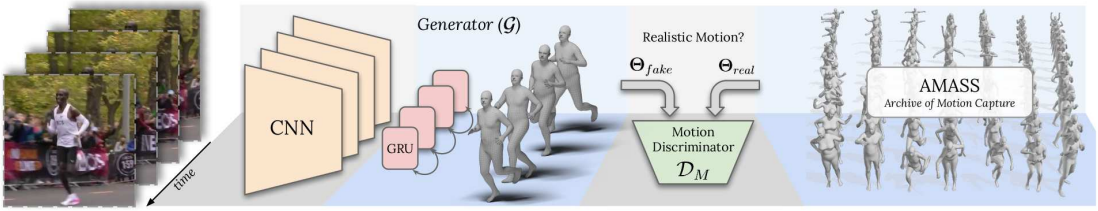
VIBE je sustav za prepoznavanje poze čovjeka iz videozapisa koji koristi skup podataka AMASS. Cilj sustava je razviti mrežu diskriminator koja neće moći razlikovati predviđene pokrete i pokrete iz skupa AMASS. Izlaz sustava VIBE je sekvenca parametara poze i oblika u SMPL formatu. Za neki videozapis čovjeka, prvo se uči vremenski model koji predviđa SMPL parametre tijela za svaku sliku (engl. *frame*) videozapisa dok diskriminator pokušava prepoznati razlike između stvarnih i generiranih sekvenci. Tako se generator potiče da stvara poze uvjerljivog ljudskog pokreta minimizirajući suparničku pogrešku učenja. Tako diskriminator implicitno uči fiziku, statiku i kinematiku ljudskog tijela koristeći podatke dobivene uzorkovanjem pokreta čovjeka (engl. *motion capture*).

Tijekom učenja, VIBE dobiva na ulaz slike čovjeka i predviđa SMPL parametre tijela čovjeka sa slike koristeći konvolucijsku neuronsku mrežu (engl. *CNN*) unaprijed naučenu za predviđanje poze i oblika čovjeka. Nakon toga slijede vremenski koder (engl. *temporal encoder*) i generator parametara tijela. Zatim diskriminator pokreta prima predviđene poze zajedno s pozama iz skupa AMASS i određuje je li ta sekvenca stvarna ili lažna.

4.1. Opis sustava VIBE

Arhitektura sustava VIBE je prikazana na slici 4.1. Slika je preuzeta iz [5]. Sustav koristi model čovjeka koji je neovisan o spolu. Za ulazni videozapis jednog čovjeka $V = \{I_t\}_{t=1}^{T}$ duljine T, izdvajaju se značajke svake slike (engl. *frame*) I_t koristeći unaprijed naučenu konvolucijsku neuronsku mrežu. Vremenski koder se sastoji od dvosmjernih propusnih povratnih ćelija (engl. *bidirectional Gated Recurrent Units, GRU*) koji kao rezultat daje latentne varijable koje sadrže informacije uključene u prošle i buduće slike (engl. *frame*). Te značajke se koriste za predviđanje SMPL parametara u svakom vremenskom trenutku.

SMPL opisuje pozu i oblik tijela parametrima Θ. Parametri Θ se sastoje od parame-



Slika 4.1: Arhitektura sustava VIBE

tara poze $\theta \in \mathbb{R}^{3\times K}$ i parametara oblika tijela $\beta \in \mathbb{R}^{10}$. Parametri poze tijela $\theta \in \mathbb{R}^{72}$ uključuju globalnu rotaciju tijela i relativne rotacije $K = 23$ zgloba u formatu os-kut (engl. *axis-angle format*). Parametri oblika tijela β su zapravo prvih deset koeficijenata PCA prostora oblika. Analiza glavnih komponenti (engl. *principal components analysis, PCA*) je statistička tehnika za određivanje varijabli koje čine koherentne podskupove koji su međusobno relativno nezavisni. Često se koristi za smanjenje dimenzionalnosti podataka. Koristeći navedene parametre, `SMPL` model je diferencijabilna funkcija $M(\theta, \beta) \in \mathbb{R}^{6890\times 3}$ koja kao rezultat daje trodimenzionalnu mrežu točaka (engl. *mesh*) u određenoj pozi s $N = 6980$ vrhova (engl. *vertices*) mreže. To se postiže oblikovanjem predloška tijela prema parametrima θ i β, pomicanjem kostiju sukladno rotacijama zglobova θ koristeći unaprijednu kinematiku.[3]

Za ulazni videozapis, VIBE računa $\hat{\Theta} = [(\hat{\theta}_1, \ldots, \hat{\theta}_T), \hat{\beta}]$ gdje su $\hat{\theta}_t$ parametri poze tijela u trenutku t i $\hat{\beta}$ parametri oblika tijela za cijelu sekvencu videozapisa. Parametri oblika tijela se zapravo računaju u svakoj slici videozapisa. Nakon izračuna parametara tijela u svakoj slici videozapisa, provodi se postupak *average pooling* za dobivanje jedinstvenih parametara $\hat{\beta}$ za cijelu sekvencu ulaznog videozapisa. Do sad opisani model koji predviđa parametre se naziva generator G. Izlaz generatora G, $\hat{\Theta}$, i uzorci iz skupa `AMASS`, Θ_{real} se predaju kao ulaz diskriminatoru pokreta, D_M. D_M pokušava razlikovati stvarne primjere od lažnih primjera.[5]

Modul iterativne 3D regresije

Cilj modula iterativne 3D regresije je izračunati parametre Θ za ulaznu sliku ϕ tako da je gubitak preslikavanja (engl. *reprojection error*) L_{reproj} minimalan.

$$L_{reproj} = \sum_i \|v_i(\mathbf{x_i} - \hat{\mathbf{x}}_\mathbf{i})\|_1$$

U izrazu za gubitak, $\mathbf{x_i} \in \mathbb{R}^{2\times K}$ označava i-ti 2D zglob na ulaznoj slici i $v_i \in 0, 1^K$ označava vidljivost svakog od K zglobova – ima vrijednost 1 ako je zglob vidljiv, a 0 inače.

Izravna regresija parametara Θ je zahtjevan zadatak, posebice zato što Θ sadrže parametre rotacije. Za pojednostavljivanje regresije parametara Θ koristi se petlja iterativne povratne veze pogreške (engl. *iterative error feedback loop, IEF*) gdje se povratnom vezom mijenjaju parametri trenutnog predviđanja. Modul prima značajke slike ϕ i trenutne parametre Θ_t na ulaz i na izlaz vraća razliku $\Delta\Theta_t$. Parametri se zatim prilagođavaju dodavanjem razlike na trenutne vrijednosti parametara: $\Theta_{t+1} = \Theta_t + \Delta\Theta_t$.[3] Takav modul se koristi u sustavu VIBE.

Mehanizam samopozornosti

Povratne neuronske mreže ažuriraju svoja skrivena stanja procesirajući slijedno podatke s ulaza. Kao rezultat, konačno skriveno stanje sadrži sažetak informacija cijelog slijeda ulaznih podataka. Mehanizam samopozornosti (engl. *self-attention mechanism*) se koristi za pojačavanje utjecaja najvažnijih sličica (engl. *frames*) videozapisa u konačnom prikazu umjesto korištenja konačnog skrivenog stanja. Koristeći taj mehanizam, konačan prikaz ulaznog slijeda $\hat{\mathbf{\Theta}}$ je naučena linearna kombinacija skrivenih stanja. Težine a_i se uče pomoću linearnog višeslojnog perceptrona (engl. *Multilayer Perceptron, MLP*) ϕ. Nad izlazima perceptrona se još provodi funkcija *softmax* da se dobije vjerojatnosna distribucija:

$$\phi_i = \phi(h_i), a_i = \frac{e^{\phi_i}}{\sum_{t=1}^{N} e^{\phi_t}}, r = \sum_{i=1}^{N} a_i h_i$$

Ovaj mehanizam se koristi u diskriminatoru pokreta D_M koji je pojašnjen u nastavku ovog diplomskog rada.

4.1.1. Vremenski koder

Buduće sličice (engl. *frames*) mogu iskoristiti informacije o pozama iz prošlih sličica. Zbog toga se koristi povratna arhitektura kod vremenskog kodera. To je korisno u slučajevima kad je poza čovjeka nejasna ili je tijelo čovjeka djelomično skriveno. U tim slučajevima, informacije iz prošlih sličica pomažu kod određivanja i ograničavanja predviđene poze. Vremenski koder se ponaša kao generator koji za slijed sličica $I_1, \ldots, I_T$ određuje odgovarajuće parametre poze i oblika tijela za svaku sličicu I_t. Slijed T sličica se šalje kao ulaz u konvolucijsku neuronsku mrežu f koja radi kao ekstraktor značajki. Izlaz te mreže je vektor $f_i \in \mathbb{R}^{2048}$ za svaku sličicu: $f(I_1), \ldots, f(I_T)$. Taj izlazni vektor se zatim šalje u sloj propusnih povratnih ćelija (engl. *Gated Recurrent Unit layer, GRU layer*) koji vraća vektor latentnih značajki g_i za svaku sličicu:

$g(f_1), \ldots, g(f_T)$ koristeći informacije iz prošlih sličica. Zatim se vektor latentnih značajki g_i šalje kao ulaz u T modula iterativne 3D regresije. Svaki modul se inicijalizira s prosjekom poza $\overline{\Theta}$ i na ulaz prima trenutne parametre Θ_k i značajke g_i u svakoj iteraciji k.

Ukupni gubitak opisanog vremenskog kodera L_G jednak je zbroju 2D, 3D gubitaka i gubitaka poze i oblika:

$$L_G = L_{2D} + L_{3D} + L_{SMPL}$$

Pojedini gubitak definiran je sljedećim izrazom:

$$\begin{aligned} L_{2D} &= \sum_{t=1}^{T} \|x_t - \hat{x}_t\|_2 \\ L_{3D} &= \sum_{t=1}^{T} \left\| X_t - \hat{X}_t \right\|_2 \\ L_{SMPL} &= \left\| \beta - \hat{\beta} \right\|_2 + \sum_{t=1}^{T} \left\| \theta_t - \hat{\theta}_t \right\|_2 \end{aligned}$$

Za izračun 2D gubitka, potrebne su 3D lokacije `SMPL` zglobova $\hat{X}(\Theta) = WM(\theta, \beta)$ koje se računaju iz vrhova tijela koristeći unaprijed naučeni model linearne regresije W. Koristi se *weak-perspective* model kamere. Takva projekcija koristi iste principe kao i ortografska projekcija, ali zahtijeva specificiranje faktora skaliranja i tako osigurava da bliži objekti izgledaju veće u projekciji i obratno. Može se promatrati kao prijelazna projekcija između ortografske i perspektivne i opisuje se kao ortografska projekcija kojoj se još nadodaje skaliranje. Za takvu projekciju, koriste se parametar skaliranja $s \in \mathbb{R}$ i parametri translacije $t \in \mathbb{R}^2$. S ovim parametrima, 2D projekcija 3D lokacija $\hat{X}$ se računa kao: $\hat{x} = s\Pi(R\hat{X}(\Theta)) + t$ gdje je $R \in \mathbb{R}^{3\times3}$ matrica globalne rotacije i Π označava ortografsku projekciju.[5]

4.1.2. Diskriminator pokreta

Diskriminator pokreta i gubitak preslikavanja (engl. *reprojection loss*) potiču generatora da stvara uvjerljive poze iz stvarnog svijeta. Međutim, ograničenja koja proizlaze iz jedne slike nisu dovoljna za slijed poza. Mnoge netočne poza bi mogle biti prepoznate kao valjane ako bi se zanemario vremenski slijed među njima. Da bi se smanjila ta pogreška, koristi se diskriminator pokreta D_M koji za generirani slijed poza odlučuje podudara li se sa stvarnim slijedom ili ne. Izlaz generatora, $\hat{\Theta}$, se predaje kao ulaz u višeslojni model propusnih povratnih ćelija (engl. *GRU*) f_M koji je prikazan na slici 4.2. Slika je preuzeta iz [5]. Izlaz tog modela je latentna varijabla $h_i = f_m(\hat{\Theta}_i)$ za svaki vremenski trenutak i. Kod agregiranja skrivenih stanja $[h_1, \ldots, h_T]$, tj. kod određivanja konačnog stanja koristi se mehanizam samopozornosti (engl. *self-attention*

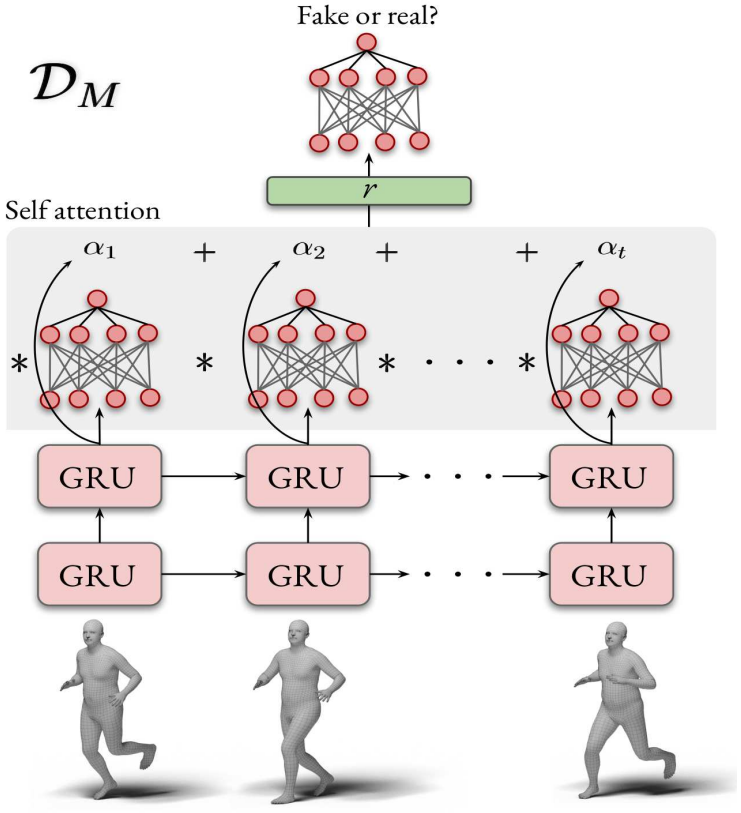
mechanism) opisan ranije u diplomskom radu. U konačnici, linearni sloj računa izlaznu vrijednost $D_M(\hat{\boldsymbol{\Theta}})$ iz intervala $[0, 1]$ koja predstavlja vjerojatnost da $\hat{\boldsymbol{\Theta}}$ pripada u uvjerljive ljudske pokrete. Ukupnom gubitku vremenskog kodera L_G se još dodaje suparnički gubitak L_{adv} (engl. *adversarial loss*):

$$L_{adv} = \mathbb{E}_{\Theta \sim p_G}[(D_M(\hat{\boldsymbol{\Theta}}) - 1)^2]$$

Suparnički gubitak je zapravo očekivanje vjerojatnosti da generirani podaci nisu stvarni, tj. da su generirani podaci umjetni. Ciljna funkcija diskriminatora D_M je L_{D_M}:

$$L_{D_M} = \mathbb{E}_{\Theta \sim p_R}[(D_M(\boldsymbol{\Theta}) - 1)^2] + \mathbb{E}_{\Theta \sim p_G}[D_M(\hat{\boldsymbol{\Theta}})^2]$$

gdje je p_R slijed stvarnih pokreta iz skupa podataka `AMASS`, a p_G slijed generiranih pokreta.[5]



Slika 4.2: Diskriminator pokreta D_M

4.2. Implementacija

4.2.1. Korišteni alati

Sustav VIBE izrađen je koristeći programski jezik *Python* i razne pomoćne biblioteke. Korištena je programska biblioteka strojnog učenja *PyTorch*. *PyTorch* se često koristi u aplikacijama za računalni vid i obradu prirodnog jezika. Omogućeno je korištenje `CUDA` operacija. `CUDA` (engl. *Compute Unified Device Architecture*) je platforma koja NVIDIA grafičkim procesorima omogućava izvršavanje programa. `CUDA` omogućava veliko povećanje performansi oslanjajući se na snagu grafičkog procesora u računalu. `CUDA` poziva paralelne funkcije koje se izvršavaju u mnogo paralelnih dretvi.

Za učenje sustava koriste se podaci iz skupa podataka `AMASS` (engl. *Archive of Motion Capture as Surface Shapes*). To je veliki skup podataka pokreta ljudi koji objedinjuje više skupova podataka dobivenih mnogim uzorkovanjima pokreta čovjeka (engl. *motion capture*) koristeći zajedničku parametrizaciju. `AMASS` je koristan i često se koristi za animacije, vizualizacije i generiranje podataka za učenje u metodama dubokog učenja.[4]

Rezidualna neuronska mreža

Rezidualna neuronska mreža (engl. *residual neural network, ResNet*) je vrsta umjetne neuronske mreže u kojoj postoje skokovi među slojevima. Tipični modeli rezidualne mreže koriste preskakanje dva ili tri sloja neuronske mreže, tj. izlazi neurona iz jednog sloja postaju ulazi neurona u dva ili tri sloja iza tog sloja. Jedan od razloga zašto bi se preskakale veze između susjednih slojeva je izbjegavanje problema nestajećeg gradijenta pri učenju duboke neuronske mreže. Ako je preskakivanje učinkovito, mreža postaje jednostavnija i brže uči. Često korištena rezidualna neuronska mreža je tzv. *ResNet-50* mreža. Ta mreža se često koristi u mnogim zadacima računalnog vida. *ResNet-50* mreža ima 50 slojeva i može razlikovati, tj. klasificirati slike u tisuću kategorija.[2]

Optimizacijski algoritam Adam

Jedan od najčešće korištenih optimizacijskih algoritama neuronskih mreža je gradijentni spust. U tom algoritmu, računa se gradijent funkcije gubitka u ovisnosti o parametrima mreže. Zatim se ti parametri podešuju u smjeru suprotnom od gradijenta. Optimizacijski algoritam Adam (engl. *Adaptive Moment Estimation*) je nadogradnja stohastičkog gradijentnog spusta. Kod stohastičkog gradijentnog spusta, stopa učenja

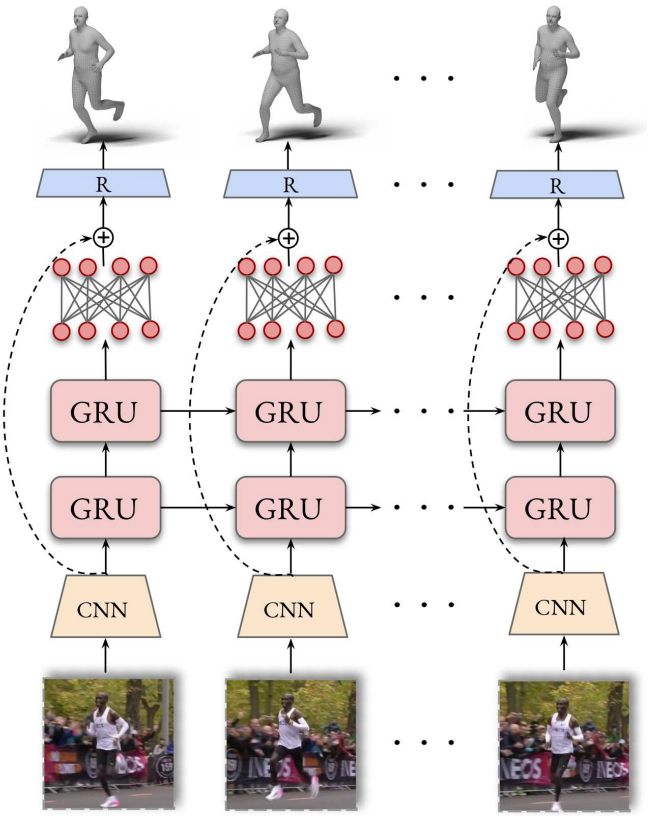
je jednaka za sve težine i ostaje konstanta tijekom učenja. Algoritam Adam koristi dva bitna poboljšanja stohastičkog gradijentnog spusta:

– AdaGrad (engl. *Adaptive Gradient Algorithm*) algoritam koji koristi zasebne stope učenja za svaki parametar (težinu) mreže. Tim algoritmom se poboljšavaju performanse na problemima s nestajućim gradijentima poput obrade prirodnog jezika i računalnog vida.

– RMSProp (engl. *Root Mean Square Propagation*) algoritam koji također koristi zasebne stope učenja za svaki parametar. Stope učenja se mijenjaju ovisno o iznosu prethodnih gradijenata za svaki parametar.

Adam je vrlo popularan algoritam u raznim problemima dubokog učenja jer brzo postiže dobre rezultate.

4.2.2. Učenje sustava



Slika 4.3: Generator poze G

Arhitektura generatora poze G je prikazana na slici 4.3. Slika je preuzeta iz [5]. Koristi se unaprijed naučena *ResNet-50* mreža za prepoznavanje poze i oblika čovjeka

sa slike. Za duljinu slijeda se koristi $T = 16$ i veličina mini-grupe (engl. *mini-batch*) za učenje je 32 što omogućuje učenje na jednoj NVIDIA GTX 1660 Ti grafičkoj kartici. Za vremenski koder, koristi se dvoslojan *GRU* sa skrivenom veličinom 1024. Model za regresiju `SMPL` modela sadrži 2 potpuno povezana sloja od kojih svaki sadrži 1024 neurona. Na ta dva sloja, nadovezuje se izlazni sloj koji na ulaz još prima i izlaz *ResNet-50* mreže kao pomoć pri učenji koji na izlazu vraća $\hat{\mathbf{\Theta}} \in \mathbb{R}^{85}$ – parametre poze, oblika i kamere.

Izlazi generatora se predaju kao ulazi diskriminatoru pokreta D_M kao lažni uzorci zajedno sa stvarnim uzorcima pokreta čovjeka. Arhitektura diskriminatora pokreta jednaka je arhitekturi vremenskog kodera. Za mehanizam samopozornosti (engl. *self-attention*), koriste se 2 sloja perceptrona od kojih svaki sadrži 1024 neurona. Za aktivacijsku funkciju koristi se funkcija th za učenje težina među neuronima. Posljednji linearni sloj računa vrijednost iz intervala $[0, 1]$ i predviđa vjerojatnost da je uzorak na ulazu stvaran. Koristi se optimizacijski algoritam Adam sa stopom učenja 5×10^{-5} za generator G i stopom učenja 1×10^{-4} za diskriminator D_M.

Korišteni skupovi podataka

Za učenje i testiranje sustava korišteni su razni skupovi podataka:[5]

- `MPI-INF-3DHP` – skup podataka dobiven uzorkovanjem pokreta čovjeka bez markera, sadrži podatke o pokretu u zatvorenom prostoru. Koristi se u postupku učenja.
- `Human3.6M` – sadrži 15 akcijskih slijedova dobivenih uzorkovanjem u kontroliranom okruženju. Sastoji se od 1.5 milijuna slika s 3D oznakama. Koristi se u postupku učenja i testiranja.
- `3DPW` – sastoji se od 60 videozapisa (24 za učenje, 24 za testiranje i 12 za validaciju) s raznim aktivnostima u otvorenom i zatvorenom prostoru. Koristi se u postupku učenja i testiranja.
- `PennAction` – sadrži 2326 video slijedova 15 različitih pokreta i 2D oznake za svaki slijed. Oznake uključuju zglobove čovjeka (2D lokacije i vidljivost) i 2D omeđujući pravokutnik (engl. *bounding box*). Koristi se u postupku učenja i testiranja.
- `InstaVariety` – noviji skup podataka koji sadrži videozapise s društvene mreže *Instagram*. Sadrži videozapise u ukupnom trajanju od oko 24 sata s 2D oznakama. Koristi se u postupku učenja i testiranja.

– `PoseTrack` – sadrži 1337 videozapisa podijeljenih u tri skupa: 792 videozapisa za učenje, 375 videozapisa za testiranje i 170 za validaciju. Koristi se u postupku učenja.

4.3. Rezultati

Model VIBE je uspješan u stvaranju realističnih i ispravnih poza i oblika čovjeka iz videozapisa. Budući da je rezultate iz videozapisa nemoguće prikazati na papiru, prikazani su rezultati prepoznavanja poze u jednom trenutku izvođenja na slikama 4.4 i 4.5. Na slici 4.4 je prikazan rezultat na videozapisu koji je preuzet s internetske stranice *YouTube*, a na slici 4.5 je prikazan rezultat na videozapisu na kojem se pojavljujem ja u svrhu testiranja rada.

(a) Pogled s pozicije kamere

(b) Pogled sa strane

Slika 4.4: Videozapis preuzet s *YouTube*-a

(a) Pogled s pozicije kamere

(b) Pogled sa strane

Slika 4.5: Moj videozapis

5. Zaključak

Ljudski pokret jedan je osnova za razumijevanje ljudskog ponašanja. Unatoč napretku u prepoznavanju 3D poza i oblika čovjeka iz slike, prijašnji modeli nisu proizvodili zadovoljavajuće rezultate u prepoznavanju pokreta ljudi u videozapisu. Sustav VIBE se temelji na prijašnjim modelima, ugrađuje pojedina poboljšanja i tako ih nadmašuje. Poboljšanja koja su ključna u sustavu VIBE su:

1. Povratna arhitektura koja propagira informacije iz videozapisa kroz vrijeme,
2. Učenje pomoću diskriminatora pokreta koristeći skup podataka `AMASS` i
3. Mehanizam samopozornosti u diskriminatoru koji omogućuje da neke sličice videozapisa nose više informacije od drugih.

Postoji prostor za daljnja poboljšanja ovog sustava. Buduća poboljšanja bi uključivala razlikovanje i rad sustava za više osoba na videozapisu. Također, moguće poboljšanje je i mogućnost stvaranja 3D modela čovjeka u `FBX` ili `glTF` formatu koji bi se zatim mogao koristiti u alatima za 3D oblikovanje poput *Blendera* i *Maye* ili grafičkim pogonima za igre poput *Unityja*.

LITERATURA

[1] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter V. Gehler, Javier Romero, i Michael J. Black. Keep it SMPL: automatic estimation of 3d human pose and shape from a single image. *CoRR*, abs/1607.08128, 2016. URL http://arxiv.org/abs/1607.08128.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[3] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, i Jitendra Malik. End-to-end recovery of human shape and pose. *CoRR*, abs/1712.06584, 2017. URL http://arxiv.org/abs/1712.06584.

[4] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, i Michael J. Black. AMASS: Archive of motion capture as surface shapes. U *International Conference on Computer Vision*, stranice 5442–5451, Listopad 2019.

[5] J. Black Muhammed Kocabas, Nikos Athanasiou. VIBE: video inference for human body pose and shape estimation. *CoRR*, abs/1912.05656, 2019. URL http://arxiv.org/abs/1912.05656.

[6] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V. Gehler, i Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. *CoRR*, abs/1511.06645, 2015. URL http://arxiv.org/abs/1511.06645.

[7] Tim studenata. *Odabrane teme iz područja računalne grafike - tehnička dokumentacija*, stranice 28–33. Fakultet elektrotehnike i računarstva Sveučilišta u Zagrebu Zagreb, http://www.zemris.fer.hr/predmeti/ra/Projekti/21_Tehn/Odabrane_teme_iz_podrucja_racunalne_grafike.pdf, 2021.

[8] Marko Subašić. *Generative Adversarial Networks: Predavanje iz kolegija Duboko učenje*. http://www.zemris.fer.hr/~ssegvic/du/du9gan.pdf.

[9] Gabor Szauer. *Hands-On C++ Game Animation Programming*. Packt Publishing Ltd., Birmingham, 2020.

Prepoznavanje poze čovjeka i stvaranje 3D modela iz videozapisa

Sažetak

U ovom diplomskom radu opisuje se kako radi jednostavan sustav animacije čovjeka. Opisane su pomoćne strukture podataka pri izradi sustava animacije i procesi koji se odvijaju tijekom prikazivanja animacije. Opisan je postupak VIBE za prepoznavanje poze čovjeka iz videozapisa. Objašnjen je postupak učenja sustava VIBE i prikazani su rezultati tog sustava. Rezultat rada je funkcionalan sustav animacije izrađen u *OpenGL*-u koristeći programski jezik C++. Korisnik može tipkovnicom mijenjati animaciju koja se prikazuje i podešavati vrijeme prijelaza između jedne animacije u drugu. Također, rezultat rada je i funkcionalan sustav VIBE koji se može koristiti na proizvoljnom videozapisu za prepoznavanje poze čovjeka.

Ključne riječi: animacija, model čovjeka, prepoznavanje poze, diskriminator pokreta, VIBE, Python

Human Body Pose and Shape Estimation

Abstract

This thesis describes simple animation system. Supporting data structures and processes of animation system are described. System for human body pose estimation VIBE is described. It is described how to train VIBE. Results of VIBE are shown. Resulting work is functional animation system created using OpenGL and C++ programming language. User can choose which animation is shown and change fade time between animations. Resulting works is also functional VIBE system which can be used on arbitrary video for human body pose estimation.

Keywords: animation, human model, pose estimation, motion discriminator, VIBE, Python