SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1094

INTEGRACIJA VIRTUALNIH DINAMIČKIH LIKOVA I STVARNE SCENE

Marko Đomlija

Zagreb, lipanj 2015.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 6. ožujka 2015.

Predmet: **Diplomski rad**

DIPLOMSKI ZADATAK br. 1094

Pristupnik: **Marko Đomlija (0036461720)**
Studij: Računarstvo
Profil: Programsko inženjerstvo i informacijski sustavi

Zadatak: **Integracija virtualnih dinamičkih likova i stvarne scene**

Opis zadatka:

Proučiti postupke integracije virtualnih objekata i svarene scene, njihovog međusobnog utjecaja i problema koji se pri integraciji javljaju. Proučiti uređaj Kinect i mogućnosti koje on pruža u stvaranju dinamičkih likova te integraciji sa stvarnom scenom. Razmotriti mogućnosti ostvarivanja ekvivalentne funkcionalnosti iz video-zapisa, odnosno bez poznavanja dubinske informacije koju Kinect pruža. Ostvariti programsku implementaciju integracije virtualnih dinamičkih likova u stvarnim scenama koji zamjenjuju prepoznate stvarne likove. Načiniti ocjenu i usporedbu ostvarenih rezultata.
Izraditi odgovarajući programski proizvod. 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.

Zadatak uručen pristupniku: 13. ožujka 2015.
Rok za predaju rada: 30. lipnja 2015.

Mentor:

Prof. dr. sc. Željka Mihajlović

Djelovođa:

Doc. dr. sc. Igor Mekterović

Predsjednik odbora za diplomski rad profila:

Prof. dr. sc. Krešimir Fertalj

Sadržaj

Uvod

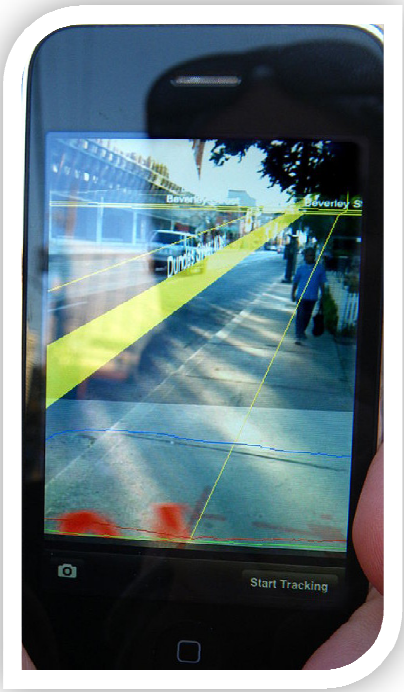
U radu je obrađena tema integracije virtualnih dinamičkih likova i stvarne scene, koju ugrubo možemo smjestiti u domenu proširene stvarnosti. Ukratko je opisana motivacija za razmatranje teme i konkretan problem kojim se rad bavi.

Prvo je ukratko izložen promatrani problem i njegovo načelno rješenje, potom je svaka faza algoritma detaljnije opisana. Također su dane reference na postojeća rješenja sličnih problema. Kao stvarna scena uzima se video zapis čijom se obradom dobivaju informacije o objektima na njoj. Na temelju te informacije u scenu se dodaju virtualni objekti.

Prezentirana je razvijena aplikacija koja rješava zadatak. Opisane su njene mogućnosti, rezultati koje ostvaruje, kao i tehnologija koja je korištena prilikom njene izrade. Na kraju se nalazi kratki osvrt na konačni rezultat.

1. Virtualni objekti na stvarnoj sceni

Razvoj računalne tehnologije, a samim tim i računalne grafike, značajno je promijenio ulogu računala u životu čovjeka. S porastom broja korisnika računala, prijenosnika, pametnih telefona, tableta i slično, množe se i ideje kako ih koristiti. Spoj virtualnog i stvarnog svijeta, odnosno proširena stvarnost, jedna je od tih ideja, a ujedno je i tema ovog rada. Danas postoje mnoge aplikacije s elementima proširene stvarnosti koje su u primjeni u raznim poljima ljudske djelatnosti: u vojnoj industriji, medicini, arhitekturi, edukaciji, zabavnoj industriji, navigaciji, itd. [1].



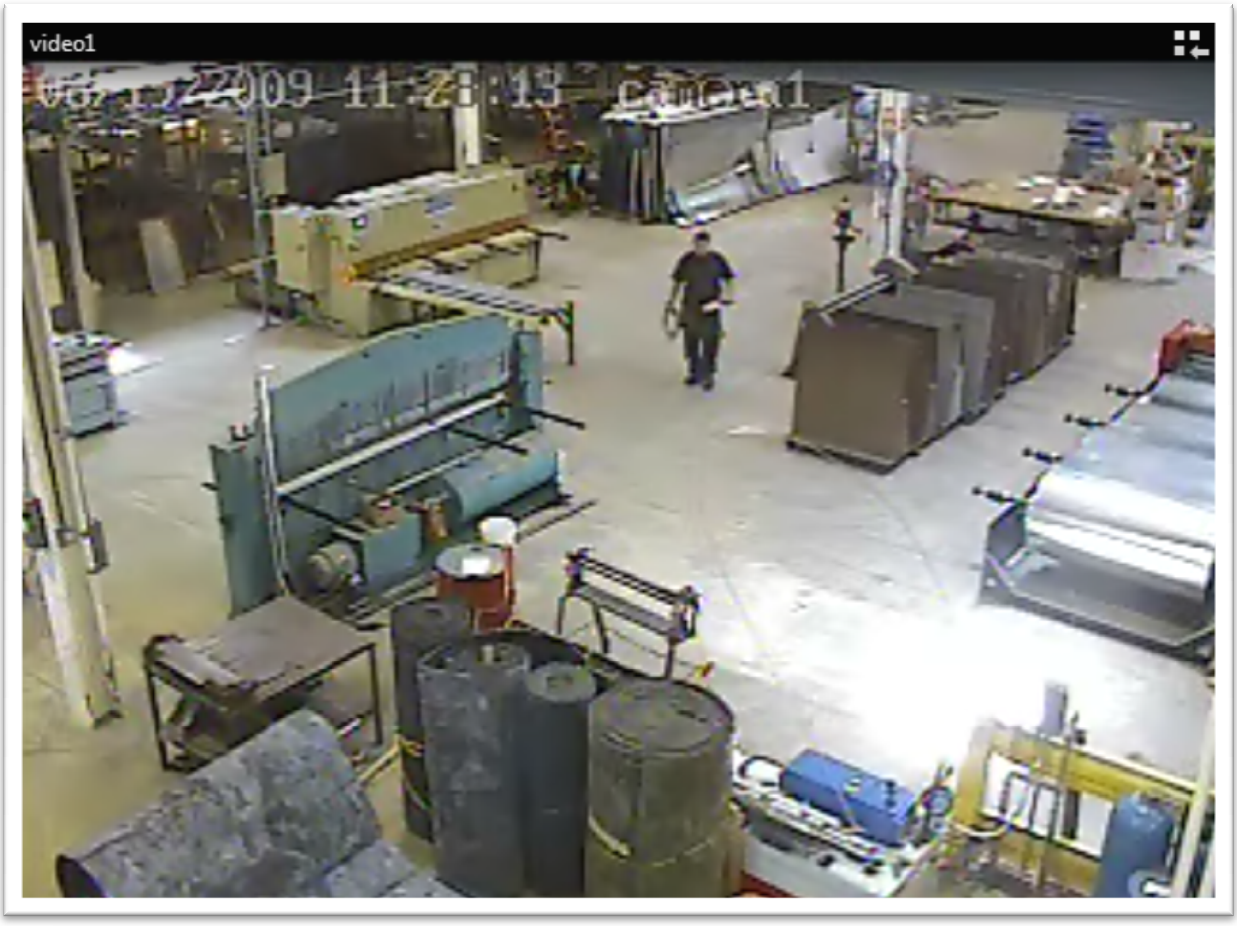
Slika 1.1: Aplikacija za navigaciju na iPhone-u [2]

Slika 1.1 prikazuje primjer proširene stvarnosti u praksti, aplikaciju na uređaju iPhone koja služi kao vizualna pomoć pri navigaciji.

Jedan od izazova koji je povezan s umetanjem dodatnog sadržaja u postojeću scenu je detekcija objekata na sceni i njihovo prepoznavanje. U nastavku ćemo pokušati razraditi jedan vid spomenutog problema na primjeru analize video sekvence. Kao rezultat analize trebali bismo dobiti podatke o lokaciji objekata koji se nalaze na sceni i, ukoliko se radi o ljudima, podatke o položaju njihovog tijela.

1.1. Analiza video sekvence

U ovom ćemo potpoglavlju preciznije odrediti kakvim ćemo se problemom baviti. Za scenu u koju će biti umetnuti virtualni objekti uzet ćemo video sekvencu. Kao ulaz u algoritam moguće je postaviti video materijal koji zadovoljava određena ograničenja i na kojem postoje objekti, bilo statični, bilo pokretni, koji bi nam mogli biti od interesa. Za sada nećemo razmatrati ograničenja koja vrijede za ulazne podatke, spomenut ćemo ih kad dođemo do opisa određene faze algoritma s kojim su ta ograničenja povezana.



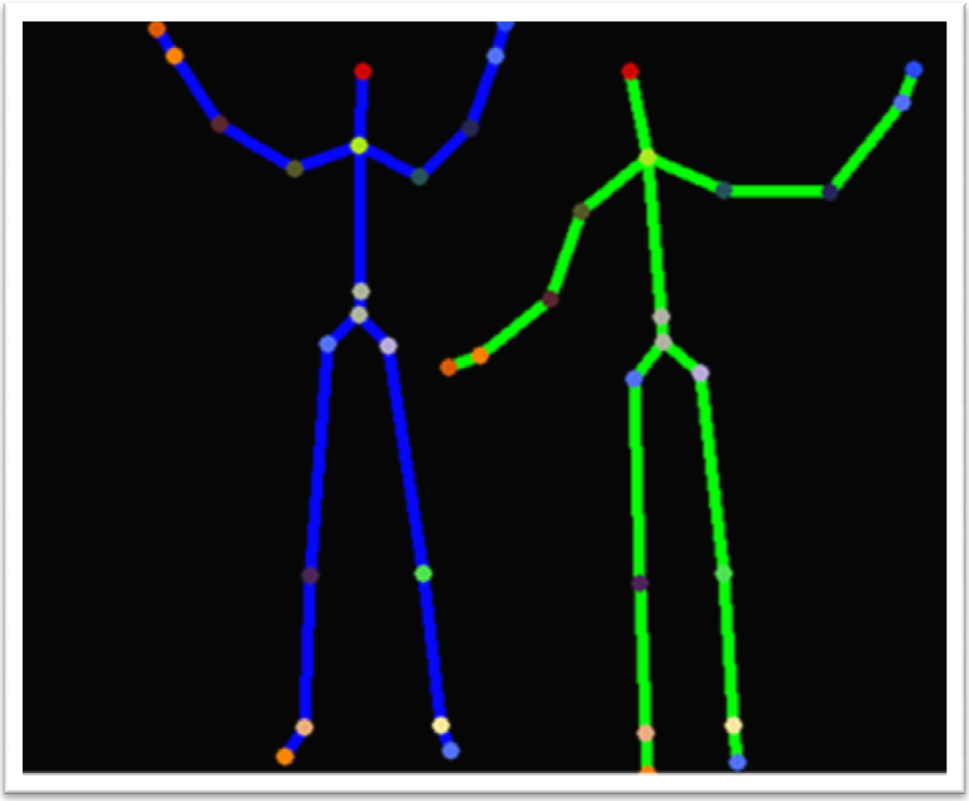
Slika 1.2: Snika nadzorne kamere u radionici [3]

Primjer isječka takvog videa nalazi se na slici (Slika 1.2). Riječ je o videu koji je snimila nadzorna kamera u radionici. Čovjek na slici primjer je objekta koji bismo želje analzirati i za njega dobiti podatke. Zadatak prilikom obrade video sekvence jest dobiti podatke o lokaciji detektiranih objekata. Ukoliko je riječ o ljudima, potrebno je dobiti podatke o položaju tijela na temelju kojih je moguće na poziciji čovjeka iscrati virtualnog čovječuljka. Ukoliko iz bilo kojeg razloga nije moguće dobiti odgovarajuće podatke,

dovoljno je na odgovarajućoj lokaciji iscrtati generički objekt koji signalizira da je objekt na sceni detektiran.

1.2. Praćenje položaja tijela

Kao rezultat obrade video sekvence trebali bismo dobiti podatke o lokacijama objekata na sceni. Zanimljiviji i izazovniji dio te analize je određivanje položaja tijela čovjeka s ciljem umetanja virtualnog lika u scenu. Jedan od poznatijih uređaja na tržištu koji obavlja spomenuti zadatak je Microsoft Kinect.

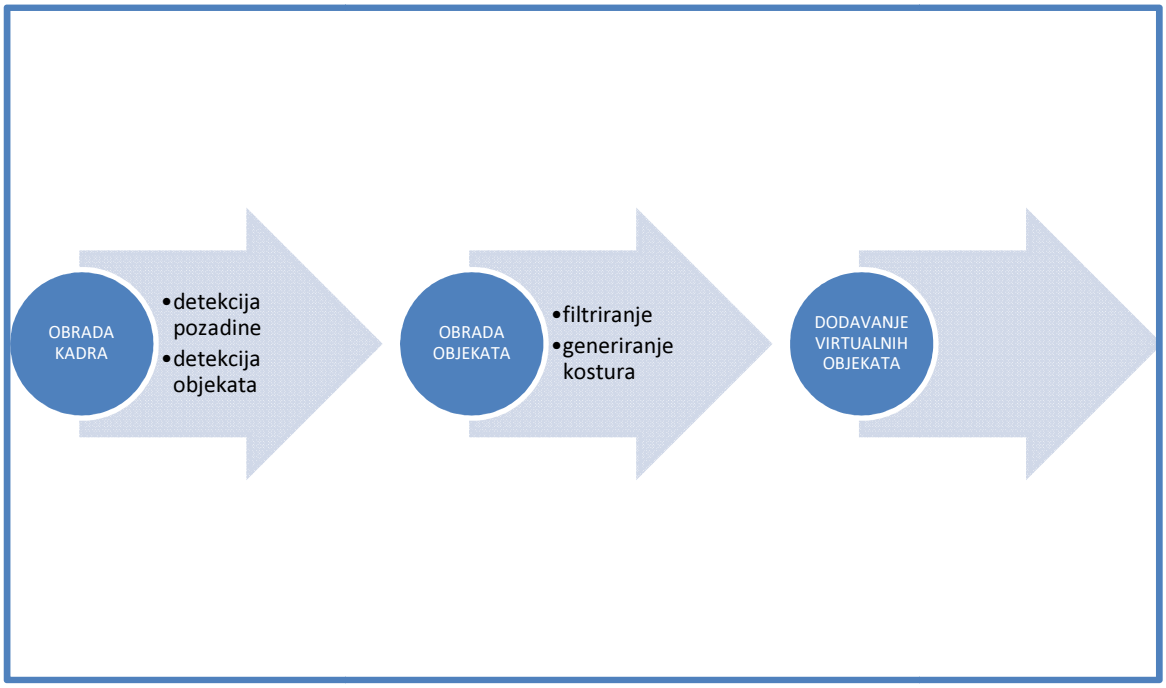


Slika 1.3: Praćenje kostura, Microsoft Kinect [4]

Primjer praćenja ljudskog kostura uz pomoć uređaja Microsoft Kinect prikazan je na slici (Slika 1.3). Cilj ovog rada bio bi istražiti u kojoj je mjeri moguće sličnu funkcionalnost postići analizom same video sekvence, bez dodatnih podataka o sceni kao što je dubinska informacija. Kako bi to bilo uspješno obavljeno, dovoljno je odrediti lokacije nekoliko ključnih točaka na čovjeku kako bi iz njih bilo moguće kreirati morfološki kostur. Jednostavnosti radi, ograničit ćemo se na određivanje položaja glave, vrata, ramena, laktova, šaka, kralježnice, kukova, koljena i stopala.

2. Prikaz algoritma

U nastavku ćemo prezentirati skicu rješenja, odnosno načelne korake koje ćemo poduzeti kako bismo iz ulaza, odnosno video sekvence, dobili izlaz, odnosno podatke o pozicijama objekata.

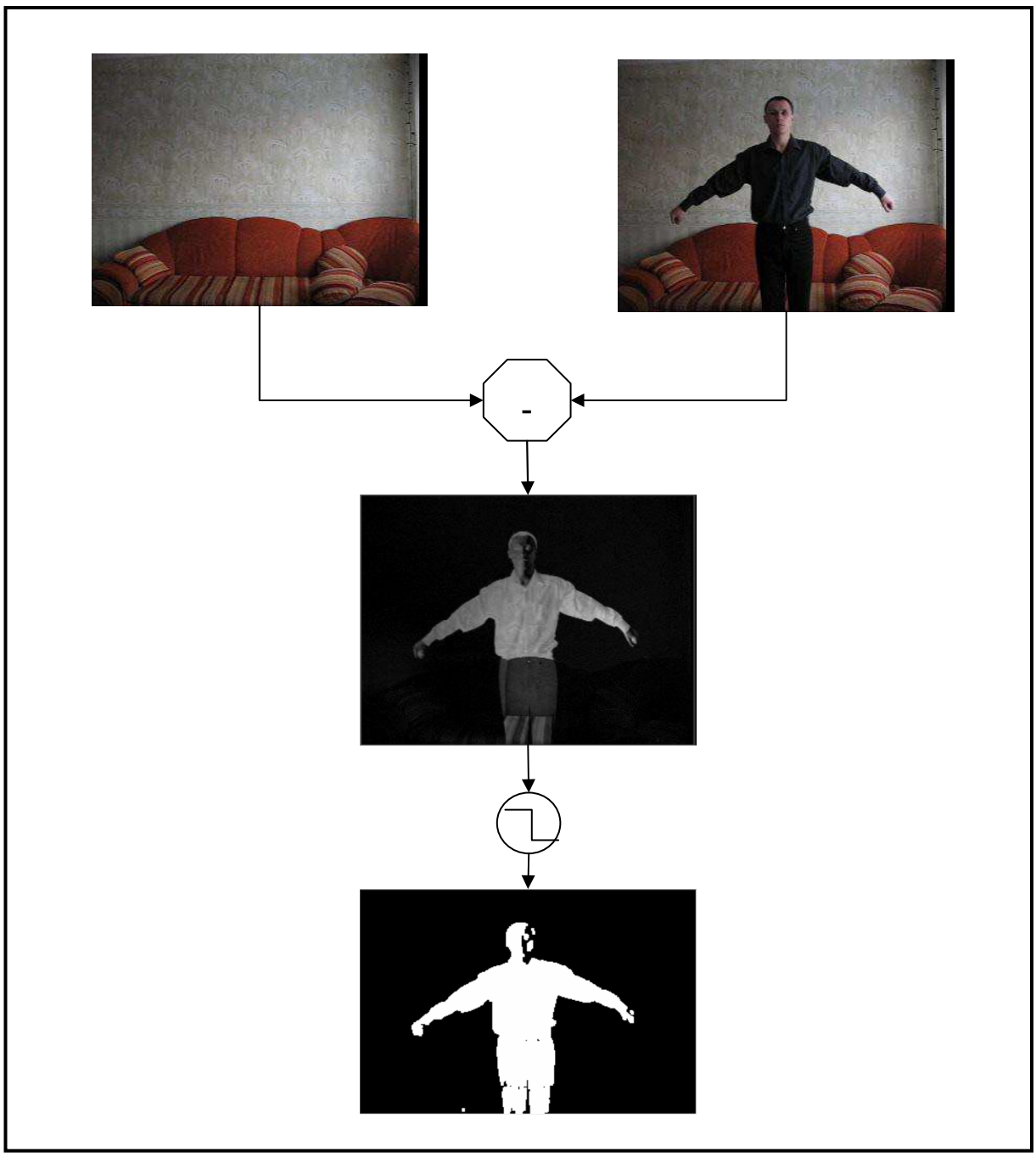


Slika 2.1: Proces obrade ulaza

Slika 2.1 prikazuje proces obrade ulaznih podataka. Prvi korak u analizi je obrada kadra, odnosno jedne sličice video zapisa. Pritom je potrebno razlučiti što u kadru tvori pozadinu, a što su objekti. Nakon što je taj korak uspješno proveden, slijedeći korak je obrada objekata na sceni. Prije obrade objekata obavlja se preliminarno filtriranje koje isključuje objekte koji nisu zanimljivi za daljnju analizu (na primjer, zbog pre male površine koje zauzimaju u kadru). Objekti koji su prošli filtraciju nastavljaju u iduću fazu, a to je generiranje morfološkog, imaginarnog kostura koji predstavlja objekt. Ukoliko se radi o čovjeku ili kakvom drugom čovjekolikom objektu, određivanje pozicija ključnih točaka koje smo spomenuli na kraju prošlog poglavlja bit će uspješno obavljeno. U konačnici detektirane objekte crtamo u kadru na lokaciji gdje se nalaze objekti u stvarnoj sceni. Ukoliko je kreiranje ljudskog kostura uspješno provedeno, na mjestu objekta bit će iscrtan čovječuljak s odgovarajućim položajima udova. U nastavku će biti opisan algoritam zajedno sa svim fazama.

2.1. Obrada kadra

Postoji mnogo načina na koje je moguće pristupiti ovom problemu. Neki su složeniji, a neki jednostavniji. Budući da je ovo samo jedna od faza algoritma koju je potrebno provesti kako bismo došli do konačnog rezultata, jednostavnosti radi odabrani algoritam je također relativno jednostavan. Video sekvencu koja dospjeva na ulaz možemo shvatiti kao niz kadrova, odnosno sličica koje pristižu tokom vremena. Kako bismo u svakom kadru razlučili pozadinu od objekata, primjenit ćemo jednostavnu metodu oduzimanja statične pozadine. U tu svrhu u pripremi moramo imati jedan kadar scene koji predstavlja pozadinu na kojoj nema objekata od interesa. Taj kadar bit će korišten kao pozadina. Između tog kadra i svakog kadra koji dospije na ulaz računa se aritmetička razlika svake točke na odgovarajućim pozicijama slike. Kao rezultat se dobiva monokromatska slika koja u svakoj točki sadrži vrijednost ovisno o razlici u boji točaka koje se nalaze u pozadinskom kadru i trenutnom. Ukoliko ta razlika prijeđe određeni prag, smatra se da je u toj točki slike jedna od točaka objekta.



Slika 2.2: Oduzimanje statične pozadine [5]

Postupak oduzimanje statične pozadine prikazan je na slici (Slika 2.2). Lijeva slika predstavlja statičnu pozadinu koja se ne mijenja za vrijeme rada algoritma. Desna slika predstavlja kadar iz video sekvence koji sadrži objekt. Slika neposredno ispod njih sadrži rezultat operacije oduzimanja te dvije slike i zovemo ju kadar razlike. Donja slika dobiva se provođenjem operacije praga (engl. threshold). Rezultat interpretiramo binarno. Gdje je razlika dovoljno velika, odnosno iznad određenog praga, točka je bijela i radi se o točki

objekta na sceni. Tamo gdje je razlika ispod praga, radi se o pozadini i točka je crna. Taj kadar ćemo nazvati kadar razlike.

Zbog načina na koji je ova faza izvedena ulazne videosekvence je potrebno ograničiti na one u kojima su kamera i pozadina statični. Iako to uvelika sužava mogućnost primjene ovog algoritma, ovaj pristup je usvojen jednostavnosti radi. U praksi postoje algoritmi koji modeliraju pozadinski kadar i mijenjaju ga tokom procesa obrade ukoliko detektiraju promjenu pozadine, no oni ovdje nisu razmatrani. Budući da je fokus na analizi objekata, odabrani pristup opravdan i stoga je ova faza pojednostavljena.

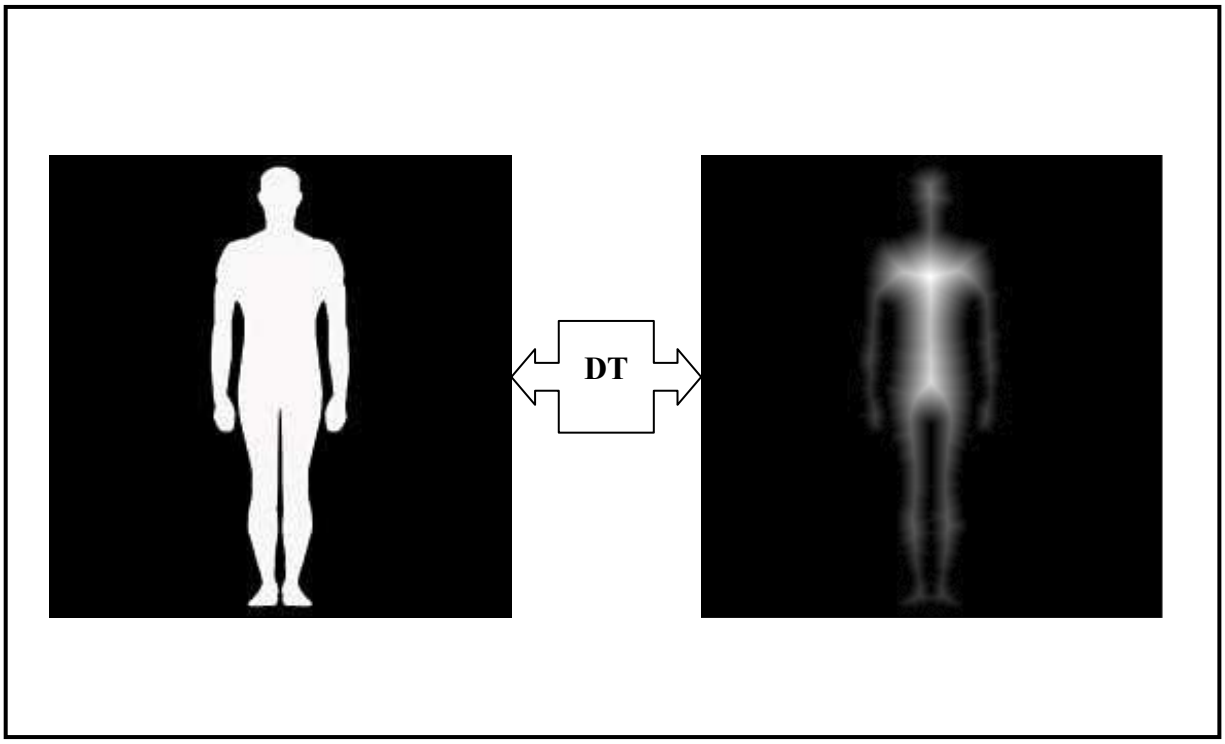
Nakon što je provedena analiza cijelog kadra, potrebno je razlučiti pojedine objekte u kadru. Detalji algoritma nisu dostupni budući da je ovaj zadatak prepušten metodama sadržanim u biblioteci koja je korištena u implementaciji programa. Kao rezultat ove faze algoritma dostupna je lista pravokutnika koji predstavljaju područja na slici gdje se nalaze objekti. Svaki pravokutnik je minimalni granični okvir (engl. minimal bounding box) pojedinog objekta na slici. U slijedeću fazu algoritma se šalju oni fragmenti kadra razlike koji su obuhvaćeni pravokutnicima.

2.2. Obrada objekata

Ulaz u ovu fazu algoritma je lista pravokutnika, odnosno fragmenata, koji su spomenuti na kraju prošlog potpoglavlja. Budući da je, ovisno o smetnjama koje su prisutne u video zapisu, moguće kao ulaz dobiti veliki broj elemenata koji nisu stvarni objekti, već smetnje, prije daljnje obrade provodi se proces filtriranja. Kroz filter se propuštaju samo oni pravokutnici čija je površina veća od proizvoljno odabrane granične.

Slijedeći korak je kreiranje zamišljenog morfološkog kostura objekta. Ovdje također postoji više mogućih pristupa. Neki od njih su primjena daljinske transformacije, primjena Voronoi-evih dijagrama i erozija [6]. Ovdje je korištena metoda daljinske transformacije (engl. distance transform).

Kao ulaz u ovu fazu pristižu siluete objekata, odnosno fragmenti kadra razlike koji su obuhvaćeni pripadajućim pravokutnicima.

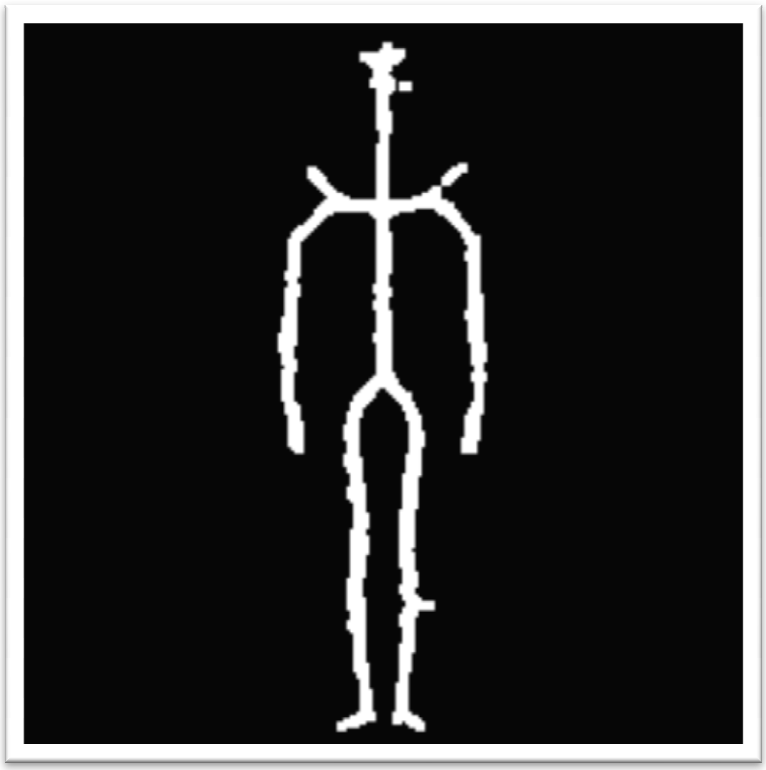


Slika 2.3: Primjena daljinske transformacije [7]

Primijenjena metoda prikazana je na slici (Slika 2.3). Od ulazne siluete kreira se nova slika koja sadrži samo tonove sive. Intenzitet svake točke odgovara njenoj udaljenosti do prvog ruba, odnosno do najbliže točke koja pripada pozadini (crna točka, intenzitet 0). Traženi kostur predstavlja konturu koja prati najmanju promjenu intenziteta, odnosno najmanji gradijent. To se postiže primjenom Laplaceovog operatora na sliku daljinske transformacije.

Slika 2.4: Primjena Laplaceovog operatora

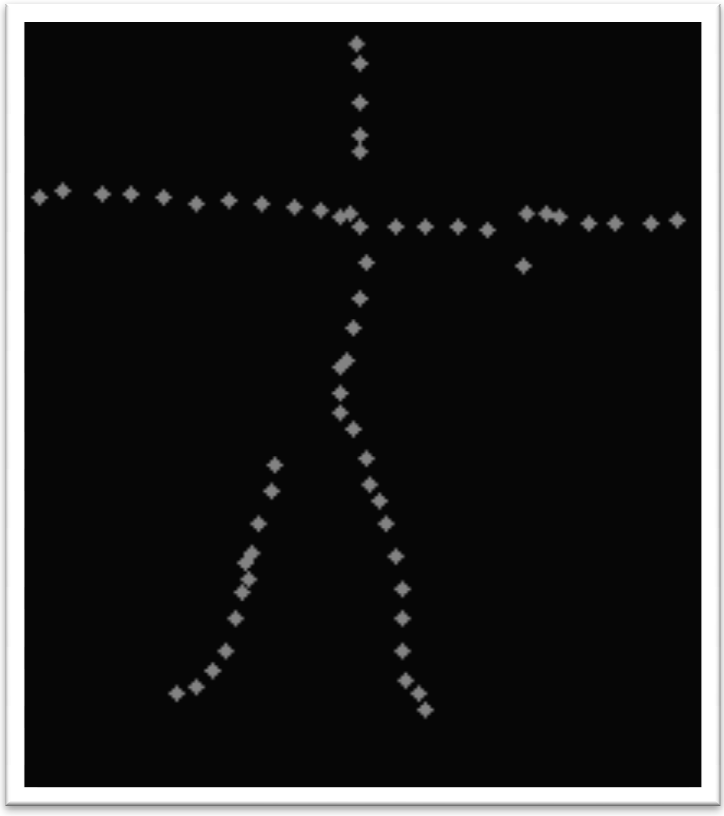
Slika 2.4 prikazuje rezultat primjene Laplaceovog operatora. Crna kontura odgovara kosturu objekta. Na taj rezultat primijenjena je operacija inverzije i praga kako bi u konačnici dobili kostur.



Slika 2.5: Dobiveni kostur

Dobivena slika sadrži mnogo točaka koje su dio kostura. Nakon što je kostur dobiven, primijenjena je morfološka operacija otvaranja koja se sastoji od erozije i dilatacije, kako bi određena doza šuma, ukoliko postoji, bila uklonjena.

Za daljnju obradu broj točaka je potrebno smanjiti. U tu svrhu određen je proizvoljni parametar *w* koji određuje dimenzije kvadratičnog prozora. Sliku prvo podijelimo na komadiće veličine tog prozora. Za svaki takav komadić generiramo jednu točku čije koordinate odgovaraju aritmetičkoj sredini koordinata svih točaka unutar tog prozora koje su bijele. Ukoliko unutar prozora nema bijelih točaka, ne generiramo točku. Na taj način drastično smanjujemo broj točaka kostura. Pritom treba paziti na veličinu parametra *w*. Ukoliko je pre velik, dobit ćemo pre grub kostur, a ukoliko je pre mali, nepotrebno ćemo povećati vrijeme izvođenja algoritma.



Slika 2.6: Generirane točke

Slika 2.6 prikazuje smanjeni broj točaka kostura, odnosno točke koje su generirane prilikom obrade točaka kostura. Ono što preostaje je, koristeći generirane točke u prošlom koraku, odrediti ključne točke kostura, odnosno poziciju glave, ruku i sl.

2.3. Određivanje ključnih točaka

U konačnoj fazi algoritma određujemo ključne točke kostura za svaki objekt čija je obrada uspješno stigla do ove točke. Ulaz u ovu fazu je skup točaka generiran u prethodnom koraku. Svaku točku karakteriziraju njena pozicija i vrijednost *dt* (engl. distance transform). U nastavku ćemo opisati postupak određivanja ključnih točaka.

Budući da je u ovoj fazi broj točaka s kojim rukujemo relativno malen, provodimo iscrpno pretraživanje. Za svaku točku kostura i za svaku ključnu točku generiramo uređenu trojku (*t*,*b*,*p*), gdje je *t* točka koju promatramo, *b* je oznaka ključne točke koju želimo dodijeliti točki *t* (glava, vrat, prsa, lijevo rame, desno rame, lijevi lakat, desni lakat, lijeva šaka, desna šaka, kralježnica, zdjelica, lijevi kuk, desni kuk, lijevo koljeno, desno koljeno, lijevo stopalo, desno stopalo), a *p* je vjerojatnost da je točka *t* upravo ključna točka *b*. Konačno, za svaku ključnu točku kostura uzimamo onu točku *t* iz uređene trojke koja ima najveću vjerojatnost *p*.

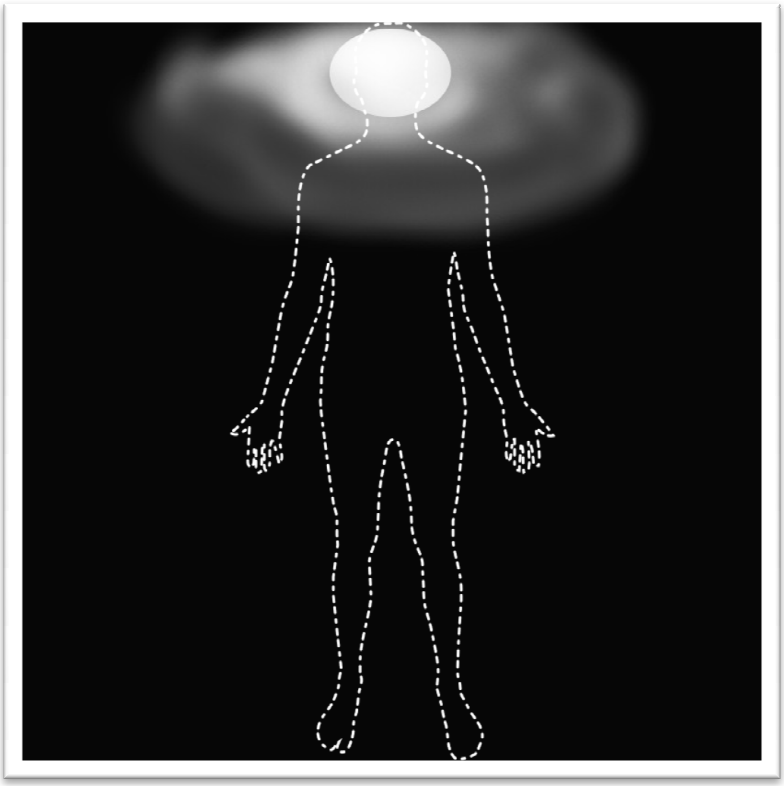
Promotrimo sada postupak korak po korak. Kao ulaz u ovu fazu algoritma pristižu uređeni parovi (*t*, *dt*), gdje je *t* točka generirana u prethodnom koraku, a *dt* je pripadajuća vrijednost daljinske transformacije. Postoji osamnaest oznaka ključnih točaka koje možemo pridjeliti ulaznim točkama, sedamnaest već nabrojanih i jedna dodatna oznaka: šum. Uzimamo u obzir mogućnost da ulazna točka nije niti jedna od ključnih točaka kostura; takvu točku klasificiramo kao šum. Za sve točke *t* razmatramo kolika je vjerojatnost da je upravo ta točka ključna točka *b*. Drugim riječima, generiramo kartezijev produkt skupa točaka *T* i skupa oznaka *B* i svakom elementu kartezijevog produkta pridjeljujemo vjerojatnost *p*, vjerojatnost da je točka *t* upravo ključna točka *b*. U konačnici oznakom *b* označavamo onu točku koja ima najveću vjerojatnost da je upravo ta točka ključna točka *b*. Način određivanja tih vjerojatnosti opisat ćemo u nastavku.

Postupak određivanja vjerojatnosti *p* povezan je s Bayesovim teoremom kako je opisano u članku koji nosi naslov „Silhouette-based probabilistic 2d human motion estimation for real-time applications“ (Oinam Binarani Devi) [9].

$$P(\omega_\alpha|t) \cong p(t|\omega_\alpha) \cdot p'(t|t,\omega_\alpha) \cdot P(\omega_\alpha) \tag{1}$$

Vjerojatnost da točka *t* nosi oznaku ω_α računa se pomoću formule (1) koja predstavlja adaptaciju Bayesovog teorema na zadatak kojim se bavimo. Prvi član, $p(t|\omega_\alpha)$, predstavlja apriornu vjerojatnost da točka *t* pripada klasi, odnosno nosi oznaku, ω_α. Te se vjerojatnosti

dobivaju iz vjerojatnosne mape za svaku pojedinu klasu. Statistički gledano, postoji izvjesna vjerojatnost da se određena ključna točka, na primjer glava, nađe u određenom dijelu radnog okvira, odnosno pravokutnika koji omeđuje snimljeni objekt.



Slika 2.7: Vjerojatnosna mapa za ključnu točku glava

Slika 2.7 prikazuje jednu takvu vjerojatnosnu mapu za ključnu točku glava. Iscrtkana ljudska figura nalazi se na slici samo u svrhu pojašnjenja postupka. Pretpostavimo da je na sceni snimljen čovjek kako hoda. Algoritam detektira objekt na sceni, kreira radni okvir, odnosno minimalni granični okvir oko objekta koji obuhvaća njegovu siluetu i šalje ga na dalnju obradu. Od siluete su generirane točke kostura kako je bilo ranije opisano i u ovom trenutku ispitujemo koja je vjerojatnost da je pojedina točka kostura upravo točka glave. U tu svrhu koristimo priloženu vjerojatnosnu mapu. Siluetu objekta preklopimo s vjerojatnosnom mapom. Potom vjerojatnost p da pojedina točka kostura upravo točka glave odgovara intenzitetu odgovarajuće točke na vjerojatnosnoj mapi. Iz vjerojatnosne mape možemo vidjeti da će se glava najvjerojatnije nalaziti pri vrhu radnog okvira, no moguće je i da bude drugdje. Za svaki element skupa B postoji pripadajuća vjerojatnosna mapa i na taj način uzimamo u obzir apriorne vjerojatnosti pojavljivanja određene ključne točke u nekom predjelu radnog okvira.

Slijedeći član formule, $p'(t|t,\omega_\alpha)$, uzima u obzir vremensku povezanost niza kadrova. Budući da su kretnje koje promatramo na sceni kontinuirane, postoji izvjesna vjerojatnost da će dana ključna točka kostura u ovom kadru biti u blizini mjesta na kojem se nalazila u prošlom kadru. Tu činjenicu koristimo kako bi stabilizirali detekciju kostura i učinili ju otpornijom na šum.

$$p'(t|t,\omega_\alpha) = e^{-\frac{d(t,t_{\omega\alpha})}{r}} \tag{2}$$

Pomoću formule (2) izračunavamo vjerojatnost da trenutna točka *t* nosi oznaku ω_α. $d(t,t_{\omega\alpha})$ predstavlja udaljenost između trenutne točke *t* i točke $t_{\omega\alpha}$ koja je u prošlom kadru bila klasificirana oznakom ω_α. Parametar r je duljina dijagonale radnog okvira pomnožena s proizvoljnim parametrom. Taj parametar određuje koliki utjecaj vremenska komponenta gibanja ima na detekciju ključnih točaka. Što je parametar bliže nuli, to je će se ključne točke u trenutnom kadru više „lijepiti“ za lokacije na kojima su se nalazile u prethodnom kadru. Navedeni parametar može se podešavati u sučelju aplikacije i time pratiti njegov utjecaj na obradu.

Zadnji faktor, $P(\omega_\alpha)$, predstavlja vjerojatnost detekcije pojedine klase ključnih točaka. Glava i stopala su primjeri ključnih točaka koje će vjerojatnije biti ispravno detektirane, budući da postoji manja šansa da budu prekrivene drugim dijelovima tijela, dok za ruke, recimo, postoji izvjesna šansa da budu djelomično zaklonjene i time nevidljive na kadru.

Nakon što su sve točke procesuirane na navedeni način, pojedina oznaka *b* pridaje se onoj točki koja ima najveću vjerojatnost da poprimi tu oznaku.

2.4. Dodavanje virtualnih objekata u scenu

Nakon što je provedena analiza objekata na stvarnoj sceni, moguće je u nju dodati virtualne objekte. U slučaju da je faza određivanja ključnih točaka uspješno provedena, na danoj poziciji bit će iscrtan čovječuljak koristeći informacije dostupne iz kostura. U slučaju da skeletizacija nije uspješno provedena, bit će iscrtan generički objekt koji ugrubo prekriva prostor na kojem se nalazi stvarni objekt na sceni. Dodavanje objekata provodi se jednostavnim crtanjem po zaslonu preko video sekvence

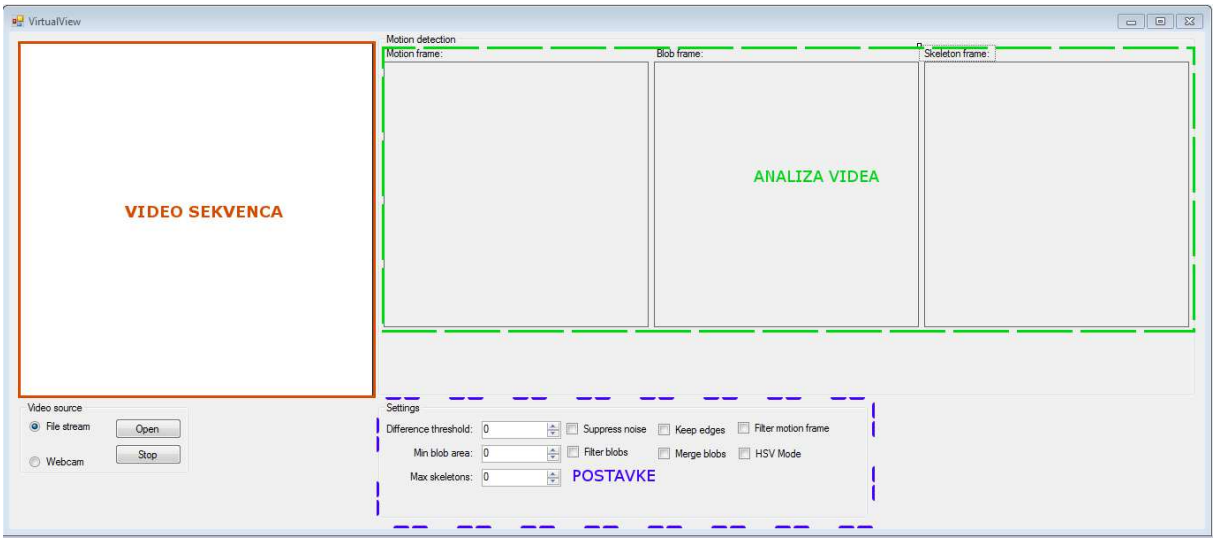
3. Aplikacija VirtualView

U sklopu ovog rada razvijena je aplikacija VirtualView koja implementira navedeni algoritam i provodi analizu video sekvence. Izrađena je u okruženju Microsoft Visual Studio 2012 koristeći programski jezik C# i .NET framework.

3.1. Korištena tehnologija

Aplikacija je izrađena koristeći Microsoft Visual Studio 2012 i napisana je u jeziku C#. Temeljena je na Microsoft .NET framework-u 4.5. U izradi su također korištene biblioteke EmguCv i AForge.NET. EmguCv je biblioteka koja pruža OpenCV sučelje prema .NET platformi i korištena pri obradi slikovnih podataka. AForge.NET je biblioteka koja sadrži niz mogućnosti kao što su obrada slike, videa, podrška za strojno učenje, neuronske mreže i sl. Također je korištena prilikom obrade slikovnih podataka i učitavanja video sekvence. Grafičko sučelje je ostvareno u standardnom Windows Forms okruženju koje je dio .NET platforme.

3.2. Izgled aplikacije



Slika 3.1: Izgled aplikacije VirtualView

Izgled aplikacije prikazan je na slici (Slika 3.1). Video sekvenca koja se trenutno obrađuje pušta se u najvećem prozoru s lijeve strane (prikazano crvenom bojom). S desne strane, u

području koje je označeno zelenom bojom, prikazane su neke od ključnih točaka u analizi svakog kadra, a to su:

- Motion frame: slika kadra razlike, područja koja zauzimaju objekti
- Blob frame: područja, odnosno objekti koji su odabrani za daljnju analizu
- Skeleton frame: prikaz dobivenog kostura

U području koje je označeno plavom bojom nalaze se postavke programa. Moguće je podesiti parametre praga prilikom oduzimanja pozadine, filtriranja pravokutnika prije obrade, maksimalan broj objekata na slici i neke parametre povezane s bibliotekom koja je korištena (AForge). Također je pružena mogućnost pretvorbe ulaznog videa iz RGB u HSV (hue, saturation, value) prostor boja, budući da u određenim slučajevima pruža bolju osnovu za oduzimanje pozadine. S malom promjenom boje točke u RGB sustavu obično se mijenjaju sve tri komponente, dok to nije slučaj u HSV sustavu [8].

3.3. Programska implementacija

Najveći dio funkcionalnosti koji nije dio standardnog .NET frameworka implementiran je korištenjem biblioteke AForge. Kontrola grafičkog sučelja koja prikazuje video sekvencu sadržana je u njoj. Također, pripadajuće klase koje služe za učitavanje videa i rukovanje s kadrovima, odnosno slikama, također pripadaju toj biblioteci. AForge također sadrži niz slikovnih filtera koji su korišteni prilikom obrade. Operatori matematičke morfologija kao što su dilatacija, erozija i otvaranje također su dio ove biblioteke. Ove su operacije korištene pri obradi i stabilizaciji rezultata. Oduzimanje pozadine također je implementirano koristeći klase *MotionDetector* i *CustomFrameDifferenceDetector* koje pripadaju biblioteci AForge. Operacija daljinske transformacije i Laplaceov operator dio su biblioteke EmguCv.

```
void videoSourcePlayer_NewFrame(object sender, ref Bitmap image)
        {
            if(config.hsvMode)
                rgbhsv.ApplyInPlace(image);

            // process motion
            motionDetector.ProcessFrame(image);

            // get motion frame
            UnmanagedImage motionFrame = cfdd.MotionFrame;

            // process blobs and update display
```

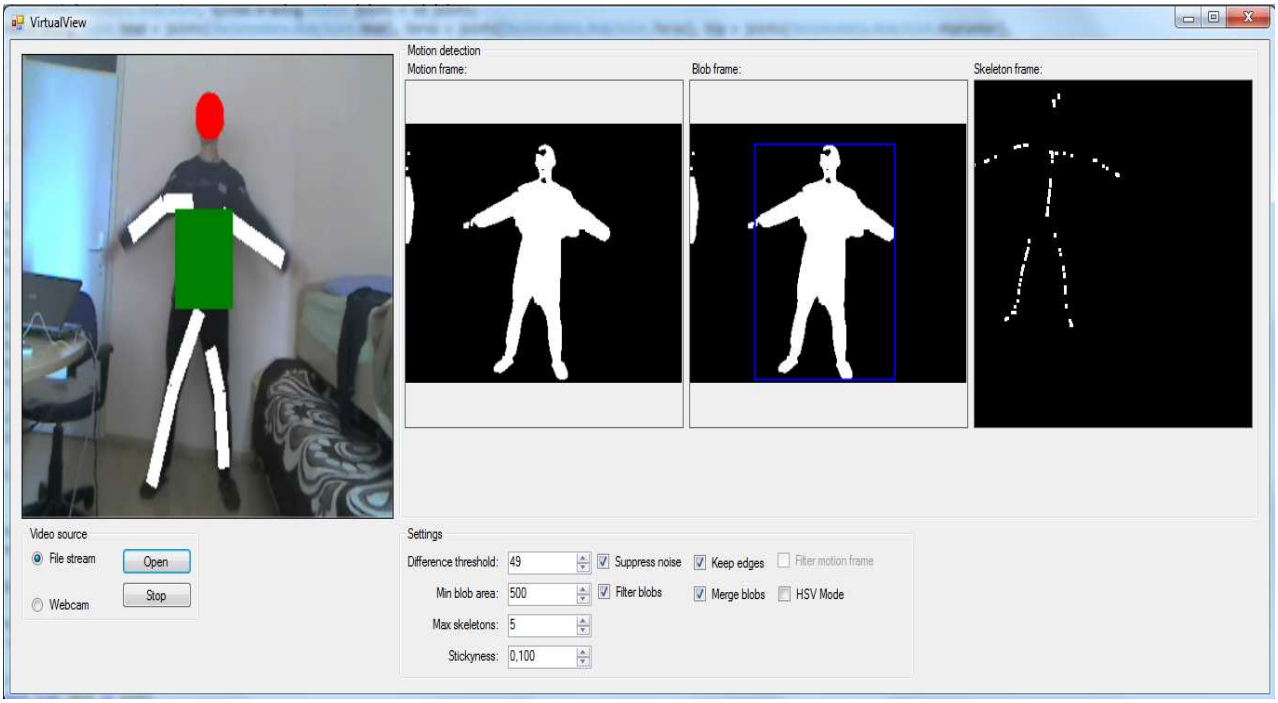
```
            if (motionFrame != null)
            {
                // apply filter?
                if(config.filterMotionFrame)
                    csm.ApplyInPlace(motionFrame);

                motionFrameView.Image =
motionFrame.ToManagedImage();

                // generate blobs
                blobCounter.ProcessImage(motionFrame);
                blobList =
blobCounter.GetObjectsInformation().ToList();

                // merge overlapping blobs
                //List<Rectangle> rList =
Utility.MergeBlobs(blobList, config.MergeBlobs);
                List<Rectangle> rList =
Utility.GetCandidates(blobList, config.maxSkeletons,
(int)(config.MinBlobArea * 1.25));
                DisplayBlobs(rList);
                ProcessSkeletons2(rList, motionFrame.Clone());
                DrawObjects(image);
            }
        }
```

Prikazani dio izvornog koda predstavlja središnju metodu koja obrađuje video zapis. Prilikom otvaranja datoteke s video zapisom kreira se objekt tipa *FileVideoSource* koji obavlja čitanje videa iz datoteke. Podatke koje čita prosljeđuje klasi *VideoSourcePlayer*, koja je ujedno i kontrola u grafičkom sučelju koja projicira video. Metoda *videoSourcePlayer_NewFrame* poziva se svaki put kada pristigne nova sličica videa i u njenom kodu možemo pratiti tijek procesa obrade. Na početku, ukoliko je potrebno, obavlja se pretvorba iz RGB sustava u HSV. Potom se pristigla sličica prosljeđuje klasi *MotionDetector* koja provodi postupak oduzimanja statične pozadine. MotionFrame je slika koja sadrži spominjani kadar razlike (8), odnosno područja na kojima se nalaze objekti. Ukoliko je bilo pomaka na slici, pristupa se daljnjoj obradi. Ako je uključeno dodatno filtriranje radi smanjenja šuma, kadar razlike se filtrira primjenom *ConservativeSmoothing* filtera biblioteke AForge. Klasa *BlobCounter* provodi detekciju objekata na kadru razlike i vraća listu takozvanih „blobova“, odnosno područja koja sadrže objekte na slici. Lista objekata se filtrira i oni čija je površina veća od praga prolaze dalje. U konačnici slijedi prikaz „blob“ objekata, odnosno pravokutnika ocrtavaju poziciju objekta, kreiranje kostura tih objekata i crtanje virtualnih objekata preko stvarnih.

```
public bool Process2()
        {
            try
            {
                // image processing
                DT();
                GenerateSkeletonImage();

                // skeleton processing
                GenerateSkeletonPoints3();

                // build clusters
                BuildClusters();

                // infer joints
                InferJoints2();
            }
            catch (Exception e)
            {
                return false;
            }

            ok = true;

            return true;
        }
```

Ovaj odsječak izvornog koda prikazuje proces obrade objekata. Za svaki objekt na sceni koji prođe filtriranje kreira se objekt klase *SkeletonData* koji služi za kreiranje kostura. Tom objektu se prosljeđuje silueta objekta od koje se kreira kostur. Iz izvornog koda možemo vidjeti kako teče proces. Prvo kreiramo takozvani *DistanceTransform* siluete, zatim kreiramo sliku kostura kako je navedeno u poglavlju 2.2 (metoda *GenerateSkeletonImage*). Nakon toga slijedi generiranje točaka kostura koristeći prozorčić veličine *w*, podjela tih točaka u grupe na temelju vrijednosti njihove vrijednosti distance transform i određivanje pozicija ključnih točaka kao što su glava, šake i sl (metoda *InferJoints2*).

Budući da je naglasak rada na detekciji kostura, estetskom vidu nije pridan prevelik značaj. Zbog toga je crtanje čovječuljka i drugih objekata izvedeno koristeći jednostavne operacije crtanja linija i pravokutnika koje su dostupne u Windows Forms okruženju.



Slika 3.2: Crtanje kostura na sceni

Slika 3.2 prikazuje izgled aplikacije u trenutku obrade video zapisa. Video zapis je snimljen web kamerom i potom otvoren u aplikaciji. U krajnjem desnom prozoru nalazi se prikaz skupa točaka koje pripadaju kosturu. Nad tim točkama kostura je proveden postupak određivanja ključnih točaka opisan u potpoglavlju 2.3. Ogledni izgled kostura aplikacija crta preko prozora video sekvence koji se nalazi krajnje lijevo.

4. Analiza performansi

Budući da se radi o aplikaciji koja kao ulaz prihvaća vrlo ograničen skup video sekvenci koji zadovoljavaju određena ograničenja koja smo spomenuli ranije, u nastavku će, zbog nedostupnosti uzorka za testiranje, biti dana samo kvalitativna analiza performansi. U nastavku slijedi osvrt na dobivene rezultate po pojedinim fazama obrade. Gdje god je moguće, osvrnut ćemo se i na prednosti i nedostatke korištenih biblioteka za obradu.

Opisane faze procesa obrade mogu biti promatrane i pojedinačno, budući da se problematika kojom se bave pojavljuje i kao potproblem pri rješavanju drugih problema na području proširene stvarnosti, obrade videa i slično. Faze su opisane odvojeno kako bi se modularnost rješenja bila istaknuta.

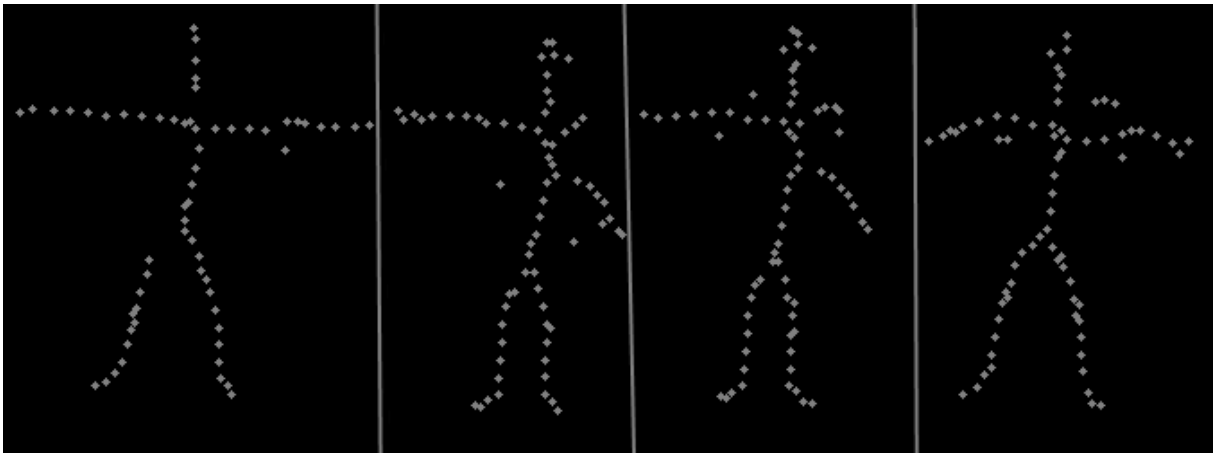
4.1. Obrada kadra, detekcija pozadine i objekata

Obradu kadra i detekciju pozadine u potpunosti obavljaju klase biblioteke AForge, što je i ranije spomenuto. Zbog nedostupnosti detaljnijeg opisa načina funkcioniranja tih komponenti bilo je teško utvrditi optimalan način njihovog korištenja. Sama detekcija objekata bazira se na ranije opisanoj metodi oduzimanja statične pozadine i provođenja operacije praga (engl. threshold) nad dobivenim kadrom. Budući da kvaliteta korištenih videa nije veoma visoka, u svim video sekvencama postoji dosta šuma. Taj šum se odražava na preciznost dobivenih silueti objekata. Što su objekti udaljeniji od kamere i što su njihove siluete manje, to šum više utječe na kvalitetu obrade. Trebamo naglasiti da je ovaj korak u obradi veoma važan jer uspješnost svih daljnjih operacija ovisi o kvaliteti rezultata ove faze. Nad svakim kadrom provode se operacije filtriranja kako bi rezultirajuće siluete bile što kvalitetnije i ujednačenije. Budući da primijenjena metoda promatra razliku u boji dviju točaka na kadrovima i to uzima kao kriterij detekcije objekta na sceni, moguće je da zbog sličnosti u boji pojedinih područja scene, promjene osvjetljenja na sceni, šuma i sličnih nepredvidivih faktora dođe do greške u procjeni. Iz tog razloga se u siluetama pojavljuju rupe, kao što je vidljivo na prethodnoj slici (Slika 3.2). Dimenzije objekta utječu i na dimenzije dobivenih silueti. Budući da je u svakom video zapisu prisutan šum, postoji stanovito ograničenje po pitanju minimalne veličine siluete koja može biti kvalitetno obrađena. Što su siluete manje, to šum više utječe na njihov

oblik. Iz provedenih pokusa utvrđeno je da aplikacija najbolje rukuje sa scenama u kojima objekti zauzimaju relativno velike površine na zaslonu. U suprotnom, šum na slici prevlada i generirana silueta ne odgovara stvarnom objektu. Također, algoritam raspoznavanja granica objekata koji generira spomenute „blobove“, odnosno pravokutnike, ne uspijeva zahvatiti cijelu siluetu zbog rupa na njoj. Ukoliko dođe do takvih grešaka, daljnji postupak obrade također pati, a konačni rezultat nije zadovoljavajući.

4.2. Obrada objekta

Prilikom obrade radnog okvira objekta, provođenja daljinske transformacije i generiranja točaka kostura problem stvaraju smetnje iz prethodnog koraka. Neujednačene siluete koje pristižu iz prethodne faze obrade, a sadrže šum u vidu „rupa“ i „plutajućih točaka“ otežavaju proces obrade i čine uniformiranje izlaznih rezultata gotovo nemogućim. Iz tog razloga je također bilo izazovno odabrati pogodan algoritam smanjenja broja točaka u kosturu koji se može nositi sa spomenutim šumom, a u isto vrijeme biti dovoljno jednostavan kako ne bi pretjerano opteretio proces obrade. Algoritam smanjenja broja točaka opisan u potpoglavlju 2.2 u određenim situacijama daje neodgovarajuće rezultate. Njegove performanse uvelike ovise o odabiru parametra veličine prozora. Pre mali prozor može produljiti vrijeme obrade do uočljivih razmjera, a pre velik prozor čini generirani kostur pre grubim za potrebe daljnje obrade. Pre velik prozor također može dovesti do toga da točke budu generirane na mjestima koje nisu dio kostura. To se događa u slučaju da prozor zahvati dva nepovezana dijela kostura, recimo dio lijeve noge i dio desne. U tom slučaju će točka biti generirana u praznom prostoru između nogu, što nije dio kostura objekta. Kao rješenje ovog problema razmotrena je mogućnost korištenja algoritma koji, umjesto stvaranja nove točke koja može, a i ne mora biti dio kostura, odabire stvarnu točku kostura kao element koji će generirati, no kriterij odabira točke nije bio uspješno definiran, pa je stoga odabran prvi algoritam.



Slika 4.1: Generirane točke kostura

Slika 4.1 prikazuje neke probleme prilikom generiranja točki kostura. Od četiri prikazana isječka tri sadrže točke koje smatramo šumom, budući da kvare strukturu kostura koji bi trebao biti minimalne debljine na svim dijelovima. Ovo je posljedica propagacije grešaka kroz faze algoritma. U početnoj fazi, prilikom detekcije objekata postoji greška prilikom oduzimanja pozadine. Nadalje, šum u videu utječe na rezultat obavljanja operacije praga koja razlučuje točke na kojima je došlo do promjene od onih na kojima nije došlo do promjene. Zbog svega navedenog algoritam kreiranja minimalnih graničnih okvira također povremeno generira neodgovarajuće okvire, a nakon svih tih operacija provodi se generiranje točaka kostura. Zbog kumulativnog efekta koji pritom nastaje, generirani morfološki kosturi su dosta neujednačeni.

4.3. Određivanje ključnih točaka

Jednostavnost izložene metode koja se temelji na vjerojatnosnoj analizi ima svoje prednosti i nedostatke. Iako je implementacija algoritma relativno jednostavna, uz dobro podešene parametre i kvalitetne ulazne podatke pruža solidne rezultate, pogotovo ako se uzme u obzir činjenica da aplikacija rukuje sa „sirovim“ video ulazom i nema pristup nikakvim dodatnim informacijama, kao što je na primjer dubina pojedine točke, odnosno udaljenost od kamere. Međutim, za povećanje kvalitete rezultata bilo bi potrebno dublje, po mogućnosti na većem uzorku, analizirati utjecaj pojedinih parametara na rezultat i izraditi kvalitetnije vjerojatnosne mape. Mape koje su korištene prilikom izrade rada nemaju podlogu u statističkoj analizi, već su kreirane proizvoljno kako bi aplikacija mogla biti testirana. Da su statistički podaci bili dostupni, zasigurno bi i rezultati bili bolji. Također je važno spomenuti da algoritam ne provodi nikakvu klasifikaciju objekata koje

analizira. Podaci svih objekata koji dospiju do ove faze obrade bit će tretirani kao podaci ljudskog kostura. Budući da algoritam ne uzima u obzir nikakve pragove prilikom dodjele pojedine oznake određenoj točki, već oznaku dodjeljuje točki koja ima najveću vjerojatnost poprimanja te oznake, moguće je da pojedina oznaka bude dodijeljena točki koja ima prilično malu vjerojatnost poprimanja te oznake, no jednostavno ne postoji bolji kandidat od nje. Ovo je jedna od značajki algoritma koja bi mogla biti dorađena.

4.4. Umetanje virtualnih likova

Umetanje virtualnih likova provodi se direktnim crtanjem po kontroli grafičkog sučelja koja prikazuje ulaznu video sekvencu. Budući da naglasak rada nije na estetic umetnutih likova, već na istraživanju procesa obrade video zapisa, umetnuti likovi sastoje se od debelih bijelih linija kojima se označavaju ruke i noge čovjekolikih objekata, zelenog pravokutnika koji prekriva torsto, te crvenog kruga koji prekriva glavu. Primjer umetnutog lika može se vidjeti na slici koja je navedena ranije (Slika 3.2). Budući da je operacija crtanja jednostavna, nema veliki utjecaj na performanse.

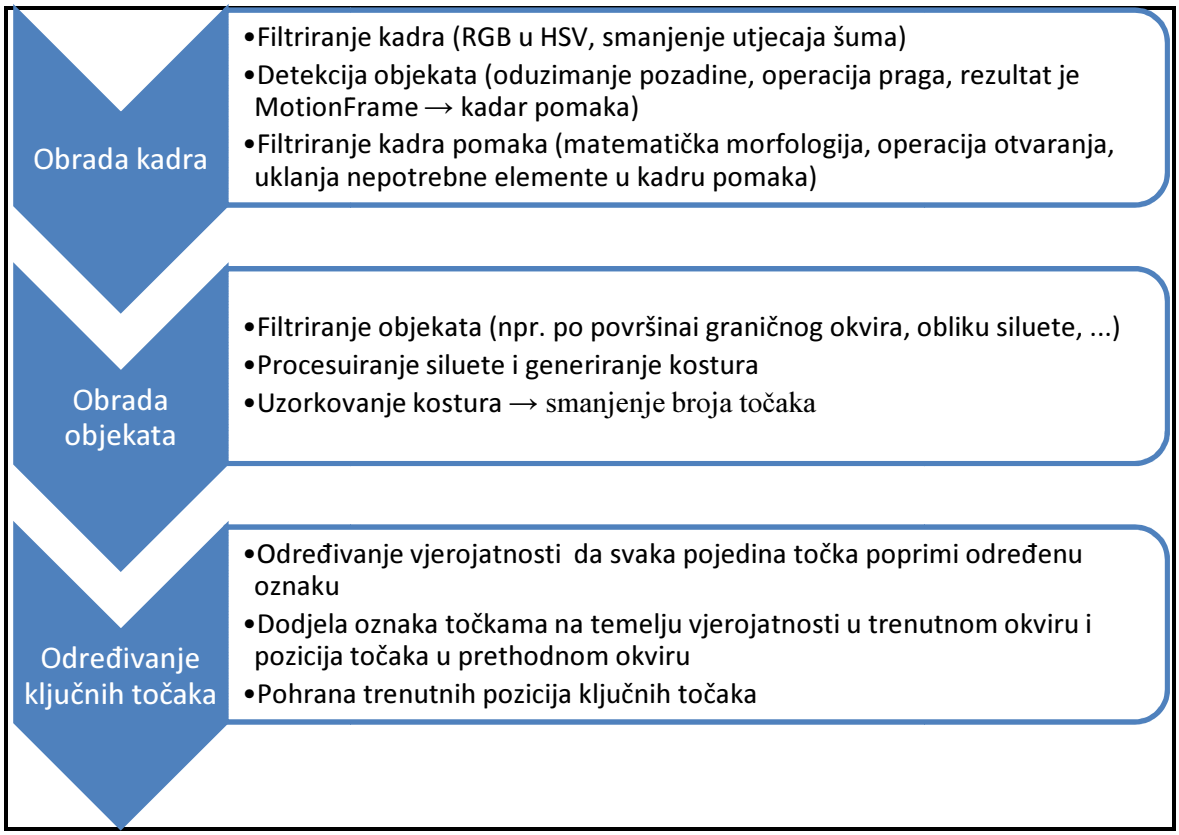
Aplikacija pruža mogućnost umetanja više likova istodobno, a maksimalni broj umetnutih likova može se regulirati koristeći korisničko sučelje programa. Iako razvijeno rješenje teoretski omogućava umetanje neograničenog broja likova na scenu, ograničenje ipak postoji. Budući da kvaliteta silueta objekata ovisi i o njihovoj relativnoj veličini u odnosu na veličinu kadra, što smo ranije spomenuli, obrada pre velikog broja objekata nema smisla.

4.5. Relevantnost rezultata

U nastavku ćemo dati kratku analizu relevantnosti dobivenih rezultata, odnosno njihovog značaja u okviru tema koje su povezane s proširenom stvarnošću. Implementirani algoritam obavlja zadatak generiranja morfološkog kostura čovjekolikih objekata na sceni, odnosno na ulaznoj video sekvenci. Ulazni podaci pritom su ograničeni na one video sekvence u kojima su kamera i pozadina statični. Iz navedenog možemo vidjeti da se radi o dosta specifičnom problemu i pripadajućem rješenju koje ima mogućnost obrade vrlo ograničenog skupa ulaznih podataka. Također, zbog opisanih manjkavosti algoritma, dobiveni rezultati se ne mogu mjeriti s već postignutima po pitanju kvalitete. Zbog

navedenih razloga možemo opravdano postaviti pitanje doprinosa koji opisani algoritam ima u domeni proširene stvarnosti.

Iako se postignuti rezultati ne mogu mjeriti s komercijalnim rješenjima, izloženi algoritam ipak ima potencijala, moguće ga je dalje nadograđivati, a i takav kakav jest pruža uvid u neke principe obrade scena koje sadrže pokretne objekte, primarno ljudske figure. U načelu, faze procesa obrade ulaza podudaraju se s onima koje su prisutne u već postojećim rješenjima sličnih problema. Na neke od tih radova smo se već referencirali. Ovaj rad, između ostalog, pruža generalni pregled procesa obrade ljudskih pokreta na sceni. Budući da postoje brojne sličnosti između mnogih postojećih rješenja ovakvih problema, predstavljeni algoritam pruža uvid u generalni tijek obrade podataka prilikom njihovog rješavanja. U tom smislu može poslužiti kao referenca.



Slika 4.2: Proces obrade scene

Slika 4.2 prikazuje proces obrade scene, svi ključni koraci su navedeni. Operacije filtriranja nisu obavezne i nisu strogo zadane. Njihova primjena ovisi o svojstvima ulaznih podataka. Kao što je već spomenuto, postupak je sličan i u ostalim rješenjima koja su

referencirana u radu, iz čega možemo zaključiti da, iako se radi o vrlo specifičnom rješenju, ono ima i određenu općenitu važnost koja je povezana s problematikom analize video scene, detekcije objekata i generiranja kostura. Te teme su aktualne i kao zasebne cjeline i kao dijelovi većih problema.

Zaključak

U ovom je radu istražena tema integracije virtualnih dinamičkih objekata i stvarne scene. Problem je razrađen na primjeru analize video sekvence i umetanje virtualnih objekata u nju, s naglaskom na kreiranje morfološkog kostura čovjekolikih objekata. Nakon analize dobivenih rezultata utvrđene su neke značajke razvijenog programskog produkta i izložene su u poglavlju 4.

Implementirano rješenje je relativno jednostavno i, s obzirom na ograničenost ulaznih podataka, solidno obavlja zadaću. Jednostavnost i općenitost algoritma za detekciju ključnih točaka kostura ostavlja prostor za daljnju analizu i omogućava nadogradnju. Demonstrirani algoritam, uz odgovarajuće statističke podatke, zasigurno ima potencijala.

Literatura

[1] Skupina autora, Augmented reality, https://en.wikipedia.org/wiki/Augmented_reality, 26.06.2015.

[2] https://upload.wikimedia.org/wikipedia/commons/thumb/7/7b/MediatedReality_on_iPhone2009_07_13_21_33_39.jpg/450px-MediatedReality_on_iPhone2009_07_13_21_33_39.jpg, 26.06.2015.

[3] http://www.faps.com/CLIPS_Security_Camera_DVR_H.264.htm, 12.05.2015.

[4] http://www.codeproject.com/KB/dotnet/KinectGettingStarted/8.png, 26.06.2015.

[5] Andrew Kirillov, Hands Gesture Recognition, http://www.aforgenet.com/articles/hands_gesture_recognition/, 19.04.2015.

[6] Kálmán Palágyi, Skeletonization, http://www.inf.u-szeged.hu/~palagyi/skel/skel.html#Introduction, 19.05.2015.

[7] http://www.freevectors.net/files/large/FreeVectorHumanSilhouette.jpg, 04.06.2015.

[8] Oinam Binarani Devi, Nissi S Paul, Y Jayanta Singh, Robust statistical approach for extraction of moving human silhouettes from videos, http://airccse.org/journal/ijit/papers/3314ijit06.pdf, 25.05.2015.

[9] Pedro Correa, Jacek Czyz, Toshiyuki Umeda, Ferran Marques, Xavier Marichal, Benoit Macq, Silhouette-based probabilistic 2d human motion estimation for real-time applications, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.69.5954&rep=rep1&type=pdf, 25.04.2015.

Sažetak

Integracija virtualnih dinamičkih likova i stvarne scene

U ovom radu obrađena je tema integracije virtualnih dinamičkih likova i stvarne scene. Dan je kratak uvod u problematiku detekcije objekata na stvarnoj sceni i njihove obrade. Problem je razrađen na primjeru analize video sekvence i umetanja virtualnih objekata u nju, s naglaskom na detekciju čovjekolikih objekata i određivanje njihovog morfološkog kostura.

Prezentirana je aplikacija VirtualView koja implementira opisani algoritam i analizirani su dobiveni rezultati.

Ključne riječi: proširena, stvarnost, kostur, detekcija, pokret, pozadina, oduzimanje

Abstract

Integration of virtual characters and real scenes

In this paper the topic of integration of dynamic virtual shapes and real scenes is analyzed. A short introduction to the problem of detecting objects in real scenes and their processing is given. The problem is elaborated upon given the example of analyzing a video sequence and inserting virtual objects into it, with the emphasis given to constructing the objects' morphological skeleton.

The application VirtualView, which solves the problem using the described algorithm, is presented and the obtained results are analyzed.

Keywords: augmented, reality, skeleton, detection, motion, background, removal

Uputa za instalaciju

Na priloženom CD-u se nalazi mapa „Instalacija“. Sadržaj tog direktorija potrebno je kopirati na proizvoljno mjesto na tvrdom. U toj mapi nalaze se i .dll datoteke biblioteka EmguCv i AForge.NET koje su potrebne za rad aplikacije. Također je potrebno preuzeti i instalirati Microsoft .NET Framework 4.5 ako već nije instaliran na računalu (http://www.microsoft.com/en-us/download/details.aspx?id=30653).

Nakon toga potrebno je pokrenuti „VirtualView“ dvostrukim klikom na ikonu. Program je namijenjen pokretanju na operacijskom sustavu Microsoft Windows 7 i Microsoft Windows 8.