SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3870

Automatska detekcija izvora svjetlosti u slici

Adriano Orioli

Zagreb, lipanj 2015.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 4. ožujka 2015.

ZAVRŠNI ZADATAK br. 3870

Pristupnik: **Adriano Orioli (0036463501)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Automatska detekcija izvora svjetlosti u slici**

Opis zadatka:

Detekcija izvora svjetlosti u slici važna je zbog mogućnosti reprodukcije virtualnih predmeta u istoj slici pod jednakim uvjetima osvjetljenja, što daje bolji vizualni dojam u primjenama proširene stvarnosti. Radi velikog interesa za takve primjene na pokretnim uređajima, pogotovo je zanimljivo detektirati izvor svjetla i prikazati osjenčane virtualne predmete u što kraćem vremenu, reda veličine 40 ms, kako bi se detekcija svjetla mogla koristiti u stvarnom vremenu koristeći samo kameru koja je ugrađena u pokretne uređaje.
Vaša je zadaća proučiti sustav proučiti metode za detekciju izvora svjetlosti u slici, te implementirati metodu za detektiranje izvora svjetlosti uz ograničenje na slike ljudskog lica te uz korištenje postojećeg sustava za praćenje lica.
Svu potrebnu literaturu i uvjete za rad osigurat će Vam Zavod za telekomunikacije.

Zadatak uručen pristupniku: 13. ožujka 2015.
Rok za predaju rada: 12. lipnja 2015.

Mentor:
Prof. dr. sc. Igor Sunday Pandžić

Djelovođa:
Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula:
Prof. dr. sc. Siniša Srbljić

SADRŽAJ

1. Uvod

Uz napredak tehnologije, sve veći broj korisnika je okružen uređajima koji imaju mogućnost fotografiranja i snimanja, te mogućnost obrade tih zapisa. Od mobilnih uređaja do stolnih računala, većina ih dolazi opremljena uređajima za snimanje slike i zvuka. Te mogućnosti se danas koriste, među ostalim, za marketinške, edukativne i zabavne svrhe u obliku aplikacija koje proširuju stvarnost koju korisnik vidi i umeću virtualne elemente poput grafičkih sučelja i 3D modela.

U ovom radu će se predložiti metoda pomoću koje možemo odrediti smjer sjene kako bi se 3D modeli pomoću kojih proširujemo stvarnost mogli što realnije prikazati. U tu svrhu će se prvo napraviti kratka analiza postojećih metoda detekcije sjena i određivanja njihovog smjera. Zbog kompleksnosti problema, rješenje će biti predstavljeno na teorijskoj bazi s crticama programskog rješenja. Također će se napraviti kratka analiza više metoda detektiranja izvora svjetla baziranih na dostupnosti informacija prije nego što se krene analizirati sama slika, poput oblika objekta koje baca sjenu, ili tipa materijala na kojem detektiramo osvjetljenje.

2. Određivanje sjena

2.1. Nepromjenjive slike

Nepromjenjive slike definiramo kao slike u kojima je eliminiran utjecaj svjetlosti na materijale, tj. kao slike koje prikazuju boju materijala uz minimiziran utjecaj sjena. Sve metode pretpostavljaju da su na slici prikazani materijali za koje vrijede Lambertova svojstva refleksije svjetlosti(Lambert).

2.1.1. Nepromjenjive slike pomoću kromatičnosti

Kromatičnost je specifikacija kvalitete boje neovisna o svjetlini. Klasično kromatičnost računamo kao

$$\{r, g, b\} = \frac{\{R, G, B\}}{R + G + B}$$

i time normaliziramo boju na vrijednost između 0 i $\frac{1}{3}$.

Koristeći metodu opisanu u (Drew et al., 2003) možemo generirati sliku u sivim tonovima koja će nam pomoći u detektiranju sjena. Prvi korak u generiranju nepromjenjive slike je reduciranje slike iz trokanalanog $\{R, G, B\}$ prostora u dvokanalani logaritamski prostor. Definiramo $\chi = \{\frac{R}{G}, \frac{B}{G}\}$ kao dvodimenzionalni vektor kromatičnosti.

$$\chi' = \{\log(\frac{R}{G}), \log \frac{B}{G}\}$$

predstavlja logaritamsku vrijednost kromatičnosti piksela.

Kod 2.1: kromatičnost

```
fn chromacity(src: &DynamicImage) -> DynamicImage {

  let mut buf = image::ImageBuffer::new( src.width()
                                        , src.height());

  for (x, y, pixel) in imgbuf.enumerate_pixels_mut() {
    let src_pixel = src.get_pixel(x,y);
    let mut sum :f32 = 0.0;
    for i in 0..3 {
        sum += src_pixel[i] as f32;
    }

    let mut rgb = [128u8; 3];
    for i in 0..3{
        let v = ((src_pixel[i] as f32 / sum) * 255.0);
        rgb[i] = v as u8;
    }
    *pixel = image::Rgb(rgb);
  }
  image::ImageRgb8(imgbuf)
}
```

Kod 2.2: logaritamska R kromatičnost

```
fn image_rg(src :&DynamicImage) -> DynamicImage{

  let mut buf = image::ImageBuffer::new( src.width()
                                        , src.height());

  for (x, y, pixel) in imgbuf.enumerate_pixels_mut() {
    let src_pixel = src.get_pixel(x,y);

    let mut lum :f32 = src_pixel[0] as f32;
    if src_pixel[1] != 0 {
        lum /= geometric_mean( src_pixel[0]
                             , src_pixel[1]
                             , src_pixel[2]);
    } else {
        lum /= 255.0;
    }

    lum = lum.ln() * 255.0;
    *pixel = image::Luma([lum as u8]);
  }
  image::ImageLuma8(imgbuf)
}
```

Kod 2.3: logaritamska B kromatičnost

```
fn image_bg(src :&DynamicImage) -> DynamicImage{

  let mut buf = image::ImageBuffer::new( src.width()
                                        , src.height());

  for (x, y, pixel) in imgbuf.enumerate_pixels_mut() {
    let src_pixel = src.get_pixel(x,y);

    let mut lum :f32 = src_pixel[2] as f32;
    if src_pixel[1] != 0 {
        lum /= geometric_mean( src_pixel[0]
                             , src_pixel[1]
                             , src_pixel[2]);
    } else {
        lum /= 255.0;
    }

    lum = lum.ln() * 255.0;
    *pixel = image::Luma([lum as u8]);
  }
  image::ImageLuma8(imgbuf)
}
```

Sve vrijednosti χ' unutar slike tretiramo kao točke na raspršenom grafu, te aproksimiramo pravac u koji ih najbolje opisuje koristeći jednostavnu linearnu regresiju (Seltman, 2009). Potom odredimo pravac v za koji vrijedi $u \perp v$. Projiciranje vrijednosti piksela na smjer definiran s v daje skalarnu vrijednost I', gdje je $I' = \chi' \cdot v$, iz koje možemo dobiti vrijednost piksela u sivom tonu

$$I = \exp\left(I'\right)$$

koja je nepromjenjiva u odnosu na osvjetljenost.

Kod 2.4: linearna regresija s pomoćnim funkcijama

```
fn best_fit(points :&Vec<Point>) -> (f32, f32){
  let len = points.len() as f32;
  let (mean_x, mean_y) = average(points);
  let sum_x_squared = sum_x_squared(points);
  let sum_xy_squared = sum_xy_squared(points);
  let a = (sum_xy_squared / len - mean_x * mean_y)
        / (sum_x_squared / len - mean_x * mean_x);
  let b = a * mean_x - mean_y;
  (a, b)
}
fn average(points :&Vec<Point>) -> (f32, f32){
  let mut mean_x :f32 = 0.0;
  let mut mean_y :f32 = 0.0;
  for point in points.iter() {
    mean_x += point.x;
    mean_y += point.y;
  }
  (mean_x / points.len() as f32
  , mean_y / points.len() as f32)
}
fn sum_x_squared(points :&Vec<Point>) -> f32{
  let mut sum: f32 = 0.0;
  for point in points.iter(){
    sum += point.x * point.x;
  }
  sum
}
fn sum_xy_squared(points :&Vec<Point>) -> f32{
  let mut sum: f32 = 0.0;
  for point in points.iter(){
    sum += point.x * point.y;
  }
  sum
}
```

Kod 2.5: generiranje slike u sivom tonu

```
fn invariant_grayscale( src :&DynamicImage
                      , slope :f32
                      , offset :f32) -> DynamicImage {
  let mut buf = image::ImageBuffer::new( src.width()
                                       , src.height());

  let y1 = slope * 5.0 + offset;
  let y2 = slope * 6.0 + offset;
  let (vx, vy) = (6.0 - 5.0, y2 - y1);

  for (x, y, pixel) in buf.enumerate_pixels_mut() {

    let src_pixel = src.get_pixel(x,y);
    let gm = geometric_mean( src_pixel[0]
                           , src_pixel[1]
                           , src_pixel[2]);

    let (rg, bg) = ( (src_pixel[0] as f32 / gm).ln()
                   , (src_pixel[2] as f32 / gm).ln());

    let luma = (rg * vx + bg * vy).exp() * 255.0;
    *pixel = image::Luma([luma as u8]);
  }
  image::ImageLuma8(buf)
}
```

2.1.2. Poboljšanje nepromjenjivih slika minimizacijom entropije

Proces opisan u prošlom odjeljku može dati krive rezultate prilikom računanja pravca smjera kromatičnosti, za poboljšanje rezultata koristimo postupak opisan u (Finlayson et al., 2004). Prva promjena koju možemo napraviti kako bi se poboljšala preciznost izračuna je korištenje geometrijske sredine vrijednosti $\{R, G, B\}$ umjesto samo kanala G.

Kod 2.6: geometrijska sredina RGB kanala

```
if src_pixel[1] != 0 {
    lum /= geometric_mean( src_pixel[0]
                         , src_pixel[1]
                         , src_pixel[2]);
```

Nakon generiranje vrijednosti logaritamske kromatičnosti piksela krećemo u proces minimizacije entropije. Za raspon kuteva $\phi = 1..180$ računamo vrijednosti piksela projicirane na pravac definiran vektorom smjera pod kutem ϕ, po formuli

$$I = \chi_1 * \cos(\phi) + \chi_2 * \sin(\phi)$$

te zatim računamo entropiju generirane slike koristeći formulu

$$\eta = -\sum_i p_i(I) \log(p_i(I))$$

vrijednost kuta ϕ pri kojem je entropija η najmanja je traženi smjer pravca kromatičnosti i nepromjenjiva slika koju generira je najbolja.

Generiranje nepromjenjivih slika metodom minimizacije entropije ima problema u brzini izvršavanja zbog toga što se mora kroz svaki pixel proći više puta kako bi se izračunale sve vrijednosti entropije. Unatoč tome, ideju o korištenju geometrijske sredine umjesto jednog kanala se koristi u programskom rješenju zbog boljih rezultata koje vraća.

2.2. Rubovi sjena

Kako bi detektirali sjene moramo znati gdje one počinju i završavaju tj. moramo odrediti rubove sjena. Rubovi sjena se određuju korištenjem algoritama poput (Sobel i Feldman, 1968) ili (Canny, 1986). Prvi korak prilikom detektiranja rubova sjena je određivanje rubova na originalnoj slici koju želimo analizirati. Nakon što odredimo rubne piksele s originalne slike, generiramo nepromjenjivu sliku koristeći gore opisani postupak. Potom na nepromjenjivog slici pokrenemo algoritam detektiranja rubova. Ono što sada imamo su dva skupa rubova, jedan s i jedan bez sjena, koje možemo jednostavno oduzeti kako bi dobili konačan skup u kojem se nalaze samo rubovi sjena. Ovaj postupak detektiranja rubova sjena je opisan u (Fredembach i Finlayson).

3. Određivanje izvora svjetlosti

3.1. Bez modela

U (Johnson et al., 2008) je predložena metoda za detekcija izvora svjetlosti na licu koristeći točke sjena i njima pripadajuće točke na objektu koji tu sjenu baca, poput nosa osobe. Takva metoda je jednostavna no propada u slučaju da objekt koji baca sjenu nije moguće detektirati. Korištenje ove metode na stvarnim osoba putem kamere je moguće uz korištenje tehnologije koja unaprijed može odrediti poziciju nosa, očiju i drugih elementara lica koje mogu bacati sjenu. Druga metoda koja je predložena je određivanje obrisa najsvjetlijeg dijela siluete lica, računanje normala na površinu siluete , te koristeći (Li et al.) određivanje kuta pod kojim svijetlo upada na obris.

(Tuan et al., 2011) predlaže metodu u kojoj se slika ljudskog lica prvo pretvori u sive tonove, te se odredi histogram vrijednosti koje se potom tretira kao distribuciju vjerojatnosti. Srednja vrijednost i varijanca se odrede i koriste za otkrivanje minimalnih i maksimalnih vrijednosti funkcije. Kako bi otkrili i polusjene, a ne samo potpune sjene, kao graničnu vrijednost funkcija se uzima radna točka u kojoj graf funkcije prelazi iz linearnog u eksponencijalni dio. Micanjem očiju i usana iz slike koristeći njihove unaprijed određene lokacije može se dobiti relativno precizno određeno područje svjetla i sjene na ljudskom licu, koje se onda može kombinirati s rješenjem predloženim u (Johnson et al., 2008) za određivanje kuta upada svjetlosti.

3.2. Koristeći 3D model objekta

3.2.1. Nepoznate geometrije

Uz poznavanje područja unutar sjene u slici i određene pretpostavke što se tiče oblika objekta koje slika prikazuje moguće je odrediti smjer svjetla koje stvara sjenu, takvu metoda opisana je u (Panagopoulos et al., 2011).

3.2.2. Poznate geometrije

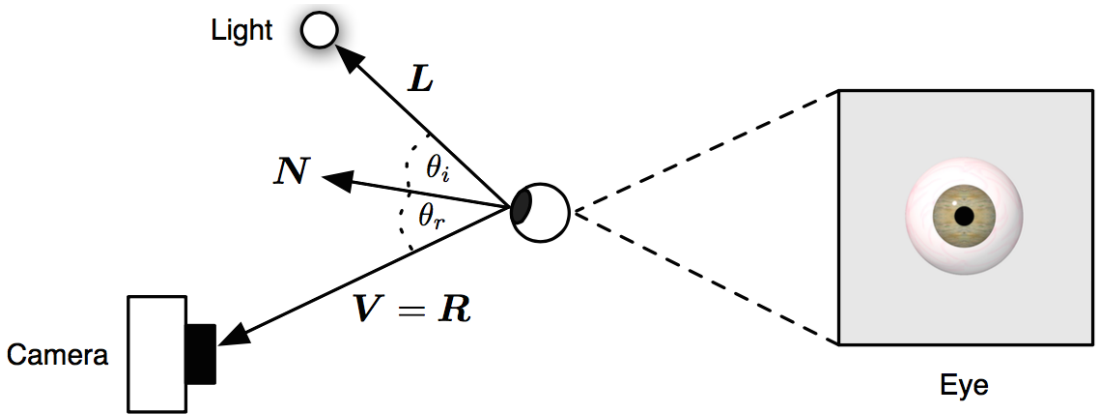
Za određivanje sjena na licu čovjeka kao prednost imamo upravo saznanje o poznatoj geometriji lica. Korištenjem unaprijed znanih modela možemo brže odrediti smjer upada svjetla. Detekcijom svjetla na općenitom objektu se bave (Liu et al., 2015). Prvi korak metode je određivanje pozicija sjene u stvarnoj slici. Zatim se model skalira i pozicionira tako da što preciznije opisuje objekt, te se pomoću generiranih mapa sjene određuje ona koja najbolje odgovara stvarnoj slici.

Koristeći općeniti oblik lica s konstatnom reflektivnošću kao poznati model moguće je primjeniti još jednu metodu koja je opisana u (Johnson et al., 2008). Prvo se određuje područje lica osobe i kut pod kojim je glava osobe pomaknuta u odnosu na kameru, potom se lice pozicionira na model kako bi se dobio teksturirani model iz kojeg je moguće odrediti normale i vrijednost teksture u sivim tonovima. Koristeći težinski prosjek vrijednosti piksela

$$\hat{s} = (\sum_{ij} n_{ij} n_{ij}^t)^{-1} \sum_{ij} L_{ij} n_{ij}$$

određuje se najbolji prosječni kut upada svjetlosti.

Kao bržu metodu koristeći postojeće modele, u slučaju da unaprijed znamo elemente lica osobe poput očiju, je bolje koristiti onu predloženu u (Johnson i Farid, 2007). Koristeći spekularnost očiju kao vodilju, te činjenicu da je oko približno sferičnog oblika, moguće je odrediti kut upada svjetlosti pomoću spekularnog odsjaja na zjenici oka.



Slika 3.1: Pozicija spekularnog odsjaja je određena normalom na površinu N, relativnim smjerom izvora svjetlosti L i kamerom V. (Johnson i Farid, 2007)

4. Tehnologija

Programski Jezik

Kao programski jezik je odabran Rust (Mozzila), jezik namjenjen za sistemsko programiranje kojeg razvija Mozzila u suradnji s volonterima iz cijelog svijeta. Specifičnost Rusta koja ga čini podobnim za ovakav rad je modularnost koja olakšava uključivanje stranih biblioteka u kod i sigurnosne provjere na razini memorije i višedretvenosti prilikom kompajliranja koda. Kako Rust ne koristi *garbage-collector*, već se sam brine o čišćenju resursa, može ga se koristiti zajedno s C bibliotekama bez potrebe za pisanjem dodatnog koda koji usporava procesiranje.

Dodatne biblioteke

Kao biblioteka za procesiranje slika je korištena *rust-image* (Piston), koja omogućava otvaranje slika s čvrstog diska, dekompresiranje slika iz formata poput *PNG* ili *JPEG* i pristup pojedinačnim pikselima same slike.

Visage SDK

Kao postojeći sustav za detekciju elemenata lica se može koristiti *Visage SDK* koji je razvijen na Fakultetu Elektrotehnike i Računarstva.

5. Zaključak

Automatska detekcija izvora svjetlosti je bila i ostaje netrivijalan zadatak. Koristiti algoritme za detekciju na nepoznatim objektima trenutno nije moguće učiniti u realnom vremenu na svim uređajima, a pogotovo ne na mobilnim platformama koje imaju memorijska ograničenja, međutim koristeći poznate modele i specijalizirane algoritme poput spekularnosti očiju ili poznatu geometriju licu moguće je ubrzati proces i poboljšati dobivene rezultate do prihvatljivih granica unutar kojih korisnik ne primjećuje pogreške.

Zbog nedostatka resursa, nije bilo moguće implementirati predložene algoritama kako bi se empirijski dokazalo koji su brži ili bolji, no u slučaju nastavka rada na ovom području cilj bi bio napraviti modul u jeziku Rust ili C koja prima informacije o modelu glave, poziciji elemenata lica i teksturi samog lica kako bi nadopunio postojeće mogućnosti Visage SDK sustava.

PRIMJERI KODA

LITERATURA

J. Canny. A computational approach to edge detection. 1986. URL `http://dl.acm.org/citation.cfm?id=11275`.

Mark S. Drew, Graham D. Finlayson, i Steven D. Hordley. Recovery of chromaticity image free from shadows via illumination invariance. U *Workshop on Color and Photometric Methods in Computer Vision*, stranice 32–39, Nica, Francuska, 2003.

Graham D. Finlayson, Mark S. Drew, i Cheng Lu. Intrinsic images by entropy minimization. 2004.

Clément Fredembach i Graham D. Finlayson. Hamiltionian path based shadow removal.

Micah K. Johnson i Hany Farid. Exposing digital forgeries through specular highlights on the eye. U *9th International Workshop on Information Hiding*, Saint Malo, France, 2007.

Micah K. Johnson, David G. Stork, Soma Biswas, i Yasuo Furuichi. *Inferring illumination direction estimated from disparate sources in paintings: An investigation into Jan Vermeer's Girl with a pearl earring*. 2008.

Lambert. Lambertian reflectance. URL `http://en.wikipedia.org/wiki/Lambertian_reflectance`. Pristup: 2015-06-07.

Y. Li, S. Lin, H. Lu, i H.-Y. Shum. Multiple-cue illumination estimation in textured scenes. U *9th IEEE International Conference on Computer Vision (ICCV)*.

Yang Liu, Theo Gevers, i Xueqing Li. Estimation of sunlight direction using 3d object models. U *IEEE TRANSACTIONS ON IMAGE PROCESSING*. 2015.

Mozzila. The rust programming language. URL `https://doc.rust-lang.org/stable/book/`. Pristup: 2015-05-23.

Alexandros Panagopoulos, Chaohui Wang, Dimitris Samaras, i Nikos Paragios. Illumination estimation and cast shadow detection through a higher-order graphical model. U *IEEE Conference on Computer Vision and Pattern Recognition*, Colorado, USA, 2011.

Piston. rust-image documentation. URL `http://www.piston.rs/image/image/index.html`. Pristup: 2015-05-23.

Howard J. Seltman. *Experimental Design and Analysis*, poglavlje Simple Linear Regression. 2009. URL `http://www.stat.cmu.edu/~hseltman/309/Book/chapter9.pdf`.

I. Sobel i G. Feldman. A 3x3 isotropic gradient operator for image processing. 1968. URL `http://www.researchgate.net/publication/239398674_An_Isotropic_3_3_Image_Gradient_Operator`.

Tran Anh Tuan, Min Gyu Song, i lin Young Kim. A robust shadow and light region detection using within-class variance in face images. 2011.

Automatska detekcija izvora svjetlosti u slici

Sažetak

Uz napredak tehnologije, sve veći broj korisnika je okružen uređajima koji imaju mogućnost fotografiranja i snimanja, te mogućnost obrade tih zapisa. Od mobilnih uređaja do stolnih računala, većina ih dolazi opremljena uređajima za snimanje slike i zvuka. Te mogućnosti se danas koriste, među ostalim, za marketinške, edukativne i zabavne svrhe u obliku aplikacija koje proširuju stvarnost koju korisnik vidi i umeću virtualne elemente poput grafičkih sučelja i 3D modela. U ovom radu se analiziraju trenutačno dostupne metode razvijene u svrhu automatskog detektiranja svjetla na slici. Detektiranje svjetla, i shodno tome, kreiranje virtualnog osvjetljenja za elemente proširene stvarnosti bi povećalo kvalitetu aplikacija koje se predstavljaju korisnicima, i samim time i vrijednost koju oni sami dobivaju od njih.

Ključne riječi: svjetlo, svjetlina, osvjetljenost, sjena, automatska detekcija, rust

Automatic light source detection in image

Abstract

With the current advancements in technology, an ever increasing number of users is surrounded by devices with video and photo taking capabilities. From mobile devices to desktop computers, most come equipped with some sort of camera, and a processor powerful enough to process the images from that camera in realtime. Those capabilities are being used today, among other things, for marketing, educational and entertainment purposes in the form of applications for augmented reality which augment the perception of the users with graphical interfaces and 3D models. This paper will analyse the currently available algorithms for automatic light source detection in images. Light source detection, combined with the creation of 3D models illuminated by that light source, can serve a great purpose in the enhancement of augmented reality applications, the increase of user immersion, and consecutively the improvement of the user experience.

Keywords: light, luminance, illumination, shadow, automatic detection, rust