

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5092

**Praćenje igrača u snimkama
nogometnih utakmica**

Franko Šesto

Zagreb, lipanj 2017.

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

Zagreb, 8. ožujka 2017.

ZAVRŠNI ZADATAK br. 5092

Pristupnik: **Franko Šesto (0036485336)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Praćenje igrača u snimkama nogometnih utakmica**

Opis zadatka:

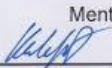
Analiza kretanja igrača tijekom utakmice pruža treneru i njegovom stručnom stožeru dragocjene informacije. Proces analize video-snimki utakmica može se olakšati i unaprijediti primjenom postupaka računalnog vida.

U okviru završnog rada treba proučiti pristupe za praćenje objekata opisane u literaturi te programski ostvariti sustav za praćenje igrača u video-snimkama nogometnih utakmica. Pripremiti bazu uzoraka za učenje i testiranje sustava. Analizirati ponašanje implementiranog sustava te prikazati i ocijeniti ostvarene rezultate.


Radu priložiti izvorni i izvršni kôd razvijenih postupaka, ispitne video-sekvence i rezultate, uz potrebna objašnjenja i dokumentaciju.

Zadatak uručen pristupniku: 10. ožujka 2017.
Rok za predaju rada: 9. lipnja 2017.

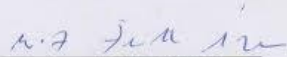
Mentor:


Izv. prof. dr. sc. Zoran Kalafatić

Djelovođa:


Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za
završni rad modula:


Prof. dr. sc. Siniša Srblić

SADRŽAJ

Sadržaj

1.Uvod	1
2.Mean-shift algoritam	2
3.Mean-shift - primjena	16
3.1 Usporedba histograma	20
3.2 Histogram backprojection	30
4.Implementacija i uporaba	35
5. Ostale metode praćenja objekata	38
6. Zaključak	41
7. Literatura	42
8. Naslov, sažetak i ključne riječi	43

1. Uvod

Danas je vrlo često, a i uobičajeno u nogometu, ali i u većini ostalih timskih sportova na velikim terenima pratiti igrače i dobivati na temelju tog praćenja mnoge korisne informacije koje mogu poslužiti treneru, ali i njegovom stručnom stožeru. Danas razni programi mogu izvesti statistike koje prije nisu bile ni moguće za izračun. Uvidi u te teško dohvatljive informacije pružaju nogometnim analitičarima podatke koji im omogućuju vrlo objektivno analizirati izvedbe igrača od početka do kraja utakmice. U te izvedbe spadaju brzina igrača, put koji je igrač prošao u određenom vremenskom intervalu, na kojem se dijelu terena kretao u ključnim trenucima itd. Trener može zajedno sa svojim timom, ali i stručnim stožerom na temelju tih informacija pokušati vidjeti koji su trenutni problemi unutar njegovog tima te to na neki način popraviti, ali isto tako može iskoristiti praćenje igrača u protivničkom timu. Može vidjeti kako se konkretni igrači iz protivničkog tima kreću, te na temelju toga izraditi konkretnu strategiju koja može pomoći dobitku same utakmice. Danas je mnogo izazova u automatskom analiziranju videa. Primjena postupaka računalnog vida olakšava proces analize video-snimaka utakmica.

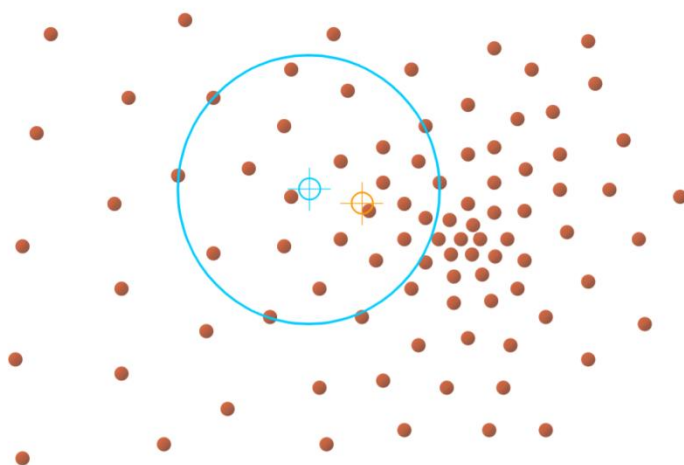
Kao osnovu obrade htio bih obraditi jedan vrlo rašireni algoritam za praćenje objekata općenito, a tu se radi o Mean-shift algoritmu. Njega sam odabrao kao algoritam koji namjeravam opisati matematički na jednostavnijem obliku dvodimenzionalnog prostora za samo razumijevanje, a zatim i u kontekstu slika videa. Namjeravam opisati dvije najpoznatije metode obrade slike prije samog korištenja Mean-shift algoritma -- jedna koja se gotovo stalno koristi uz opis Mean-shifta i druga koju ću koristiti u samoj implementaciji u Pythonu (version 2.7.9) koristeći biblioteku OpenCV (version 3.2.0) te koja je vrlo pristupačna i razumljiva korisnicima algoritma. Nakon opisa implementacije i kratkih uputa o korištenju, još ću spomenuti ukratko neke druge česte metode praćenja objekata općenito.

2. Mean-shift algoritam

Osnovni algoritam koji sam želio obraditi te koji smatram bazom ovog završnog rada je Mean-shift algoritam. Iza njega stoji vrlo jednostavna ideja, a vrlo složena matematika. Prije same primjene algoritma u praćenju nogometaša u videu nogometne utakmice objasnio bih samu teoriju Mean-shifta te kako on zapravo funkcionira općenito, a onda i u kontekstu samog praćenja objekata u videu.

Kada bih najkraće moguće opisao Mean-shift algoritam i rekao sve bitno o samom algoritmu rekao bih da je to neparametarski način (tehnika) analize svojstvenog prostora za lociranje maksimuma funkcije gustoće. Ova definicija možda na samom početku nije potpuno shvatljiva te treba dodatno obrazloženje o čemu se tu zapravo radi te ću zato objasniti što svaki dio definicije označava.

Dakle, analiziramo nekakav prostor tako da bismo locirali maksimum funkcije gustoće. Ne ulazeći još u to o kakvom prostoru je riječ, zamislimo da je naš prostor jedno dvodimenzionalno polje podatkovnih točaka (eng. data points) te svaka od njih ima određenu poziciju (x, y) . Takav jednostavan prikaz prostora obrađujem iz razloga da objasnim ostale pojmove bitne za funkcioniranje Mean-shifta.



Slika 1. Prostor podatkovnih točaka – R.Collins, Mean-shift Tracking, Spring 2006.

Izvor Slike 1. - <http://www.cse.psu.edu/~rtc12/CSE598G/introMeanShift.pdf>

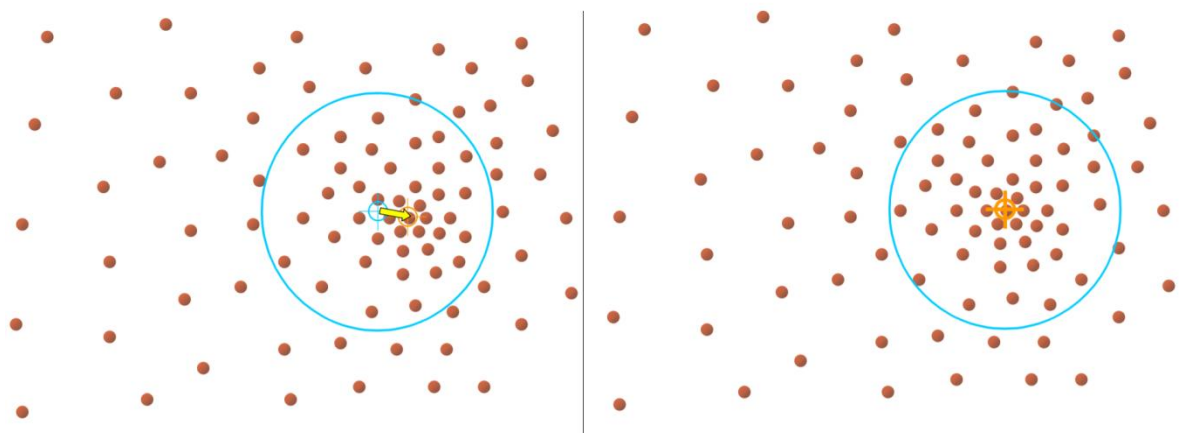
Na Slici 1. možemo vidjeti konačan broj točaka na određenim pozicijama. Cilj nam je naći najgušće područje ove distribucije. Iz toga slijedi da želimo naći područje s

najviše točaka odnosno područje gdje su točke najbliže jedna drugoj (udaljenosti su najmanje). Trebamo imati nekakav prostor (dio dvodimenzionalog polja) koji promatramo i gledamo gustoću samo unutar toga prostora. Taj prostor se naziva regija interesa (eng. region of interest, ROI). On je definiran sa svojim centrom i radijusom. Mi odlučujemo o radijusu ROI-a, a što se tiče mjesta gdje postavljamo njegov centar na samom početku to možemo postaviti slučajnim odabirom nad cijelim područjem skupa točaka. Na Slici 1. ROI je predstavljen plavom kružnicom, a njezin centar je dobiven slučajnom procjenom. Kako bismo došli do najgušćeg područja moramo pomicati centar regije interesa na način da izračunavamo centar mase svih točaka (centroid, eng. mean) unutar nje same. Na Slici 1. to je prikazano narančastim znakom unutar ROI-a. Dakle, slijed operacija koje bismo morali činiti u svakoj iteraciji (objašnjen na trivijalan način) je sljedeći:

Uzimamo svaku točku unutar regije interesa, dodamo u ukupnu sumu te podijelimo sa brojem točaka koji smo uzeli. Što smo ovime dobili?

Vrijednost centralne koordinate unutar regije interesa. Dakle, taj centar predstavlja prosjek svih x i y vrijednosti unutar regije interesa.

Sada imamo staru vrijednost centroida (na početku slučajna vrijednost) i novoizračunatu vrijednost. Te dvije točke čine takozvani Mean-shift vektor gdje je početna točka vektora stara vrijednost, a završna točka nova vrijednost. Pošto radijus regije interesa ne mijenjamo, cijelo vrijeme radimo njen pomak (regije) pomoću tog vektora. Ovaj postupak ponavljamo sve dok se više ne možemo kretati, odnosno u trenutku kada će početna točka biti jednaka završnoj što znači da Mean-shift vektor postaje nul-vektor. Zbog toga kažemo da konvergiramo prema najgušćem području.



Slika 2. Mean-shift vektor te pomak regije interesa – R. Collins, Mean-shift Tracking

Izvor Slike 2. - <http://www.cse.psu.edu/~rtc12/CSE598G/introMeanShift.pdf>

Na Slici 2. možemo vidjeti prikaz posljednje iteracije u kojoj tražimo centar mase. Na lijevoj strani slike žutom bojom je označen vektor od centra mase prošle regije interesa (plava boja) do novog centra mase (narančasta boja). Zbog toga na desnoj strani slike pomičemo i cijelu regiju interesa. Isto tako zaključujemo na desnoj strani slike da bi novi centroid bio na istom mjestu kao i prošli te možemo završiti proces jer smo konvergirali.

Napomena: U zadnjim iteracijama ovog procesa vektor će biti sve manji i manji jer smo sve bliži najgušćem području.

$$M_h(y) = \left[\frac{1}{n_x} \sum_{i=1}^{n_x} x_i \right] - y_0 \quad (1)$$

Formula (1) na trivijalan način objašnjava što radimo u svakoj iteraciji ovog algoritma. Uzimamo u obzir da je n_x broj točaka unutar ROI-a, a y_0 centar mase u prošoj iteraciji algoritma (osim u prvom slučaju kada je dobiven slučajnim odabirom). Unutar uglatih zagrada dobivamo vrijednost novog centra mase na način da zbrojimo sve točke te ih podijelimo s njihovim brojem. Na samom kraju oduzmemo početnu točku od završne te time dobivamo Mean-shift vektor. Algoritam postavlja $y_0 \leftarrow M_h$ te time možemo ići u sljedeću iteraciju jer imamo dovoljno podataka za nju.

Iz ovoga zaključujemo da Mean-shift vektor pokazuje prema smjeru maksimalnog porasta gustoće (ovo će se potkrijepiti i dokazom).

Nadalje, da bismo poboljšali formulu (uveli više informacija) uvodi se pojam težine (eng. weight). Svakoj točki unutar ROI-a ćemo pridružiti određenu težinu koja ovisi o udaljenosti od početne točke (y_0 , središte kružnice) do te točke kojoj pridajemo samu težinu.

$$M_h(y) = \left[\frac{\sum_{i=1}^{n_x} w_i(y_0) x_i}{\sum_{i=1}^{n_x} w_i(y_0) \cdot 1} \right] - y_0 \quad (2)$$

Jedina razlika od formule (1) koju ima formula (2) je što smo svakoj točki dodali koeficijent težine koju točka ima te umjesto broja točaka u ROI-u smo stavili zbroj svih težina svake točke pošto primjenjujemo težine u sumi svih točaka (radimo normalizaciju).

Sada napokon možemo primjetiti da naša formula ima sličnost sa formulom izračuna

centra mase nekog tijela.

$$r_c = \frac{1}{m} \sum_{i=1}^N m_i r_i \text{ gdje je } m = \sum_{i=1}^n m_i \quad (3)$$

Vidimo da su formule identične ako uzmemo u obzir da je težina naše točke isto što i masa pojedine čestice u nekom tijelu koje sadrži više čestica (m_i u formuli (3)), a oznaka za položaj naše čestice x_i je isto što i položaj tijela "i" (r_i u formuli (3)) unutar tijela.

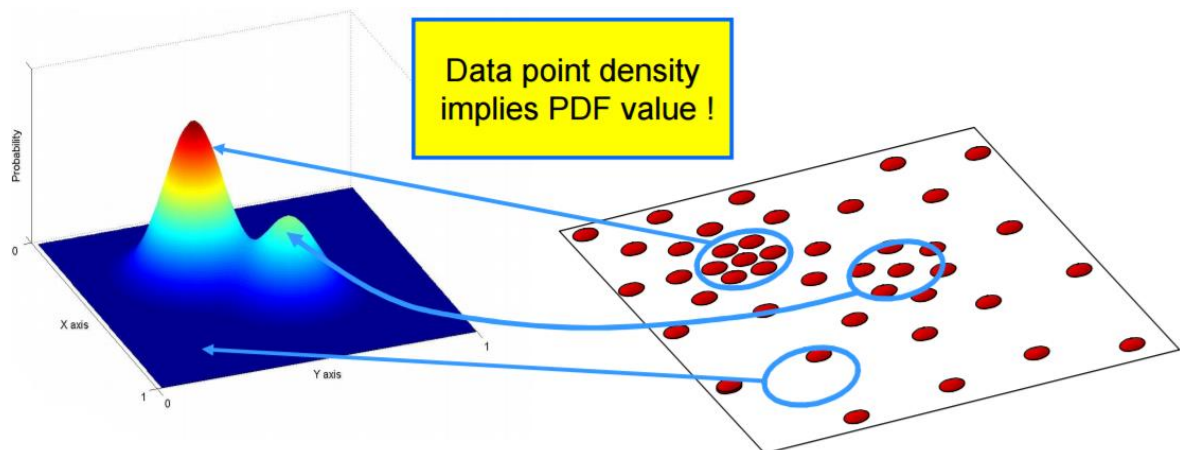
Možemo zaključiti da formula (2) u odnosu na formulu (1) je naprednija jer formula (1) praktički gleda kao da su sve težine svih točaka jednake i iznose 1, dok nam formula (2) daje puno više informacija o točkama pomoću težina izračunatih na neki način koji ovisi o udaljenosti od početne točke unutar ROI-a.

Sljedeće pitanje koje možemo postaviti je koji su to načini izračuna težina uzimajući u obzir ovisnost o udaljenosti?

U statistici se za to koriste takozvane Kernel (jezgrene) funkcije od kojih ću spomenuti najpoznatije: a) jednoliku funkciju, b) Gaussovu funkciju te c) Epanechnikovovu funkciju (najučinkovitiju). Kernel funkcije ću opisati kasnije jer je potreban kontekst uz koji su one vezane, a to je formuliranje distribucija našeg prostora podatkovnih točaka.

Ovime smo jednostavno pokazali i objasnili dio definicije Mean-shifta koji kaže da on predstavlja analizu prostora (za sada trivijalnog prostora podatkovnih točaka) za lociranje maksimuma funkcije gustoće. No, kako dobiti formulu te funkcije te da li ju je zapravo uopće moguće dobiti? Za sada smo sve to radili na prostoru točaka ne upuštajući se u sam izgled funkcije. Osnovni prostor podatkovnih točaka možemo vizualno prikazati osnovnom funkcijom gustoće vjerojatnosti. Način na koji prikazujemo funkciju gustoće vjerojatnosti (od sada nadalje koristim kraticu PDF, eng. probability density function) je taj da na trodimenzionalom modelu prikazujemo kolika je vjerojatnost da se na (x, y) poziciji nalazi određen broj točaka. Dakle, ako odaberemo slučajnu točku unutar XY ravnine u tom 3-D modelu možemo odrediti vjerojatnost. Na Slici 3. možemo primjetiti preslikavanje prostora stanja u distribuciju te kako vjerojatnost u 3-D modelu ovisi o gustoći podatkovnih točaka. Za izradu same distribucije se koriste takozvane "košare" (eng. **bins**). Svaka košara predstavlja pojedinu koordinatu (x, y) u prostoru te pri izradi distribucije uzimamo svaku točku te je stavljamo u košaru ovisno o koordinati na kojoj se nalazi. Zbog toga vrhovi

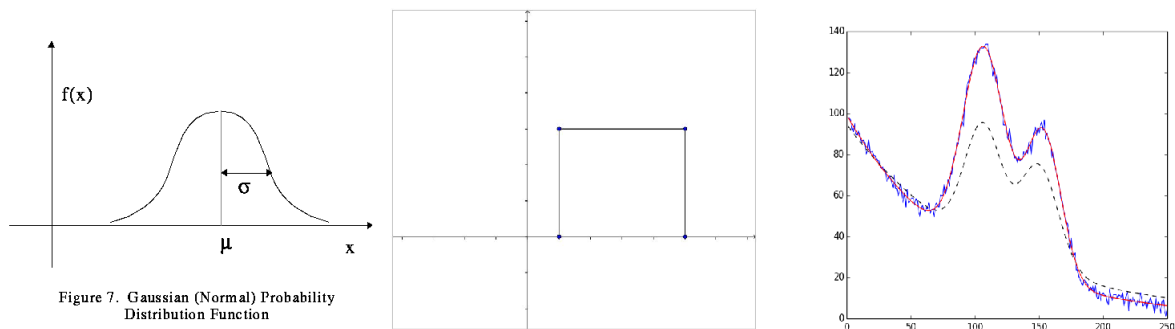
distribucije predstavljaju koordinate gdje se nalazilo najviše točaka. Ali, kakav oblik ima ta funkcija? Da li je možemo izraziti Gaussovom distribucijom (funkcijom gustoće vjerojatnosti) ili možda jednolikom distribucijom, ili možda pak nekom trećom? Postoji li univerzalni model takve distribucije? Odgovor na ovo pitanje nas vodi natrag na samu definiciju Mean-shift algoritma, a to je dio definicije koji kaže da se ovdje radi o neparametarskoj tehnici. Mean-shift analizira neparametarske distribucije. Kako bih objasnio što je neparametarska distribucija, reći ću nešto ponajprije o parametarskim.



Slika 3. Vizualni prikaz funkcije gustoće vjerojatnosti iz prostora podataka – R. Collins, Mean-shift Tracking Izvor Slike 3. -

<http://www.cse.psu.edu/~rtc12/CSE598G/introMeanShift.pdf>

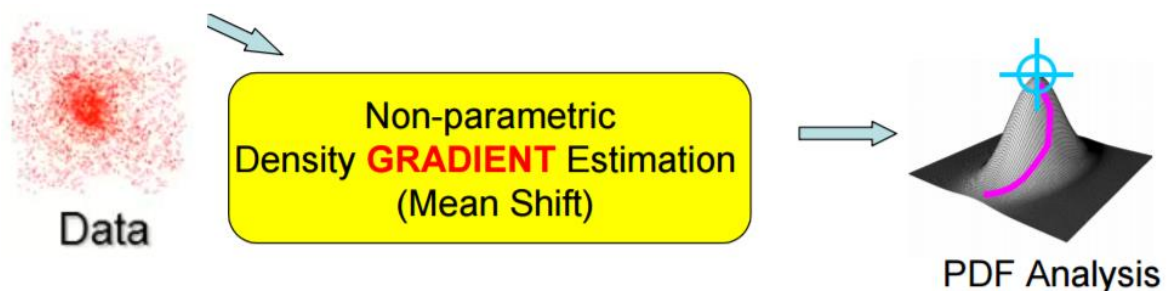
Uzmimo na primjer Gaussovu distribuciju i kao primjer uzorka populaciju te razmatramo visinu ljudi. Krivulja će ispasti zvonolikog oblika jer najviše ljudi će imati neku srednju prosječnu visinu, dok će najmanje ljudi biti oni najmanje i najveće visine. Tako će ispasti da je vjerojatnost osobe prosječne srednje visine te populacije najveća. Isto tako mogu uzet za primjer jednoliku distribuciju te primjer kocke; svaka vrijednost kocke ima jednaku vjerojatnost da će pasti prilikom njenog bacanja ako je kocka pravilna. Ove dvije distribucije te ostale poznate distribucije tipa Poissonova ili eksponencijalna distribucija – sve one su parametarske distribucije jer ih možemo izraziti formulama.



Slika 4. Parametarske distribucije redom- Gaussova, jednolika i višestruka Gaussova

<https://www.phy.ornl.gov/csep/mc/node19.html>, https://lmfit.github.io/lmfit-py/builtin_models.html, <http://math.tutorvista.com/statistics/continuous-distribution.html>

Ako za primjer uzmemo Gaussovu distribuciju, sve što trebamo je izračunati njene parametre, a to su srednja vrijednost te devijacija i mi imamo gotovu Gaussovu krivulju (jednostavan opis distribucije). Dakle iz uzoraka (u ovom slučaju bi to bile podatkovne točke) bismo izračunali parametre i nakon toga zaboravili na uzorke jer ih više ne trebamo. Sve se nalazi u krivuljama. Problem je što u većini slučajeva naše uzorke ne možemo prikazati parametarskim krivuljama. U možda nekim slučajevima bismo mogli i izraziti naš prostor s višestrukim Gausovim krivuljama (Slika 4) s više zvonolikih dijelova, ali i to možemo samo u nekim slučajevima. Ovdje ćemo samo govoriti o neparametarskim distribucijama. Osnovno svojstvo je to da nemaju formulu te da moramo spremiti uzorke što može biti puno podataka i što na kraju krajeva može biti memorijski i vremenski prezahtjevno, ali je jedini mogući način.

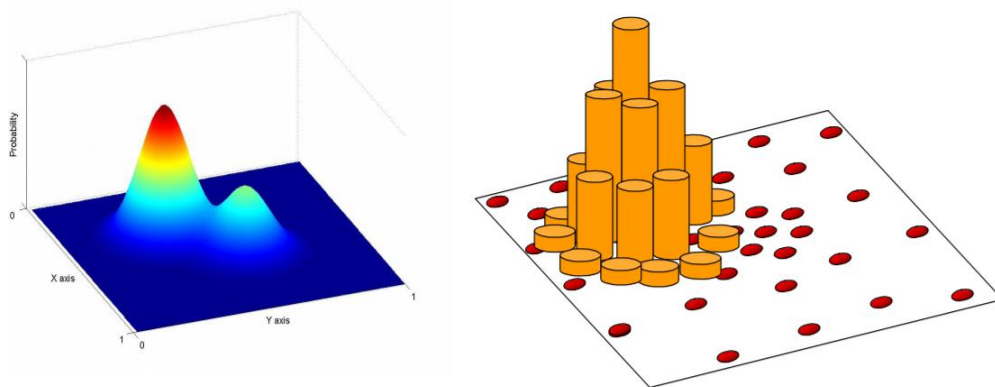


Slika 5. Uporaba mean-shifta za dobivanje gradijenta distribucije, R.Collins, Mean-shift Tracking Izvor Slike 5. - <http://www.cse.psu.edu/~rtc12/CSE598G/introMeanShift.pdf>

Na slici 5. možemo primjetiti vrlo bitnu činjenicu koju ću učvrstiti i dokazom. Bit je sljedeća – upotrebom Mean-shifta možemo dobiti gradijent neparametarske

distribucije. Prije svega bitno je napomenuti da se Mean-shift često još naziva i eng. mode-seeking algorithm pošto se može opisati kao alat za nalaženje “vrhova” unutar skupa podataka (data points). Mean-shift nalazi vrh distribucije koji se istovremeno dobiva gradijentom distribucije iz čega slijedi da je usmjeravanje Mean-shifta istoga smjera kao gradijent. Ovo je bitno svojstvo algoritma.

Na slici 6. se može vidjeti vizualizacija slike 3. u prostoru stanja gdje na mjesto koordinate stavljamo stupac koji predstavlja “košaru” odnosno broj točaka na tom mjestu u prostoru podataka te kako se to može preslikati na distribuciju. Može se primjetiti da veličina stupca prati gustoću točaka na tom području. Inače, “košare” ne moraju svaka predstavljati točno 1 koordinatu, one mogu predstavljati više koordinata pa zbog toga bi i veći broj točaka ušao u neku “košaru”. “Košare” ćemo kasnije razmatrati i u jednostavnijoj verziji distribucije - histogramu (binovi će biti skupine više vrijednosti piksela u koje ćemo stavljati piksele slike te time izraditi distribuciju).



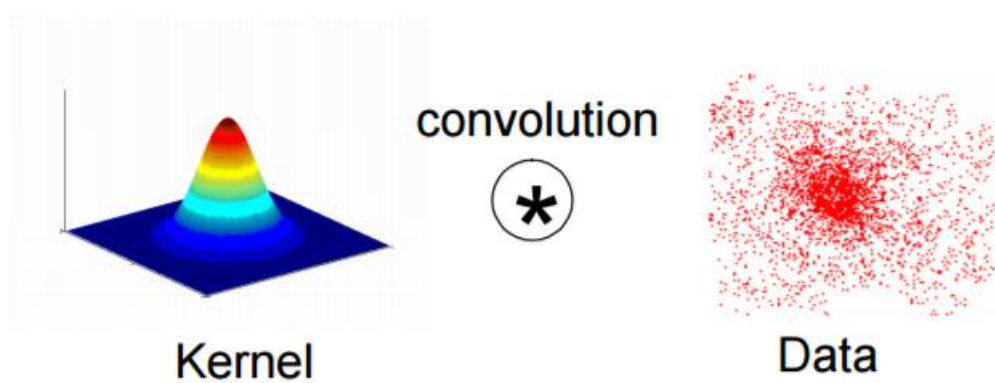
Slika 6. Vizualizacija prostora podataka “košarama” (eng. binovima) - R.Collins, Mean-shift Tracking Izvor- <http://www.cse.psu.edu/~rtc12/CSE598G/introMeanShift.pdf>

Napokon možemo odgovoriti i na pitanje o formuli distribucije. Da bismo prikazali PDF, koristit ćemo metodu takozvanu eng. Kernel density estimation (KDE, procjena gustoće kernelom) što je neparametarski način procjene PDF-a slučajne varijable. Ova metoda je također poznata još kao i eng. “Parzen window” metoda. Za samu procjenu koristimo kernel funkcije koje sam već spomenuo kada sam govorio o računanju težina u formuli za izračun Mean-shifta.

$$P(x) = \frac{1}{n} \sum_{i=1}^n K(x - x_i) \quad (4)$$

Formula (4) prikazuje općenit način izračuna PDF-a KDE metodom. To je najjednostavniji pristup u kojem “uglađujemo” podatke (eng. smooth a data set) na

način da radimo konvoluciju (Slika 7.) podataka (podatkovnih točaka, uzorka) sa stalnom kernel funkcijom K koja ima tzv. kernelovu širinu h (propusnost, eng. bandwidth). Uглаđivanjem podataka kreiramo približnu funkciju koja pokušava uhvatiti bitne dijelove podatkovnog skupa. Možemo izvući puno informacija u PDF iz podataka sve dok je “smoothing” razuman, odnosno sve dok je kernelova širina “ h ” razumna. Za kernel se u ovoj metodi još koristi termin eng. “smoother matrix” jer njime činimo linearnu transformaciju podataka u uglađene podatke. Uglavnom, nakon što izračunamo $P(x)$ možemo naći lokalni maksimum koristeći uspon gradijentom. Uskoro slijedi i dokaz svojstva da Mean-shift vektor ima smjer gradijenta distribucije.



Slika 7. Prikaz tehnike eng. Kernel density estimation – R. Collins,
<http://www.cse.psu.edu/~rtc12/CSE598G/introMeanShift.pdf>

Sada napokon imamo znanje o kontekstu gdje koristimo kernel funkcije te ću ovime definirati samu kernel funkciju. Podsjećam, kernel funkcija određuje koliku težinu pridjeliti određenoj točki u prostoru podatkovnih točaka.

Kernel funkciju možemo opisati kao nenegativnu realnu integrabilnu funkciju. Za primjenu je dobro da funkcija zadovoljava sljedeća dva uvjeta:

$$\int_{-\infty}^{\infty} K(u) du = 1 \quad (5)$$

$$K(-u) = K(u), \forall u \quad (6)$$

Prvi uvjet (5) osigurava da KDE tehnikom dobijemo kao rezultat PDF. Drugi uvjet (6) osigurava da prosjek odgovarajuće distribucije je jednak prosjeku korištenog uzorka (npr. prostora podatkovnih točaka). Nadalje, opisati ću tri konkretne kernel funkcije. Uniformna kernel funkcija je funkcija koja pridaje jednaku težinu svim točkama oko određene točke x u nekom radijusu oko te točke (formula (7)).

$$K_U(x) = \begin{cases} c, & \|x\| \leq 1 \\ 0, & \text{inače} \end{cases} \quad (7)$$

Jedna slična verzija ovome kernelu je ravni kernel (eng. flat kernel) koji umjesto da pridaje neodređenu konstantu c , pridaje 1 svim točkama oko sebe, ali u neodređenom okruhu definiranom radijusom λ . Često se koristi u kontekstu Mean-shift algoritma (formula (8)).

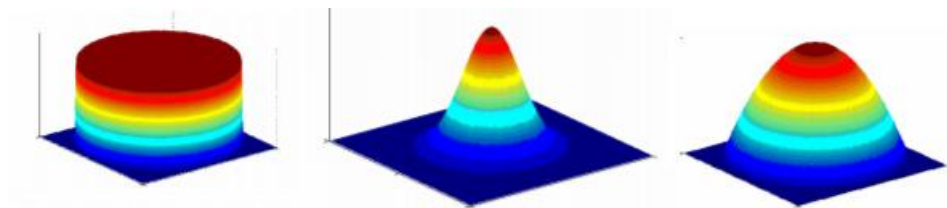
$$K_F(x) = \begin{cases} 1, & \|x\| \leq \lambda \\ 0, & \|x\| > \lambda \end{cases} \quad (8)$$

Druga često korištena i efikasnija od uniformne (jednolike) funkcije je Gaussova (normalna) funkcija. Razlika od uniformne je u tome da kako se udaljujemo od točke x tako težina koju pridjeljujemo točkama se smanjuje eksponencijalno (formula (9)).

$$K_N(x) = c \cdot e^{-\frac{1}{2}\|x\|^2} \quad (9)$$

Najučinkovitija od svih kernel funkcija je Epanechnikovova funkcija. Kod nje težine kvadratno opadaju kako se udaljujemo od središnje točke x , ali sve u radijusu od 1, inače težina je 0 (formula (10)).

$$K_E(x) = \begin{cases} c(1 - \|x\|^2), & \|x\| < 1 \\ 0, & \text{inače} \end{cases} \quad (10)$$



Slika 8. Prikaz kernelovih funkcija 3-D modelom redom: jednolika, Gaussova, Epanechnikovova - R. Collins,

<http://www.cse.psu.edu/~rtc12/CSE598G/introMeanShift.pdf>

Možemo vidjeti jedno zajedničko svojstvo kod svih kernel funkcija, a to je naravno ovisnost o udaljenosti točaka od središnje točke. Kada bismo uzeli Gaussovu kernel funkciju, PDF bi pomoću KDE tehnike izgledala na sljedeći način (formula (11)):

$$P(x) = \frac{1}{n} \sum_{i=1}^n c_i \cdot e^{-\frac{(x-\mu_i)^2}{2\delta_i^2}} \quad (11)$$

Kao što vidimo, u formuli smo zamijenili dio u koji stavljamo kernelovu funkciju. Samo da se podsjetimo, kao što sam rekao KDE se koristi kao neparametarski pristup gdje ne postoji formula koju koristimo za izračun parametara (zbog čega ne bismo koristili uzorke). Ovdje je poanta da moramo procijeniti svaki dio te formule (varijabilne

dijelove) kao što su c_i te δ_i . Želimo saznati koja je vjerojatnost u točki x pa ćemo promatrati udaljenost x od svih podatkovnih točaka u skupu koji gledamo (μ_i), procijeniti varijabilne dijelove te izračunati vjerojatnost (u ovom slučaju normalnom funkcijom). Bitno je primjetiti da sve točke doprinose vjerojatnosti x -a, a doprinos svake točke će ovisiti o tome koliko je ta druga točka (μ_i), blizu središnje (x). Ako se nalazimo daleko od (x) vrijednost funkcije će biti jako mala te neće puno pridonijeti ukupnoj sumi, ali svejedno je moramo dodati u nju zbog računanja vjerojatnosti.

$$K(x) = ck(\|x\|^2) \quad (12)$$

U praksi najčešće koristimo formu kernel funkcije koja se naziva **profil** (eng. profile). U svim opisanim kernel funkcijama se nalazio parametar $\|x\|$ što je u formuli označavalo da se radi o radijalnoj simetričnoj kernel funkciji, odnosno da su sve vrijednosti težine na istoj udaljenosti od središnje točke x imali istu vrijednost. Također pošto se radi o dvodimenzionalnom prostoru ova notacija se još kvadrira pa imamo notaciju $\|x\|^2$, a točke koje imaju jednaku vrijednost te notacije se nalaze na istoj kružnici. Ovime dobivamo više specifičan model kernel funkcije definiran slovom " k " (formula (12)) te možemo napraviti susptituciju općenitog modela kernel funkcije profil modelom (formula (13)).

$$P(x) = \frac{1}{n} \sum_{i=1}^n K(x - x_i) = \frac{1}{n} c \sum_{i=1}^n k(\|x - x_i\|^2) \quad (13)$$

Nakon toga diferenciramo jednadžbu po (x) (formula (14)).

$$\nabla P(x) = \frac{1}{n} c \sum_{i=1}^n \nabla k(\|x - x_i\|^2) = \frac{1}{n} 2c \sum_{i=1}^n (x - x_i) k'(\|x - x_i\|^2) \quad (14)$$

Zatim pojednostavljujemo notaciju derivacije profila $k'(x) \cdot (-1) = g(x)$. (formula(15)).

$$\nabla P(x) = \frac{1}{n} 2c \sum_{i=1}^n (x_i - x) g(\|x - x_i\|^2) \quad (15)$$

U sljedećem koraku pomnožimo sve što možemo unutar sume čime dobivamo dvije sume kao što možemo vidjeti u formuli (16).

$$\nabla P(x) = \frac{1}{n} 2c \sum_{i=1}^n x_i g(\|x - x_i\|^2) - \frac{1}{n} 2c \sum_{i=1}^n x g(\|x - x_i\|^2) \quad (16)$$

Pred kraj ćemo ovo još jednom pojednostaviti tako da se vidi kako je Mean-shift povezan s gradijentom PDF-a. To vidimo u formuli (17).

$$\nabla P(x) = \frac{1}{n} 2c \sum_{i=1}^n g(\|x - x_i\|^2) \left[\frac{\sum_{i=1}^n x_i g(\|x - x_i\|^2)}{\sum_{i=1}^n g(\|x - x_i\|^2)} - x \right] \quad (17)$$

Na samom kraju pojednostavljujemo u našoj notaciji izraz $g(\|x - x_i\|^2)$ sa g_i (formula (18)).

$$\nabla P(x) = \frac{1}{n} 2c \sum_{i=1}^n g_i \left[\frac{\sum_{i=1}^n x_i g_i}{\sum_{i=1}^n g_i} - x \right] \quad (18)$$

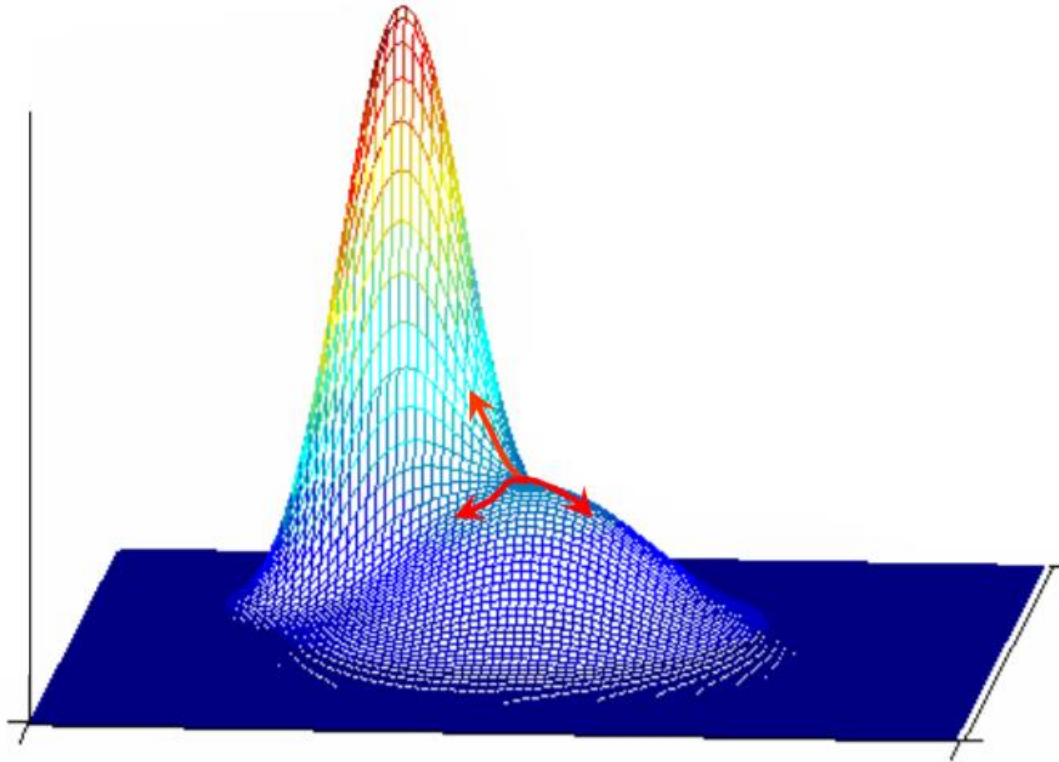
Kao što vidimo dio unutar uglatih zagrada predstavlja Mean-shift $m(x)$ (formula (19)).

$$\nabla P(x) = \frac{1}{n} 2c \sum_{i=1}^n g_i \cdot m(x) \quad (19)$$

Ovime smo dokazali jednu jako zanimljivu stvar. Dokazali smo da je Mean-shift zapravo gradijent funkcije vjerojatnosti gustoće u neparametarskom obliku. Odnosno jednaku ćemo stvar dobiti postupkom izrade funkcije gustoće vjerojatnosti Kernelovom procjenom gustoće i nalaska njezinog gradijenta te samim postupkom “prolaska” po svim podatkovnim točkama, pridruživanja težina te računanja Mean-shift vektora. Samim time što znamo vizualizirati gradijent naše PDF funkcije (i uzimajući u obzir ovaj dokaz) znamo da Mean-shift zasigurno konvergira prema vrhovima distribucije te smo ovime dokazali sam algoritam.

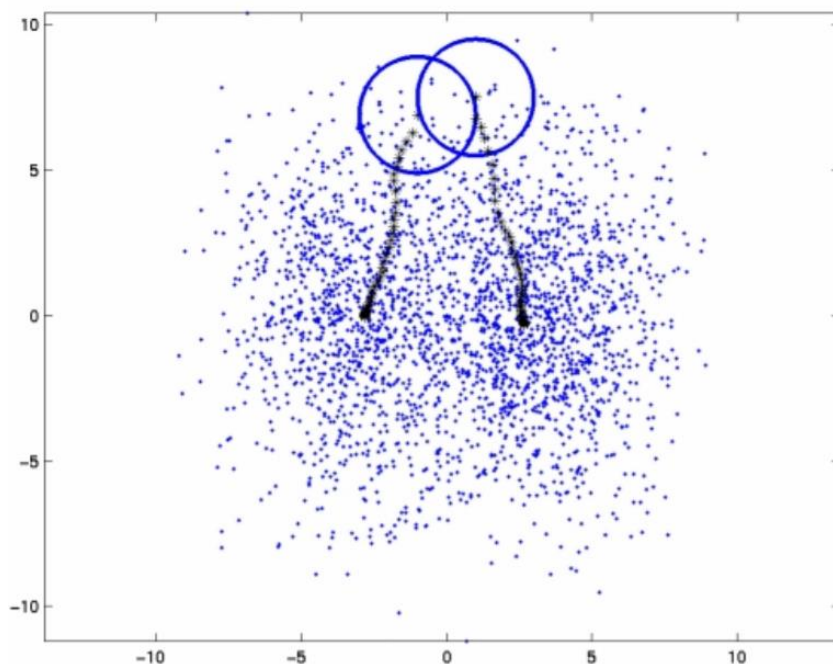
Jako bitna stvar je napomenuti da je ovo bio dokaz za konačan broj stacionarnih točaka i da vrijedi za bilo koji n-dimenzionalan prostor. No, jak dokaz za konvergenciju algoritma koristeći općenitu kernel funkciju u n-dimenzionalnom prostoru za beskonačan broj točaka još fali. Za beskonačan broj točaka je dokazana konvergencija samo u jednodimenzionalnom prostoru s time da je kernel funkcija bila diferencijabilna profil funkcija koja je strogo opadala s udaljenošću od središta. Nažalost, slučaj kada je prostor jednodimenzionalan ima jako malo primjena u stvarnom svijetu.

Sljedeće što se možemo zapitati je što ako postavimo regiju interesa u našem prostoru na početku na slučajno mjesto (tako zapravo i radimo) te dođemo do nekog lokalnog vrha, ali ne možemo doći do globalnog jer se ne možemo pomaknuti iz lokalnog. Dakle, zaglavili smo u jednom manjem vrhu. Ovaj prikaz se može vidjeti na slici 9.



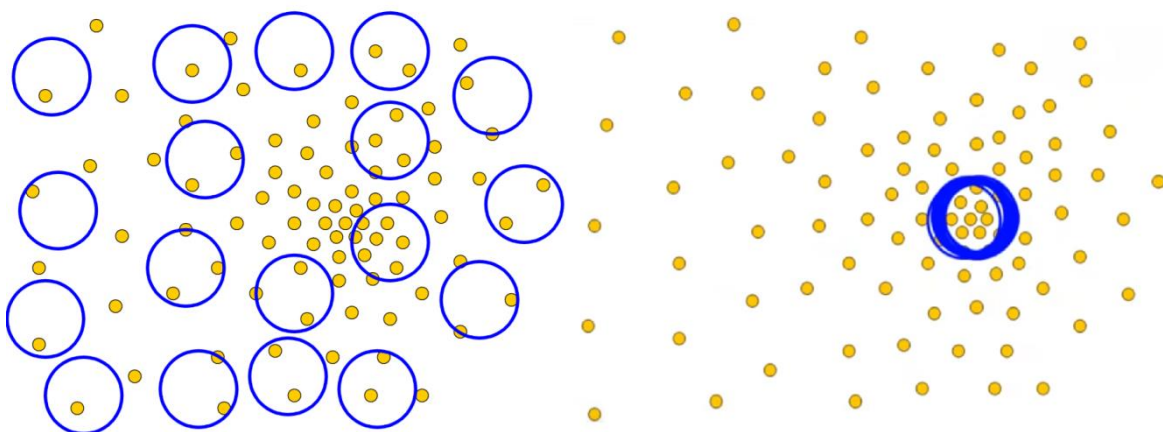
Slika 9. – prikaz lokalnog vrha (maksimum, eng. mode), odnosno eng. saddle point - R. Collins, <http://www.cse.psu.edu/~rtc12/CSE598G/introMeanShift.pdf>

Kako riješiti ovaj problem? Očito sada osim samih uzoraka moramo pamtiti i neke druge podatke kako bismo dobili globalni vrh. Na slici 10. se vide dva početna prozora (regije interesa) te put kojima prolaze do svojih lokalnih vrhova. Trajektorije tih prozora (zabilježene crnim točkama u skupu podatkovnih točaka) su zapravo smjerovi najstrmijih uspona prema vrhovima distribucija. No, to su lokalni vrhovi te mi i dalje ne možemo znati samo na temelju ovoga koji vrh je najviši u cjelom prostoru.



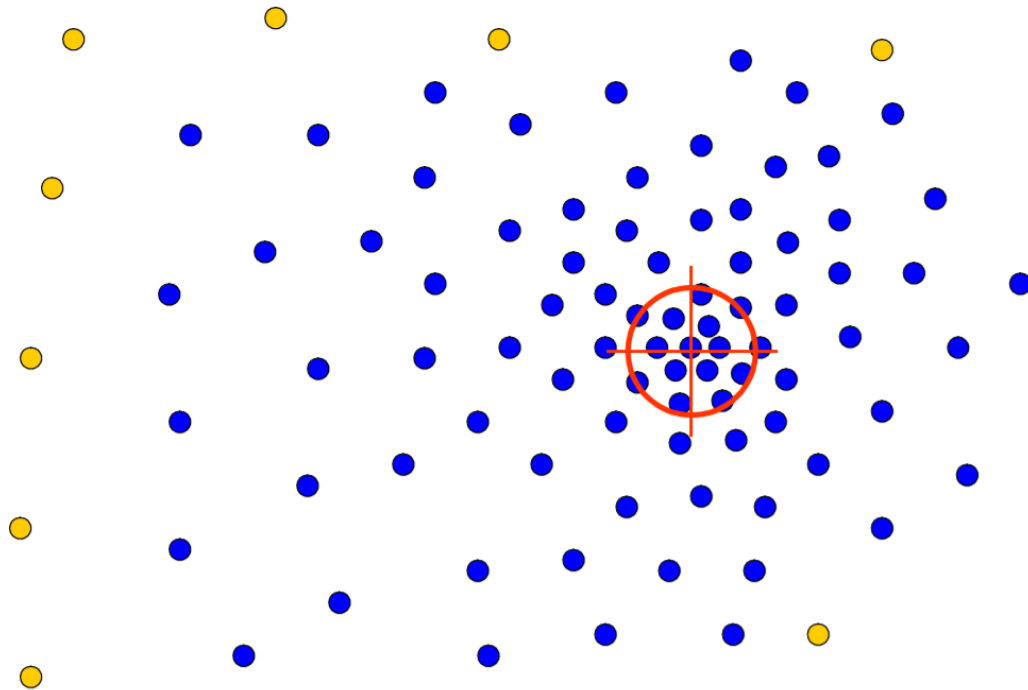
Slika 10. – prikaz pronalaska više lokalnih maksimma (eng. mode) ovisno o početnom ROI –Bob Fisher, 02.02.2006., 17:32, University of Edinburgh,
http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/TUZEL1/MeanShift.pdf

Ovime ćemo još jednom nadograditi algoritam na način da ćemo pronaći sve vrhove u prostoru podatkovnih točaka na način da paralelno izvodimo prije opisan jednostavan Mean-shift algoritam u svakoj točki konačnog skupa podatkovnih točaka. Sljedeće što ćemo napraviti je kako bismo shvatili koji je najviši vrh u prostoru je izmjenjivanje pozicije vrha i provjera da li se vraćamo ponovno natrag prema njemu. Da bi se promjenila pozicija vrha mora se naravno i mjenjati gustoća na određenoj poziciji u prostoru. Na slici 11. možemo vidjeti prostor podatkovnih točaka koji smo već prikazali na slici 1.



Slika 11. – Mean-shift algoritam započet na više mjesta paralelno - R. Collins, <http://www.cse.psu.edu/~rtc12/CSE598G/introMeanShift.pdf>

Na slici možemo jasno vidjeti više početnih regija interesa i svaka se obavlja sama za sebe. Ovaj prostor nije imao više maksimuma pa se konačna regija interesa za svaki slučaj skoro našla na identičnom mjestu. Također može se primjetiti sljedeća stvar prikazana na slici 12.



Slika 12. - Prikaz svih lokacija koje konvergiraju k istom vrhu (eng. basin of attraction) R. Collins, <http://www.cse.psu.edu/~rtc12/CSE598G/introMeanShift.pdf>

Sve točke plave boje, odnosno sve regije interesa kojima je to predstavljalo slučajno središte su konvergirale k istom vrhu označenom regijom interesa crvene boje. Cijelo to područje plave boje predstavlja eng. basin of attraction. Sve točke koje su u istom "bazenu" su udružene u isti eng. cluster (svežanj). Što se tiče preostalih točaka (žute boje), njihove regije interesa nisu konvergirale prema najgušćem području prostora već su ostale na svojim mjestima pošto su mean-shift vektori bili nul-vektori, te one svaka za sebe pripada svom nezavisnom "bazenu". Nadalje, nećemo ulaziti dublje u analizu "cluster".

3. Mean-shift - primjena

Napokon se možemo zapitati kako sa svim ovim znanjem možemo riješiti problem praćenja u stvarnom vremenu. Odnosno kako ćemo pratiti naše igrače u nogometnoj utakmici? Ovdje nemamo prostor koji čini skup podatkovnih točaka koje su manje ili više stisnute jedna s drugom. Sve što na početku imamo je video utakmica koja traje određen broj sekundi. Video koji ja koristim za ovaj završni rad, odnosno video gdje primjenjujem Mean-shift algoritam je nogometna utakmica Dinamo-Rijeka - video traje 5 minuta i 29 sekundi (+ 64 milisekunde). Video sveukupno ima 8241 sekvencu što se da iščitati iz podatka da jedna sekunda ima 25 sekvenci. Također, visina sekvence je 720 piksela, dok je širina sekvence 1280 piksela. Slika 13. prikazuje prvu sekvencu videa.



Slika 13. – prikaz prve sekvence nogometne utakmice

Bitno je napomenuti da je kamera statička, odnosno uvijek ima jednak pogled prema terenu.

Dakle, sve što imamo na početku je slika koju čine pikseli jednako udaljeni jedan od drugoga. Svaki piksel ima određenu boju koja je definirana s tri komponente RGB (BGR) modela boje, a to su crvena, zelena i plava boja te svaka ima određen udio u jednom pikselu (0% - 100%). Uz to još imamo podatak što želimo pratiti na slici (odnosno cjelom videu), a to je određeni igrač, te još imamo njegov prikaz na slici (u obliku piksela te imamo vrijednosti komponenti boja unutar svakog piksela). Kako

bismo odabrali model u inicijalnoj sekvenci, označiti ćemo ga plavim pravokutnikom linije debljine 2. Na slici 14. se vidi označen Dinamov igrač plave boje.

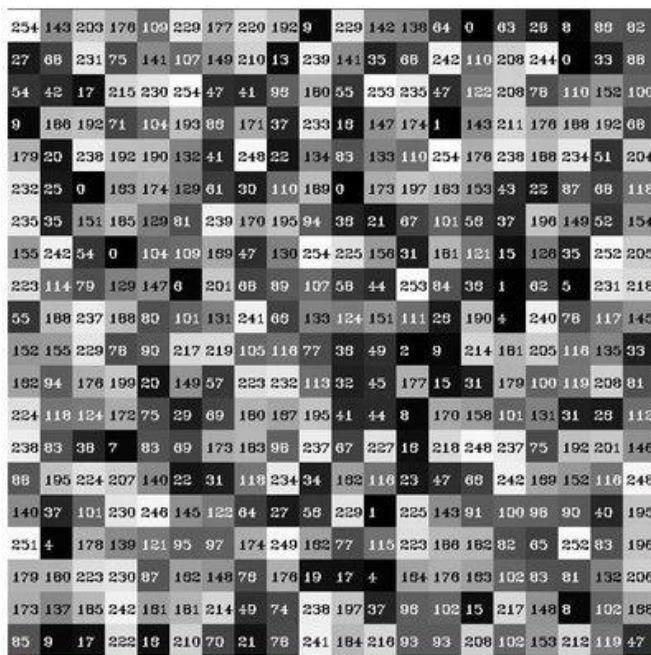


Slika 14. – odabrani model u početnoj sekvenci – ujedno i regija interesa – sekvenca iz videa Dinamo-Rijeka

Nakon što odaberemo model potrebno je odabrati svojstveni prostor. Ovime smo napokon došli do posljednjeg nepoznatog dijela naše početne definicije Mean-shifta. Rekli smo “neparametarski način analize svojstvenog prostora...”. Od početka radimo na jednostavnom prikazu prostora gdje promatramo točke te promatramo njihovu gustoću. Iako je prostor jednostavan, za svaku točku prostora su nam trebala dva podatka: x i y koordinata u dvodimenzionalnom koordinatnom sustavu. Sada ćemo kao podatak trebati broj piksela koji ima određenu boju ili koji se nalazi unutar raspona određenih boja. Svaku boju čine intenziteti crvene, zelene i plave boje. Iz ovoga vidimo potpuno dva različita prostora u kojima proučavamo različita svojstva. Ako naša slika predstavlja objekt, odnosno ako ju gledamo kao cjelokupni prostor, možemo je definirati vektorom koji za indekse ima postavljene određene boje ili raspone boja, dok za vrijednosti imamo broj piksela za taj raspon. Ovo nazivamo **svojstvenim vektorom**. Svako svojstvo možemo promatrati kao jednu dimenziju problema. Tako dolazimo do definicije **svojstvenog prostora** (eng. feature space) koji se prikazuje svojstvenim vektorom. Zapošto bismo koristili ovaj prostor? Zato što možemo pojednostaviti problem – slika je vrlo složen element i kada bismo nju koristili kao atom za obradu i praćenje objekata, naš problem bio bi memorijski i vremenski prezahtjevan. Želimo smanjiti broj dimenzija koje bismo razmatrali pri praćenju. Također, ovime izvodimo i generalizaciju – izrađujemo koncept sličnosti. Pošto u ovom slučaju imamo jednu vrijednost, radi se o jednodimenzionalnom svojstvenom prostoru (radi se o boji). Osim kvantiziranog prostora boja, također smo

za sliku mogli računati kvantizirani prostor intenziteta ili prostor gradijenta, ili prostor smjerova, ili pak nešto peto. No kvantizirani prostor boja je najjednostavniji i uz to daje više podataka od nekih drugih prostora (kvantizirani prostor intenziteta koji ima točno 256 različitih intenziteta sive boje).

Nakon što smo odabrali model, i izračunali svojstveni prostor, možemo napraviti prikaz modela u svojstvenom prostoru, odnosno distribuciju/funkciju vjerojatnosti gustoće te na samom kraju možemo primijeniti ideju Mean-shift algoritma. U ovom slučaju grafički prikaz distribucije nazivamo **histogramom**. Histogram bi se najlakše moglo opisati kao prikupljeni numerički podaci organizirani u unaprijed definiran skup binova. Ti podaci mogu biti bilo koje svojstvo koje je korisno za opis slike. U našem slučaju to su vrijednosti intenziteta određene boje.



Slika 14. Primjer matrice koja sadrži informacije o slici (intenziteti sive boje od 0 – 255) – OpenCV documentation – Histogram calculation,

http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_calculation/histogram_calculation.html

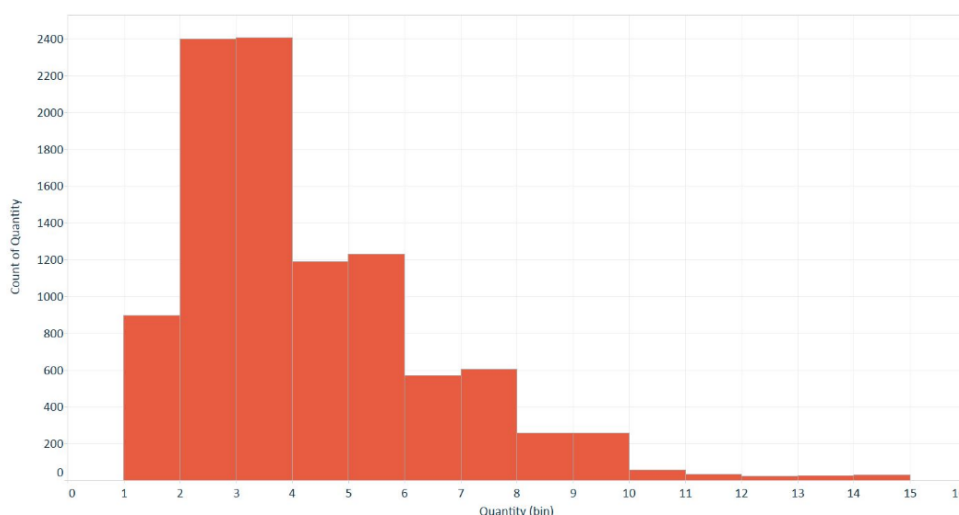
Na slici 14. Možemo vidjeti prikaz jedne slike koja je u sivim tonovima (sadrži piksele koji imaju vrijednosti intenziteta sive boje 0 – 255). Pošto znamo koliko imamo različitih vrijednosti intenziteta, možemo na vrlo jednostavan način prebrojati podatke na organiziran način. Prva ideja je uzeti 256 binova, ali znajući da vjerojatno

ćemo tada imati vrlo malo piksela u svakom binu (pogotovo uzimajući u obzir ovu matricu), uzet ćemo sveskupa 15 binova od kojih svaki sadrži piksele od 16 vrijednosti intenziteta sive boje. To izgleda ovako (formula (20) i formula (21)).

$$[0 - 255] = [0 - 15] \cup [16 - 31] \cup \dots \cup [240 - 255] \quad (20)$$

$$range = bin_1 \cup bin_2 \cup \dots \cup bin_{15}$$

Na ovaj način brojimo piksele koji padaju u određen raspon vrijednosti intenziteta od svakog bin_i . Na taj način možemo dobiti nešto što je oblika kao na slici 15. Apscisa predstavlja indeks bina, dok ordinate predstavlja broj piksela za svaki bin.



Slika 15. Raspodjela po binovima – Ryan Sleeper – How to make a histogram? - <http://www.evolytics.com/blog/tableau-201-make-histogram/>

Zaključno možemo reći da postoje tri bitna elementa histograma.

- Broj dimenzija - broj parametara po kojima želimo brojati piksele (u ovom primjeru sa slike 14. govorimo o jednoj dimenziji – broje se vrijednosti intenziteta sa slike sivih tonova)
- Broj binova – poddioba za svaku dimenziju, u primjeru su to 16 binova za svaku
- Raspon vrijednosti – granice podataka koji se mjere -- u primjeru 0 – 255

Kada bismo imali dvodimenzionalni svojstveni prostor, naš histogram bi bio trodimenzionalan, “x” i “y” os bi svaka imala svoju podjelu binova, dok bi “z” os imala broj piksela koji se nalaze u oba bina istovremeno. Sličnu stvar smo imali u našem primarnom primjeru s prostorom podatkovnih točaka (npr. slika 6.) gdje su x i y os

predstavljale vrijednosti x i y koordinata, dok je “ z ” os bila broj točaka na tim koordinatama.

Još bih prokomentirao koncept sličnosti koji sam spomenuo kod generalizacije koju dobivamo histogramom slike. Bitna je činjenica da histogramima pojednostavljujemo slike, te neki histogrami koji su jednaki mogu prikazivati različite slike. Možda se igrač nalazi negdje drugdje na terenu, ali i dalje svojom površinom prekriva jednak (ili sličan) broj piksela te je okrenut u istoj perspektivi kao i na nekoj drugoj slici.

Zašto je histogram isti? Iz razloga, što histogram uzima u obzir samo jednu značajku, a to je broj piksela određene boje, što nije zavisno o poziciji unutar slike. Ova činjenica je vrlo bitna i korisna jer upravo zbog toga možemo uspoređivati histograme više različitih slika i dobivati zaključke na temelju te sličnosti.

Sada kada smo napokon opisali histogram, opisat ću načine na koji taj histogram možemo iskoristiti za dobivanje razdiobe vjerojatnosti položaja objekta na sljedećoj sekvenci. Dakle, na neki način prije upotrebe mean-shifta moramo dobiti funkciju koja opisuje vjerojatnost da se objekt nalazi na određenoj poziciji na slici. Opisati ću najprije postupak koji opisuje usporedbu histograma, a zatim postupak koji ću i primijeniti u vlastitoj implementaciji.

3.1 Usporedba histograma

Dakle, najprije smo odabrali prvotni model omeđen crnom elipsom na slici 16. (lijevi dio). Za svojstveni prostor smo odabrali kvantitativni prostor boja te nakon toga radimo grafički prikaz (histogram) distribucije boja unutar te crne elipse u kojoj se nalazi naš model.



Slika 16. Prikaz modela, regije interesa te pronalazak sličnog histograma

Nakon toga slijedi dinamički dio. Postavljamo za početak regiju interesa koja nas zanima u kojoj se na samom početku nalazi i naš objekt (označeno je plavim pravokutnikom na slici 16.). Postavljanjem regije interesa, postavljamo prostor koji ćemo nadalje promatrati, odnosno ograničit ćemo se samo na taj prostor pri traženju objekta. Nakon toga, prebacujemo se na novu sekvencu te još uvijek imamo postavljen ROI na istom mjestu. Unutar tog ROI-a tražimo najsljedniji histogram onom histogramu objekta iz protekle sekvence (što je zapravo sličnost funkcija vjerojatnosti gustoće).

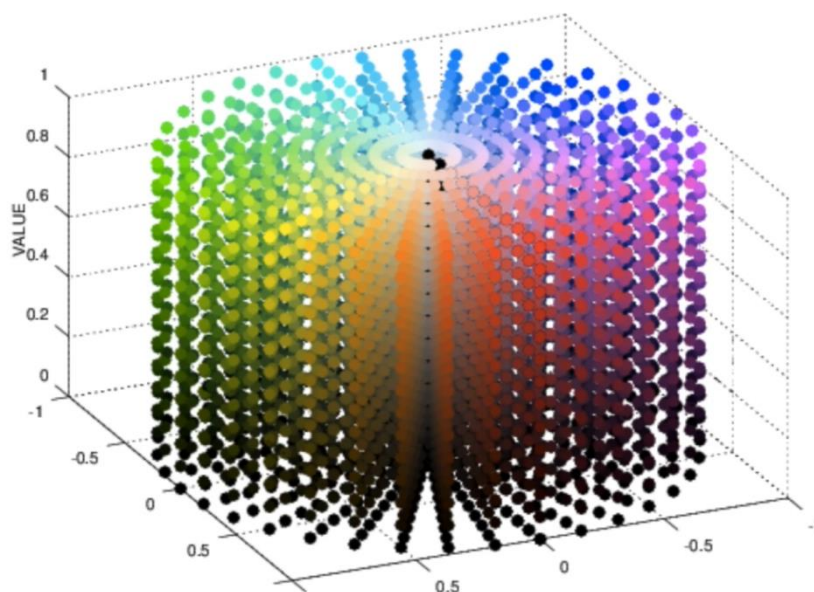
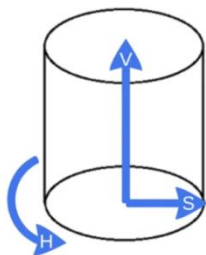


Slika 17. Prikaz modela na trenutnoj sekvenci i modela na sljedećoj sekvenci te njihovih histograma - matplotlib

Kada računamo histogram, gledamo sve moguće piksele unutar crne elipse (slika 17.) te ih razvrstavamo po binovima. Za izradu histograma sa slike 17. koristimo Pythonovu biblioteku matplotlib. Prije same podjele piksela po binovima, najčešće mjenjamo RGB (tj. BGR) sustav boja u HSV sustav boja jer HSV sustav prikazuje boje na manje kompleksan način i ovo je skoro pa uvijek dio postupka koji radimo prije Mean-shifta. Kako bismo prikazali HSV prostor boja? HSV je cilindrični prikaz boja iz RGB prostora boja. Također imamo tri komponente:

- a) H – kut unutar modela (Hue) – vrijednosti 0 - 179
- b) S – radijus unutar modela (Saturation) – vrijednosti 0 - 255
- c) V – visina unutar modela (Value) – vrijednosti 0 - 255

Hue predstavlja dominantnu boju kakvu vidi promatrač, Saturation (zasićenje) je količina bijele svjetlosti pomiješane s Hue, dok Value predstavlja intenzitet te boje (što je niža vrijednost, sličnija je crnoj boji, a što je viša, sličnija je istoimenoj boji). Prikaz ovog modela se vidi na slici 17.



Slika 17. – HSV prostor boja – Thales Sehn Korting, 09.07.2015.

https://www.youtube.com/watch?v=NAw2_NtGNaA

HSV je bolji u detekciji i praćenju objekata općenito od RGB-a zbog svoje Hue komponente koja apstrahira boju odvajajući je od zasićenja (saturation) i osvjjetljenja (value). Želimo da je slika lakša za analizu, a to nam omogućuje HSV. Najbolji primjer zašto je HSV bolji od RGB su sijene. Ako gledamo objekt koji je na jednoj slici jednim djelom pokriven sijenom, a drugi dio nije, oba dijela će imati sličnu Hue komponentu, dok će se razlika očitovati u saturation ili value komponenti. U RGB sustavu bi imali potpuno drukčije boje na tom istom objektu.



Slika 18. – prikaz prve sekvence u HSV prostoru boja – sekvenca iz utakmice Dinamo-Rijeka

Na slici 18. je prikazana prva sekvenca u HSV prostoru boja. Nakon promjene sustava,

idemo po svakom pikselu unutar modela i razvrstavamo ga po određenom broju binova. U ovom slučaju možemo uzeti 180 binova maksimalno pošto smo konvertirali sustav pa bi imali $bin_0, bin_1, \dots, bin_{179}$. Na slici 17. oba histograma imaju sveukupno 180 binova (radi ljepše slike sam uzeo binove od 0 do 55 pošto su nadalje svi binovi prazni). Kako sada iskoristiti ovaj histogram? Kako da ga izrazimo matematički? Očito ćemo ga izraziti svojstvenim vektorom.

$$\vec{q} = \{q_u\}_{u=1..m} \quad (21)$$

U formuli (21) je prikazan vektor koji predstavlja histogram modela u trenutnoj sekvenci. Bitna stvar je ta da niti jedan q_u ne predstavlja broj piksela unutar određenog bina nego svojstvenu funkciju vjerojatnosti (u ovom slučaju boje). Način na koji bismo dobili vjerojatnost da je piksel u određenom rasponu boja (tj. da je u određenom binu) se izračunava općenito (iza ba histograma) na način izražen u formuli (21a).

$$p(u) = C \sum_{x_i \in S} k(\|x_i\|^2) \delta[S(x_i) - u] \quad (21a)$$

Što možemo sve vidjeti unutar ove formule? Naravno možemo vidjeti ono što smo koristili u u podatkovnom prostoru, a to je diferencijabilna kernelova funkcija koja opada kvadratno kako se pomičemo od centra. Sada tu ulazi još jedna jako bitna stvar. Nećemo gledati sve okolne piksele, veće samo određene, a to su samo oni koji se nalaze unutar tog bina za koji promatramo vjerojatnost. Na koji način je to opisano u formuli (21a)? Dirac funkcijom koja će biti 1 samo u slučaju kada će $S(x_i)$ funkcija vratiti indeks određenog bina, inače vraća 0. Dakle, samo gledamo da li se točka nalazi u binu „u“? Ako se nalazi pridodajemo težinu te koordinate ukupnoj sumi za vjerojatnost za taj bin, inače ne radimo ništa. Napokon vidimo sličnost sa našim podatkovnim prostorom. Analogno možemo opisati vektor koji predstavlja histogram u sljedećoj sekvenci, a on ovisi o točki y koja se nalazi unutar regije interesa(formula (22)).

$$\overrightarrow{p(y)} = \{p_u(y)\}_{u=1..m} \quad (22)$$

Kada bismo opisali te brojeve kao vjerojatnosti, zbroj svih vjerojatnosti bi bio 1. To je omogućeno normalizacijskim faktorom C unutar formule (21a).

Navedeno možemo vidjeti kao dva izraza u formuli (22a).

$$\sum_{u=1}^m q_u = 1, \sum_{u=1}^m p_u = 1 \quad (22a)$$

Zaključno vezano uz ove vektore, bitno je reći da oni preciznije ne predstavljaju same histograme oko određenih točki, već predstavljaju težinske histograme (eng.

weighted histograms). Zašto? Jer računajući vjerojatnosti bina smo koristili kernelovu funkciju za doprinos okolnih točaka.

Napomenimo da pronađeni histogram najvjerojatnije neće biti identičan histogramu iz prošle sekvence, ali će sigurno biti najsličniji.

I sada napokon želimo usporediti ova dva histograma. To će obaviti funkcija sličnosti histograma izražena formulom (23).

$$f(y) = f[\vec{q}, \vec{p}(y)] \quad (23)$$

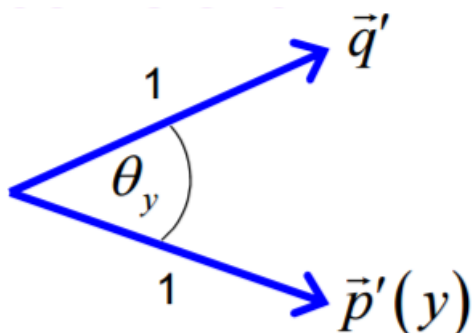
Način na koji ćemo napraviti usporedbu je pomoću poznatog eng. **Bhattacharyya coefficient** (Bhattacharyya koeficijent). Taj koeficijent predstavlja općenito mjeru između dva statistička uzorka. Dakle, kao što sam rekao i kao što je opisano u navedenim izrazima, ovdje se radi o dva vektora. Nadalje želimo naći skalarni produkt između ta dva vektora, odnosno kut između ta dva vektora koji ćemo dobiti pomoću produkta. Kako? Da budemo precizniji ne radimo skalarni produkt vektora p i q , već specifično njihovih korijena $\vec{p}'(y) = (\sqrt{p_1}, \dots, \sqrt{p_m})^T$ te $\vec{q}' = (\sqrt{q_1}, \dots, \sqrt{q_m})^T$. Na ovaj način koeficijent ima jednostavniju notaciju. Zašto? Najprije se možemo podsjetiti izraza za skalarni produkt te njegove veze sa kosinusom kuta (formula (24)).

$$\cos(\alpha) = \frac{a \cdot b}{|\vec{a}| \cdot |\vec{b}|} \text{ gdje je } a \cdot b = \sum_{i=1}^n a_i b_i \quad (24)$$

Ako uzmemo u obzir da je modul vektora koji sadrži korijene vrijednosti čiji je zbroj 1, očito će modul jednog takvog vektora biti korijen iz 1, odnosno 1, a time i sam umnožak 2 takva modula mora biti 1. Iz ovoga slijedi sljedeća formula (25).

$$\rho(p(y), q) = \sum_{u=1}^m \sqrt{p_u(y) \cdot q_u} \quad (25)$$

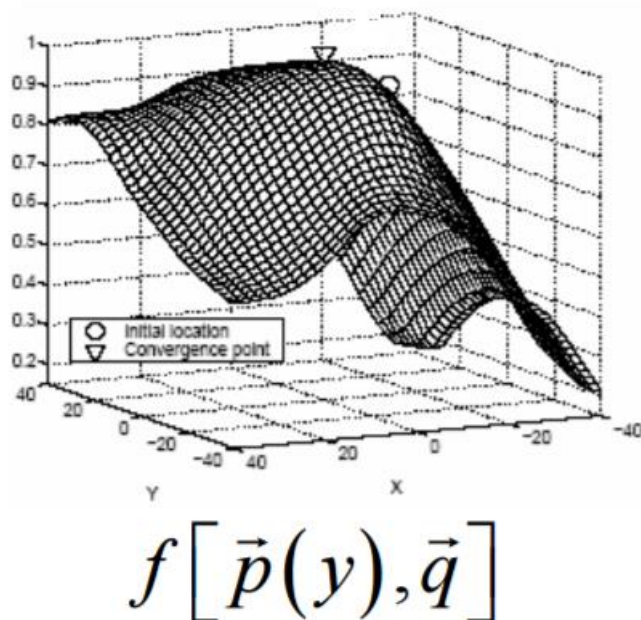
Ova formula baš zbog upotrebe korijena tih vektora se riješila nazivnika formule (24). Na slici 19. su prikazana dva vektora te kut između njih.



Slika 19. Prikaz vektora $\vec{p}'(y)$ te \vec{q}' i kuta između njih, R. Collins,

<http://www.cse.psu.edu/~rtc12/CSE598G/introMeanShift.pdf>

Obavezno se mora spomenuti kako $\rho(p(y), q)$ u formuli (24) zamjenjuje $\cos(\theta)$, a $\rho(p(y), q)$ je naš Bhattacharyya koeficijent. I sada možemo napokon shvatiti kakvu taj koeficijent ima ulogu. Ako su vektori istoga smjera, kut je 0, a time je kosinus kuta 1; za kut 90 stupnjeva, kosinus će biti 0, dok će za suprotan smjer vektora biti -1. Iz toga je definirano što zapravo znači “veliki” Bhattacharyyin koeficijent. Što je bliži 1 (odnosno što je veći) to je par našeg cilnog histograma i ispitnog kandidata sličniji. Za zaključak slijedi sljedeće: da bismo našli lokaciju novog cilnog histograma (odnosno našeg modela na slici) želimo naći maksimum Bhattacharyyinog koeficijenta. Na sljedećoj slici (Slika 20.) se može vidjeti prikaz modela u kojem računamo maksimum te funkcije.



Slika 20. Traženje maksimuma funkcije Bhattacharyyinog koeficijenta, Dr. Mubarak Shah, <http://crcv.ucf.edu/courses/CAP5415/Fall2012/Lecture-11-MeanShiftTracking.pdf>

Kada je koeficijent veći, udaljenost između dva težinska histograma je manja. Ovo se može opsati i trivijalnom formulom (26).

$$d(y) = \sqrt{1 - \rho(y)} \quad (26)$$

Očito kada će $\rho(y)$ biti 1, tada će udaljenost $d(y)$ biti 0. Dakle tražit ćemo minimum funkcije udaljenosti dva težinska histograma. Za daljnju analizu ćemo napraviti aproksimaciju funkcije Bhattacharyyinog koeficijenta Taylorovim polinomom prvog

reda (preciznost nam nije sad najbitnija, a lakše je raditi s ovako niskim stupnjem). Dakle, izraz za općenit prikaz aproksimacije funkcije Taylorovim polinomom prvog reda opisan je u formuli (27).

$$f(x) \approx f(a) + f'(a) \cdot (x - a) \quad (27)$$

Kada to primjenimo na formulu (25) dobivamo formulu (28) uzimajući u obzir da je f ekvivalent ρ , x je ekvivalent x , dok je a ekvivalent y_0 koji predstavlja početnu procjenu mjesta našeg kandidata (mjesta gdje pretpostavljamo da je naš model).

$$\rho[p(y), q] \cong \rho[p(y_0), q] + \frac{1}{2} \sum_{i=1}^m p_u(y) \sqrt{\frac{q_u}{p_u(y_0)}} \quad (28)$$

Dakle, sada kada bismo u ovom obliku htjeli naći maksimum aproksimacije Bhattacharyyinog koeficijenta, želimo naći maksimum drugog pribrojnika u formuli

(28): $\frac{1}{2} \sum_{i=1}^m p_u(y) \sqrt{\frac{q_u}{p_u(y_0)}}$. Prvi pribrojnik u formuli (28): $\rho[p(y_0), q]$ je konstanta.

Sada ćemo u taj drugi pribrojnik ubaciti formulu (21a) preko varijable $p_u(y)$ jer ona označava vektor koji predstavlja težinski histogram kandidata, uz malo preciznije definiranu udaljenost kao argument kernelove funkcije. Ono što bismo trebali maksimizirati bi izgledalo ovako kao što je navedeno u formuli (29).

$$\frac{C}{2} \sum_{i=1}^{n_x} \left[\sum_{u=1}^m \delta[S(x_i) - u] \sqrt{\frac{q_u}{p_u(y_0)}} \right] k\left(\left\|\frac{y - x_i}{h}\right\|\right) \quad (29)$$

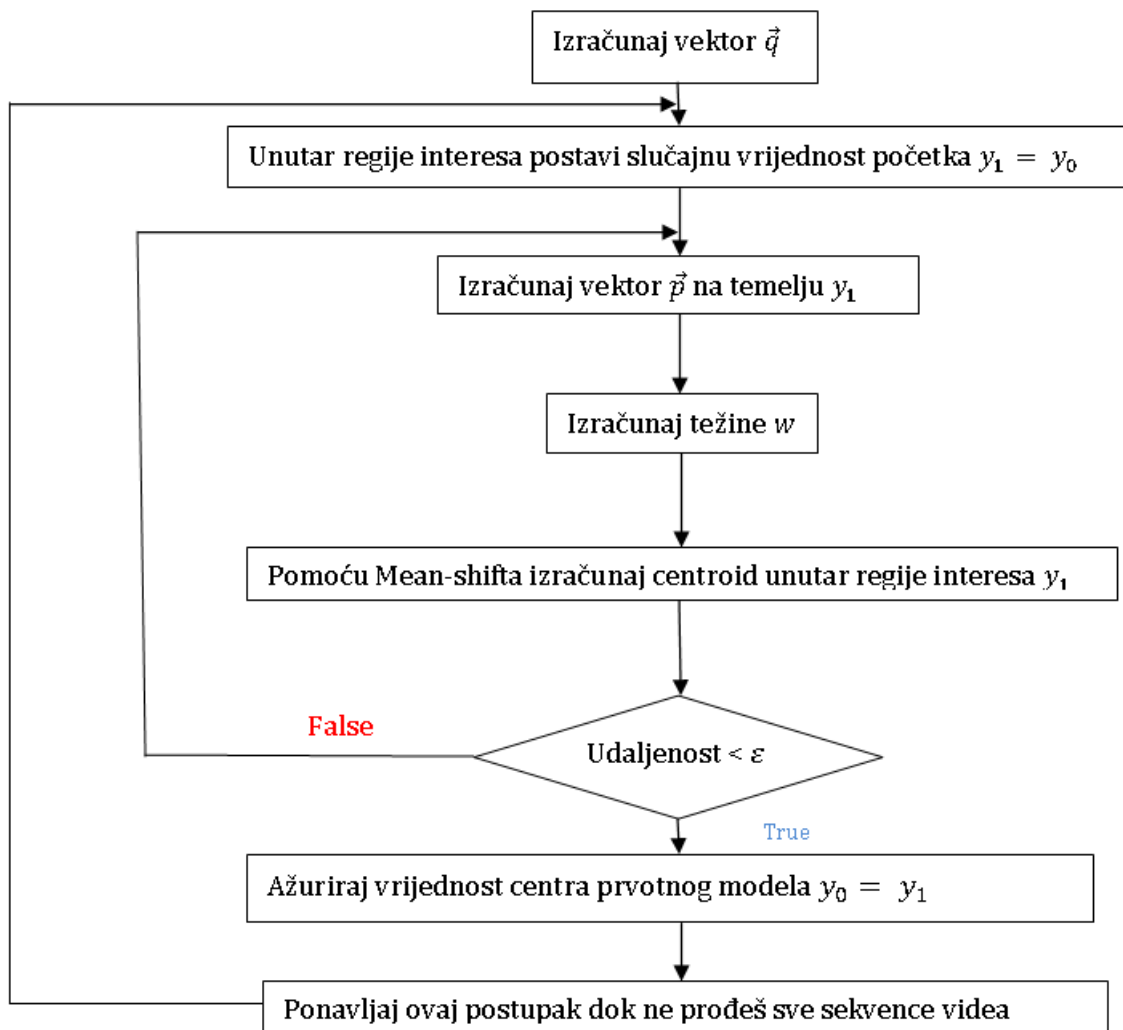
Ponovit ću još jednom što tu sve imamo. C kao što sam već bio rekao je normalizacijski faktor, $S(x_i)$ je intenzitet boje na poziciji (x_i) (odnosno funkcija koja kao rezultat vraća kojem indeks bina kojem pripada boja na toj koordinati), m je broj binova, n_x je broj okolnih koordinata (piksela) – i sada napokon dio koji nisam prije ovako precizno opisao, a to je dio unutar kernelove funkcije k : y je središte jezgre (kernela) što ima smisla jer onda udaljenost gledamo kao razliku između tog centra i koordinate x_i , dok je h kernelova širina (širina jezgre) koji sam prije spomenuo kao jedini parameter KDE. Ono što ovdje namjeravamo maksimizirati je dio formule (29) označen plavom bojom koji ćemo nadalje označavati kao w_i što predstavlja težinu. Iz ovoga slijedi zaključak da namjeravamo maksimizirati težine. Izraz za težinu je prikazan u formuli (30).

$$w_i(y_0) = \sum_{u=1}^m \delta[S(x_i) - u] \sqrt{\frac{q_u}{p_u(y_0)}}, \quad 0 < w_i(y_0) < 1 \quad (30)$$

I napokon dolazi krajnje pitanje, kako to napraviti? Pa upotrebom Mean-shifta, onakvog kakvog smo definirali još u formuli (2). Upotrijebit ćemo težine kako bismo

moгли napraviti Mean-shift. Želimo pronaći naš model na temelju doprinosa svakog piksela unutar prozora. Taj doprinos ovisi o težini, dok težina ovisi o distribucijama prvotnog modela i kandidata. Na taj način možemo naći sljedeću lokaciju našeg modela. Možemo ponovno pogledati formulu (2) da se prisjetimo oblika našeg Mean-shift algoritma koji koristi ovako definirane težine.

Sada sve ovo što smo rekli o usporedbi histograma možemo ukratko sažeti i vidjeti ugrubo kako bi izgledao algoritam. U teoriji kada bismo samo tražili maksimum funkcije Bhattacharyyinog koeficijenta imali bismo za svaku točku unutar regije interesa vrijednost između 0 i 1 te u tom najjednostavnijem slučaju našli za koju koordinatu y bismo dobili vektor p za koji bi p bio najveći u regiji interesa. Međutim koristimo način izračuna težina koji je pogodan za Mean-shift. Prije svega računamo vektor q koji predstavlja težinski histogram našeg modela. Zatim na samom početku napravimo početnu procjenu $y_1 = y_0$. Dakle, definiramo točku centra za kandidata. Nakon toga u toj točki izračunamo vektor p te iz njega možemo lako izračunati težine na temelju formule (30). Imali bismo inicijalnu točku i težine u okolnim pikselima (kao što smo u podatkovnom prostoru imali inicijalnu točku i podatkovne točke različito udaljene jedna od druge gdje su udaljenosti bile temelj izračuna težina između točaka). Sada na temelju formule (2) možemo izračunati y_1 , odnosno točku koja predstavlja najvjerojatnije područje našeg modela. U slučaju ako se više ne možemo pomaknuti za određeni ε , možemo završiti proces jer smo konvergirali (došli smo u novo središte našeg modela te možemo promijeniti vrijednost varijable y_0) i krenuti na sljedeću sekvencu i opet napraviti isti proces. U suprotnom ako smo se pomaknuli bili za vrijednost veću od ε onda prvotni model i dalje ostaje gdje je bio te nastavljamo Mean-shiftom tražiti eng. mode, ali u novoj regiji interesa gdje je centar mjesto u koje smo se pomaknuli. Cjeli ovaj opis algoritma se može vidjeti na slici 21.



Slika 21. – Prikaz slijednog dijagrama Mean-shift algoritma uz usporedbu histograma
 - Alper Yilmaz, University of Central Florida, Computer Vision Lab, -
http://crcv.ucf.edu/projects/MeanShift/MeanShift_ppt.pdf - slide 11

U teoriji, ovo je matematički potpuno točno. U praksi (izvedbi implementacije) te uz pomoć OpenCV biblioteke za Python koju koristim u ovom radu nije najprikladnije. Algoritam za primjenu ima puno nedostataka uzimajući u obzir kontekst. Ponajprije što se tiče djela računanja težina, te spajanje toga s gotovom Mean-shift funkcijom iz cv2 biblioteke nije moguće. Nadalje, imamo mogućnost koristiti dio algoritma, a to je dio usporedbe dva histograma unutar regije interesa. Dakle, usporedba histograma modela, te usporedba histograma u određenoj točki y (kandidat) unutar regije koja obuhvaća jednaku površinu kao model. Biblioteka nudi funkciju `cv2.compareHist(H1, H2, method)`, gdje se za metodu može postaviti `CV_COMP_BHATTACHARYYA`, ali tu dolazimo do problema što se tiče našeg videa.

Naš model je predstavljen vrlo malom površinom (uzimajući u obzir video utakmice Dinamo-Rijeka) gdje su igrači predstavljeni pravokutnikom kojem je širina u prosjeku 5, a visina 10, što je nekih 50 piksela. Regija interesa mora obuhvatiti 2-3 puta veće područje. U prosjeku bismo za svaki frame 50 puta računali Bhattacharyyin koeficijent za područje koje bi očito moglo sadržavati više sličnih kandidata modelu. Za rješenje bismo možda mogli uzeti ono koje je najbliže modelu, ali rezultat bi bio jako neprecizan. Prvi razlog je taj što bi naš histogram bio jako neprecizan uzimajući u obzir tako malen model. Morali bismo povećati raspone binova, ali time gubimo s druge strane specifičnost piksela, te bi time vrlo različiti pikseli mogli pripadati jednakim binovima i time bi usporedba histograma mogla napraviti krivu procjenu. S druge strane kada bi binovi bili manjih raspona teško bi bilo zaključiti koji je histogram sličniji kojem jer može doći do malih promjena u bojama piksela unutar regije interesa. Tim malim procjenama koeficijent koji promatramo će biti jako promjenjiv te može doći do lagane pogreške. Lako se može desiti da kandidat koji predstavlja model ima manju sličnost od nekog okolnog kandidata. Dakle, postoje dvije negativne stvari koje ne odgovaraju jedna drugoj. U videu su igrači prikazani malom površinom te metoda usporedbe histograma nam je malo problematična. Također, ovim načinom praktički ne bismo koristili Mean-shift zbog nepovezanosti ove dvije funkcije unutar biblioteke. Mjenjali bi središte ROI-a samo na temelju Bhattacharyyinog koeficijenta. Ovo bismo mogli reći da bi bila posebna metoda praćenja. Koristit ćemo nešto praktičnije i naprednije uz uporabu Mean-shifta.

3.2 Histogram backprojection

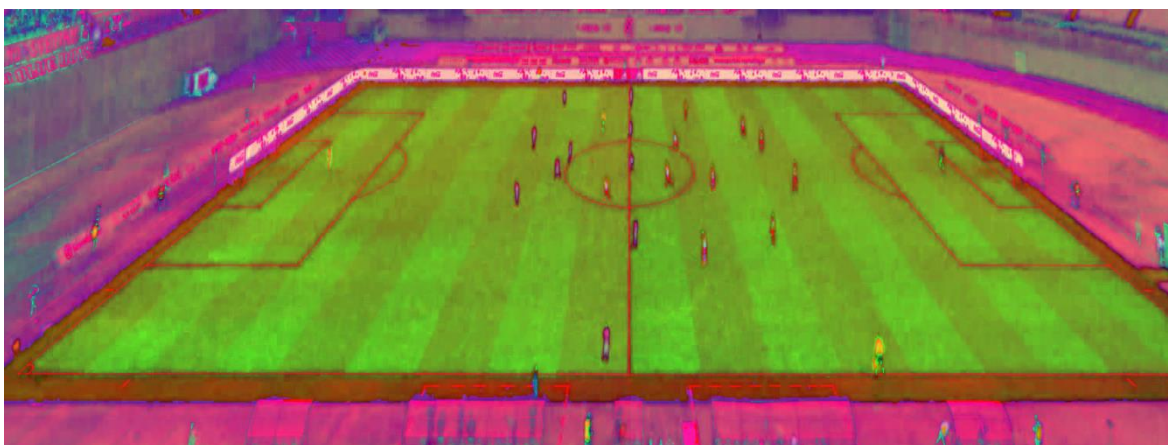
Problem ćemo probati još više pojednostaviti i olakšati za korisnika koji želi pratiti igrača. Radi lakšeg rada s objektima promatramo video utakice Dinamo-Split širine 3260 piksela, visine 570 piksela te duljine 7515 sekvenci uzimajući u obzir 25 fps. Korisnik će imati mogućnost označavanja više objekata unutar sekvence, a tako i njihovog daljnjeg praćenja. Prvi korak korisnika je označavanje modela u kojem god trenutku korisnik želi, a to će predstavljati ulaznu sliku – isti pravokutnik će predstavljati i regiju interesa zato što u njoj promatramo gdje se nalazi objekt. Za svaki model će se izraditi histogram, a nakon toga će se na temelju svakog histograma napraviti tzv. **histogram backprojection**. Ovo je najbitniji pojam koji ćemo koristiti unutar svoje implementacije, a o čemu se tu radi? Backprojection je pristup koji prikazuje koliko dobro pikseli određene slike pripadaju distribuciji piksela u modelu histograma. Jednostavnijim jezikom bismo rekli da računamo histogram neke određene značajke i onda ga koristimo za nalazak te značajke u slici (u ovom slučaju te značajke su konkretne boje). Kada bismo imali histogram igrača, postupak bi išao ovako. Nakon izrade histograma, prešli bismo na sljedeću sliku (frame) koji bi sadržavao pravokutnik u kojem je naša ispitna slika. U svakom pikselu ispitne slike $p(i, j)$ skupimo podatke o boji i nađemo pripadajući bin za taj piksel. Nakon toga pogledamo model histograma unutar tog bina i pročitamo njegovu vrijednost. Tu vrijednost spremimo u novu sliku (BackProjection). Dobro je i prije toga, nakon izrade histograma, normalizirati histogram. U statistici, vrijednosti spremljene u backprojectionu su vjerojatnosti da piksel u ispitnoj slici pripada modelu igrača bazirano na modelu histograma kojeg koristimo. Nakon izrade distribucije backprojectionom, pomoću Mean-shift algoritma pronalazimo najbliži eng. “mode” distribucije u “susjedstvu”.

Na slici 22. se vidi prikaz jedne sekvence utakmice Dinamo-Split u kojoj je korisnik označio određene igrače (primjetimo plave i crvene boje). Također je bitno napomenuti da sam zbog veličine samog videa skalirao sliku, u širini smanjio za 2 puta, a u visini ostavio kako je. Nakon toga želimo promijeniti sustav boja za lakši daljnji rad sa izradama histograma, backprojectiona, te samog praćenja. Kao što sam već prije spomenuo, to će doprinjeti otpornosti na promjene osvjetljenja objekta (ako dođe do nastanka sjene nad objektom).



Slika 22. Jedan frame iz videa Dinamo-Split, korisnik označio modele za praćenje

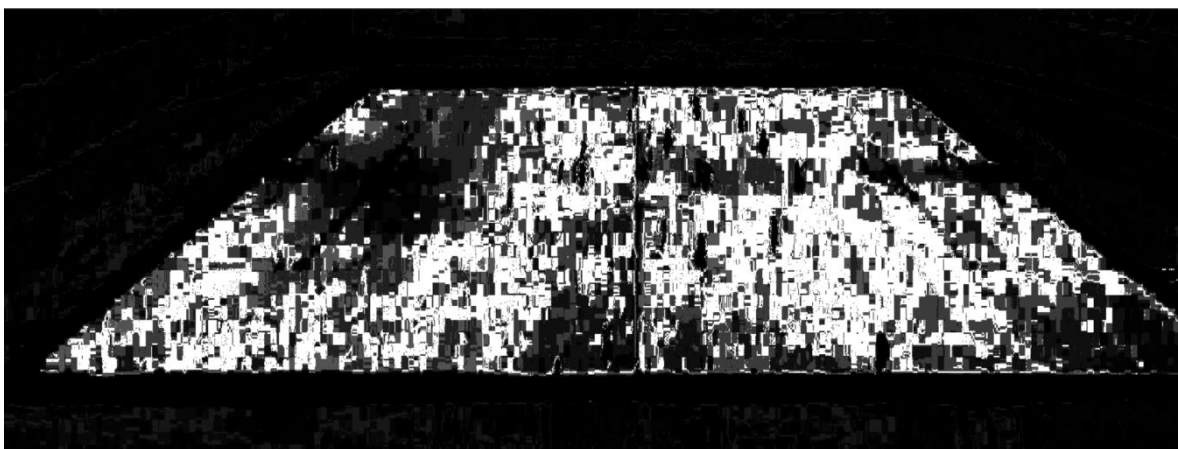
Na slici 23. Vidimo kako to izgleda nad konkretnom sekvencom.



Slika 23. Prelazak iz BGR u HSV sustav boja, video Dinamo-Split

Kada napravimo histogram nad modelom plavog igrača, te napravimo backprojection tog histograma dobivamo sljedeći rezultat kao novu sliku (distribuciju). Bitno je znati da bijela boja predstavlja veliku vjerojatnost da se igrač nalazi na tom mjestu. Nažalost, prvim ovim pothvatom ćemo se razočarati. Kao što vidimo na slici 24., gotovo sve oko našeg označenog objekta je velike vjerojatnosti da se tamo nalazi igrač, te će Mean-shift napraviti jako lošu stvar – pravokutnik će se kretati na sve strane, kako sekvence prolaze. Ali zašto? Kao i u prvom slučaju naš model je jako mal, te zbog nedostatka podataka, histogram ne opisuje vjerno početni objekt. Dosta boja koje se nalaze unutar modela su posvuda po terenu. To nam ne odgovara te želimo reducirati skup boja koji razmatramo unutar modela. Trebamo nekako znati koje boje

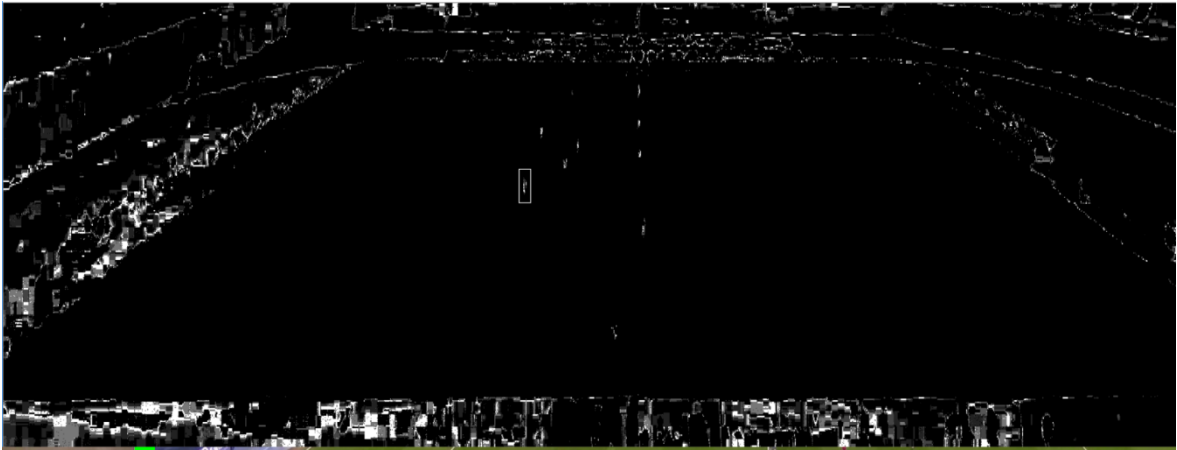
je naš igrač kojega smo označili. U našem programu, potpuno ćemo se posvetiti ovom konkretnom videu te bojama njegovih igrača. Ovime smo ušli u podskup videa nogometnih utakmica u kojima je igrač plave ili crvene boje, ali to nam olakšava stvar za daljnje razmatranje. Također smo dobili gotovu datoteku u kojoj se nalaze trajektorije igrača koje su dobivene detekcijom (njen format ću još objasniti ukratko u poglavlju o implementaciji i korištenju). One su također ručno popravljene te su svi igrači dobili svoj ID. Iz te datoteke ćemo iskoristiti taj ID za sam početak kako bismo znali o kojoj boji igrača se radi. Igrači u videu od 1-10 su plave boje, dok su igrači od 13-22 crvene boje. Prije izrade histograma ispitujem s kojim pravokutnikom u toj sekvenci se presiječe moj pravokutnik te time dolazim do ID, pa tako i do boje.



Slika 24. Backprojection version 1. – Dinamo-Split

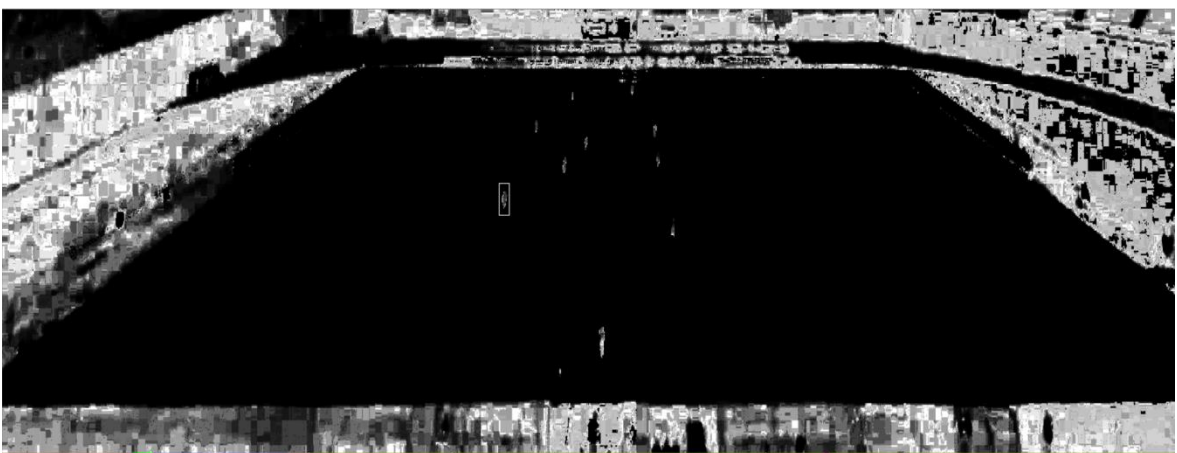
Kako iskoristiti boju? Uzimajući u obzir Pythonovu funkciju iz OpenCV biblioteke: `cv2.calcHist(images, channels, mask, histSize, ranges)` koja uzima kao argumente sliku, kanal iz sustava boja, masku koja uzima u obzir samo neke boje, veličinu histograma i domet vrijednosti kanala koje želimo mjeriti. Prvi puta smo uzeli kao sliku, naš pravokutnik (model), kao kanal smo uzeli Hue komponentu [0], domet smo uzeli 0-179 (dakle, sve Hue vrijednosti), veličina histograma nam je 180 binova (ovdje je vjerojatno prva greška jer smo za 1 vrijednost uzeli po 1 bin, ali to nećemo mjenjat jer su ovdje razlike vrlo male pošto nam je model premalen, pa zbog toga ne bi trebalo biti niti neke razlike u backprojectionu) te nismo uzeli nikakvu masku. Sada ćemo postaviti masku u raspon plave/crvene boje kako bismo samo nju uzeli u obzir pri razvrstavanju u binove. Naša funkcija bi sada izgledala ovako:
`roi_hist = cv2.calcHist([hsv_roi],[0], mask,[180],[0,179])`. Kako bismo izračunali distribuciju, odnosno backprojection slike, moramo koristiti sljedeću funkciju:

c2.calcBackProject(images, channels, hist, ranges, scale). Kao argumente redom zapisujemo sliku iz koje radimo backprojection, kanale koje koristimo kako bismo izračunali backprojection (broj kanala mora biti kompatibilan s dimenzionanošću histograma), zatim sam histogram, granice binova koje koristimo [0 – 179] I na kraju krajeva factor skaliranja koji nećemo koristiti (odnosno ostavit ćemo ga kao 1). Sada smo dobili distribuciju kao na slici 25.



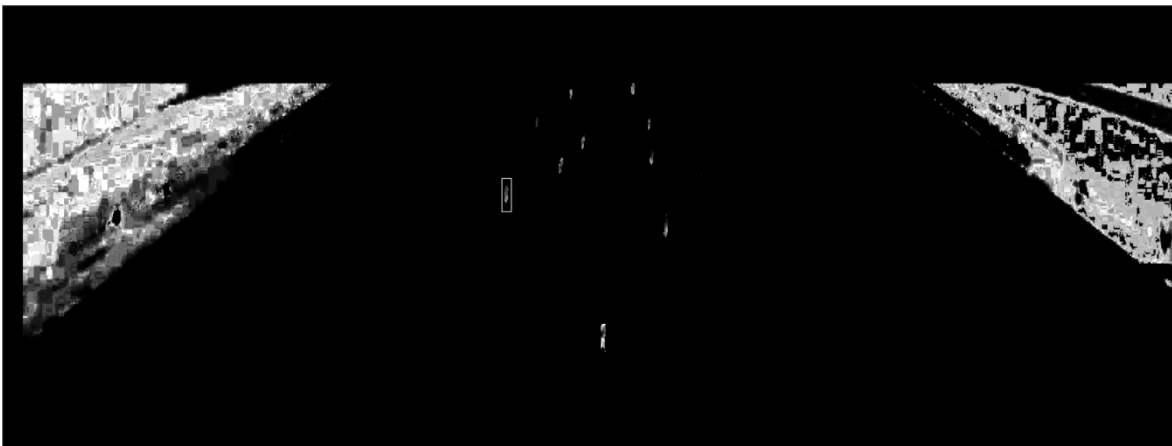
Slika 25. Backprojection version 2. – Dinamo-Split

Sada smo dobili mnogo bolji rezultat jer smo se riješili cijelog terena koji nam je smetao jer se uvijek dio terena nalazio uz igrača unutar pravokutnika. No, uz sve ovo što imamo, mi možemo rezultat još više poboljšati (trenutno možemo primjetiti da na trenutke nestane površina bijele boje na mjestu gdje se nalazi igrač u određenim sekvencama zbog kvalitete videa i promjene perspective igrača). Pošto je model koji imam premalen, možemo uzeti veći model, primjerice veličinu cijele sekvence s maskom za igrača određene boje. Ovime dobivamo rezultat kao na slici 26.



Slika 26. Backprojection version 3. – Dinamo-Split

Možda na prvi pogled ne izgleda bolje, ali mnogo je bolje jer gotovo u svakom trenutku vidimo vjerojatnosti za igrače konkretne boje na svojim pozicijama te Mean-shift može teško pogriješiti (osim u specijalnom slučaju kada se spoje 2 ili više igrača iste boje te pravokutnik ode u smjeru igrača kojeg do sada nismo pratili). No, sada imamo i problem na rubovima pošto i rubovi obuhvaćaju velik postotak boje iz maske (u histogramu). Igrač prilikom dodira s rubom izgubi pravokutnik sa sebe, te on odluta i vrlo brzo se kreće po rubnim dijelovima. Kako bismo ovo riješili, izvadio sam dio sekvence koji uzima samo teren u obzir. Presjekao sam gornji i donji dio, te ga stavio u praznu (crnu) sliku jednake veličine kao sekvenca (pomoću polja koje čine same nule). Lijevi i desni dio nije bilo potrebno riješiti jer je tamo manja vjerojatnost igrača u backprojectionu, ali je i manja vjerojatnost da će igrač dolaziti na to mjesto (osim u slučaju kornera). Rezultat bi izgledao kao na slici 27.



Slika 27. Backprojection version 4. – Dinamo-Split

Nadalje, koristimo Mean-shift algoritam pomoću OpenCV funkcije `cv2.meanShift(probImage, window, criteria)`. Kao argumente uzima redom backprojection sliku, prozor (regiju interesa, pravokutnik koji okružuje model, te kriterij za zaustavljanje Mean-shift algoritma. Procedura će se ponavljati sve dok određen broj iteracija nije odrađen (`criteria.maxCount`), te sve dok se centar prozora ne pomakne za broj piksela koji je manji od granice (`criteria.epsilon`). Ovo se gotovo svaki puta koristi 10 iteracija kao `maxCount` te 1 piksel kao `epsilon`.

Napokon, sada kada znamo kako izgleda jedan izračun histograma te jedan izračun backprojectiona, možemo napraviti eng. multi-tracker, odnosno za svaki označeni

model unutar jedne sekvence videa možemo napraviti njegov histogram (moram napomenuti da će to bitijedini izrađen histogram za taj model do kraja videa, dakle nadalje nakon ove sekvence se više ne izrađuju novi histogrami za isti model, jer histogram je statički i odnosi se samo na sekvencu koju je korisnik odabrao). Nakon toga, za svaki histogram ćemo u svakoj novoj sekvenci računati distribuciju te na temelju svake te distribucije u svakoj novoj sekvenci ćemo na svakoj primjeniti Mean-shift algoritam koji će pomaknuti pravokutnik koji okružuje model u određenom smjeru. Dakle, u svakoj sekvenci ćemo imati toliko pomaka pravokutnika koliko smo napravili modela. Ovime smo napravili funkcionalni multi-tracker. Na slici 28. se jasno mogu vidjeti više označenih modela. Funkcionalnost praćenja treba se pogledati korištenjem implementacije.

4. Implementacija i korištenje

Htio bih još malo više reći, odnosno dokumentirati što sam koristio, i u kakvom je to formatu. Također, napomenuo bih kako se koristi kod koji se dobije uz ovaj rad.

Kao što sam već bio napomenuo, konkretna implementacija radi uz točno ovaj video konkretne širine, visine i boja igrača uz dobivenu datoteku ručno izrađenih trajektorija za ispravljanje lošeg praćenja. Mogli bismo reći da je to premalen skup videa na kojima ovo funkcionira, ali htio sam pokazati kako uz znane informacije o videu možemo poboljšati praćenje skoro do idealnosti. Kada ne bismo uzimali dosta bitne informacije o videu u obzir, došlo bi do raznih problema. Što se tiče veličine



Slika 28. – Trenutak utakmice kada su modeli već definirani , dok nad njima radi multi-tracker – Dinamo-Split

video, mogli smo još nekako napraviti generički eng. “resize”, ali nisam htio previše komplicirati za zadani video te moramo uzeti u obzir različitu rezoluciju, ali i veličinu ekrana. Nadalje, što se tiče same izrade histograma, histogram bez maske će u većini slučajeva biti loš histogram, kao što sam i pokazao u poglavlju 3, pogotovo uz slučaj ovako malog modela. Jedan mogući generički način je taj da korisnik upiše konkretne boje igrača (ako si postavimo da govorimo o samo osnovnim bojama dresova). No, i tu nailazimo na problem. Kakvu masku napraviti ako je boja dresa bijela, ili crna? Tu dolazi do problema jer bijela i crna zapravo nisu boje HSV sustava boja, tj. točnije njene Hue komponente, te možemo odrediti konkretnu masku jer bilo koji Hue raspon može biti unutar toga, te tu dolazimo do većeg problema.

Da ponovim, kod je izrađen u Pythonu 2.7.9. uz korištenje biblioteke OpenCV 3.2.0 iz koje sam uglavnom koristio funkcije za “Analizu pokreta i praćenje objekata”.

Postavio sam na početku ArgParser kako bismo u naredbenom retku mogli odabrati željeni video s diska unutar kojeg želimo pratiti igrače, ali bi on naravno trebao spadati u naš podskup videa i imati točno isti format datoteke s trajektorijama koji koristimo. Pa ako nam se u direktoriju u koji smo se pozicionirali nalazi i video i Python datoteka, u naredbenom retku bismo to pozvali u sljedećem obliku:

`>python datoteka.py -v vid.mp4` ili `>python datoteka.py --video vid.mp4`. Dakle, imamo dugi i kratki oblik opcije za postavljanje videa (`-v` i `--video`).

Nakon nekoliko trenutaka video započinje i korisnik ima mogućnost zaustaviti video kako bi označio objekte. Napomena: korisnik je u mogućnosti ovo učiniti samo jedanputa, ali mu postoji funkcionalnost da u tom trenutku dok je video u tom stanju pauze može ispraviti svoje oznake.

Kako bi zaustavio video kako bi mogao označiti modele, korisnik mora kliknuti na tipku “**r**”. U tom trenutku je u mogućnosti označavati modele klikom i povlačenjem miša oko modela. Default boja za označavanje je plava (koristi se za označavanje plavih igrača). Kako bismo odabrali drugu boju moramo kliknuti na tipku “**c**”, a to je crvena boja (koristi se za označavanje crvenih igrača). Ova funkcionalnost se nudi korisniku ako želi. Nadalje, tipkom “**d**”, brišemo sve dosadašnje modele koje smo označili. Kada korisnik definira sve modele koje je htio, treba kliknuti na tipku “**e**”, kako bi opet pokrenuo video s definiranim modelima. Tipkom “**q**” završavamo video.

r	Video se zaustavlja, te ulazi u stanje za označavanje modela od strane korisnika
e	Kada korisnik definira sve modele, i želi započeti praćenje tih istih modela
d	Korisnik briše sve dosadašnje modele u stanju za označavanje modela
c	Korisnik mjenja boju za izradu oznake modela (plava -> crvena, crvena -> plava)
q	Korisnik prekida video

Tablica 1. – Korisničke upute

Još je potrebno opisati format datoteke s trajektorijama (koje su naknadno ručno popravljene). Na samom početku tekstualne datoteke se nalaze oznake igrača jednog i drugog tima – indeks i boja igrača. Nakon toga slijede trajektorije u sjedećem formatu – broj sekvence, indeks igrača, minimalna y vrijednost, maksimalna y vrijednost, minimalna x vrijednost, maksimalna x vrijednost, te “dummy” vrijednosti na koje se ne obaziremo. Pomoću minimalnih i maksimalnih vrijednosti možemo doći do bilo koje koordinate vrha pravokutnika, pa tako i do središnje točke pravokutnika. Na samom početku kako bismo znali koje je boje naš igrač, tražimo presjek onoga što smo definirali modelom s određenim pravokutnikom u toj istoj sekvenci u trajektorijama. Zapravo, ono što tražimo je najbliža središnja točka od tih 23 pravokutnika, središnjoj točki našeg pravokutnika. Pronalazak nije spor pošto koristimo rječnik koji koristi raspršeno adresiranje gdje je prosječno vrijeme pristupa konstantno.

Kao posljednje, trebam napomenuti da sam napravio i kod koji popravljiva praćenje objekata u svakih 200 sekvenci (8 sekundi) u slučaju da dođe do pogreške, odnosno praćenja krivog objekta, ili ako ostane na istom mjestu izgubivši ikakve veće vjerojatnosti od 0 unutar regije interesa. Time što sam zapamtio indeks igrača za pravokutnik, mogu vidjeti da li se slažu taj pravokutnik te pravokutnik iz datoteke s trajektorijama. Na koji način vidim da li se slažu? Na temelju udaljenosti između središnjih točaka. Zbog toga sam postavio epsilon udaljenost koju bi prelaskom mogli reći da smo sigurni da više ne pratimo model koji smo definirali. Epsilon vrijednost sam “hardkodirao” te ona iznosi 20 piksela. Opet kažem, vrijedi uz priloženi video, ali i podskup takvih videa. Isto tako mogli bismo se zapitati da li je provjera svakih 8 sekundi možda previse, ili premalo? Ne želimo provjeravati svaku sekvencu, jer bi se cijeli proces praćenja jako usporio. Svaku iteraciju (pomak sekvence za 1) bismo

morali raditi provjere udaljenosti što ne želimo. Izmjene sekvenci bi bile prespore (iako to ne mora značiti da to možda i ne želimo – možda baš želimo što sporije analizirati događaje na terenu). S druge strane ne želimo raditi provjere u što većim intervalima jer što ako nakon same provjere “izgubimo” objekt. Želimo što duže vremena pratiti originalni model. Isto tako, želimo da korisnik vidi kada je došlo do greške, pa je u ovom slučaju vremenski raspon 8 sekundi u redu.

5. Ostale metode praćenja objekata

Iako smo se u radu bazirali na algoritam Mean-shift i metode obrade slike za dobivanje distribucije kao ulaz za taj algoritam, postoje mnoge druge metode i algoritmi za praćenje objekata. Naprimjer, što ako je kamera postavljena pod jako malim kutom? Da li je Meanshift i dalje najbolji algoritam za praćenje igrača? Odgovor je, naravno, ne. Što ako na početku označimo naš model na bližem rubu terena gdje je pravokutnik dosta velik jer u tom slučaju igrač je najveće veličine. Ukoliko se igrač pomiče prema drugom kraju terena postaje sve manji, te Mean-shift jako teško prati. Igrači postaju male bijele točke u distribuciji unutar većeg pravokutnika. Pravokutnik lako može ostati na mjestu ili na kraju krajeva pratiti drugog igrača. Još gora situacija je ako napravimo suprotnu stvar, ako napravimo mali pravokutnik te se igrač približava prema kameri. Mean-shift u ovim slučajevima očito nije najbolji algoritam. Nije nam poželjna jednaka veličina pravokutnika. Radi toga je nastao **Cam-shift** algoritam (eng. Continously adaptive Mean-shift). Pravokutnik se treba prilagoditi ovisno o veličini i rotaciji označenog objekta. Rješenje je isto došlo iz OpenCV laboratorija. Kako funkcionira? Prvo se primjenjuje Mean-shift koji je dobro opisan u poglavlju 2. Nakon što konvergira, dakle nakon što Mean-shift vektor postane 0, odnosno kada smo došli do objekta na temelju distribucije, ažurira se veličina pravokutnika. Također, računa se najbolja elipsa koja bi okruživala taj objekt, te na temelju nje se pravokutnik rotira (optimalna rotacija). U OpenCV biblioteci, postupak je identičan kao kod MeanShifta, ali se osim vrha gornjeg desnog vrha, širine i visine vraća i orijentacija. Orijentacija u našem slučaju nije ni toliko bitna pošto se igrači ne okreću. Kako zapravo to uspijeva raditi, odnosno kako se shvaća gdje su granice objekta i pod kojim je nagibom? Izvodi se suma vjerojatnosti nakon izrade

backprojectiona unutar regije interesa, te ukoliko je velika (veća od nekog postotka u odnosu na neku najecu moguću vjerojatnost) tada će se regija interesa proširiti. U našem podskupu videa, Cam-shift nije toliko potreban jer veličine regija interesa se ne razlikuju išavši s prozorom od gornjeg do donjeg kraja terena, te nam nije nužan toliko za bolji rad, iako može poslužiti za mali postotak slučajeva kada bi zbog toga nastala greška. Inače, ukoliko i dođe do ove situacije kada je kamera postavljena pod dosta niskim nagibom, problem će postati i prekrivanje daljih igrača zbog onih bližih, te lako može doći do pogreške, ali takvi videi su izvan skupa videa koje smo mi obradili te to ostavljam za kasniji rad.

Još za kraj bih htio ukratko opisati dosta zanimljivu metodu za praćenje objekata drugačije vrste od dosadašnjih, a to je Kalmanovo filtriranje, algoritam obrade signala koji se koristi za predviđanje lokacije objekta koji se giba u smjeru bazirano na prošloj informaciji pokreta u prošloj iteraciji. Dosta je zastupljen, ali za razliku od Mean-shifta i Cam-shifta ovdje ne postoji lociranje maksimuma funkcije gustoće. Kalmanov filter je optimalni rekurzivni algoritam obrade podataka. Sastoji se od dvije faze: predviđanje i ispravljanje. Prva faza je predikcija sljedećeg stanja (položaja modela u sljedećoj iteraciji) koristeći trenutni skup zapažanja i ažuriranje trenutnog skupa mjera za predviđanje. Druga faza ažurira pak predviđene vrijednosti i daje mnogo bolju aproksimaciju sljedećeg stanja. Pokušava postići ravnotežu između predviđenih vrijednosti i šuma. Postoje i težine, a njihove vrijednosti su određene modeliranjem jednadžbi stanja. U radu "Izvedba usporedbe Kalmanovog filtra i Mean-shift algoritma za praćenje objekata" (<http://www.mecs-press.org/ijieeb/ijieeb-v5-n5/IJIEEB-V5-N5-3.pdf>) autori pokazuje kakve su performanse ova dva algoritma, te kada je bolje koristiti Kalmanov filter. Pokazali su da je Kalmanov filter mnogo bolji za uporabu u videima gdje su atmosferski uvjeti puni šuma (kišovito i maglovito vrijeme), što je vrlo dobro za nogometne utakmice koje su snimljene pod tim uvjetima. Dokazali su da i pri običnom translacijskom kretanju objekata, Kalmanov filter je inicijalno mnogo brži te je manja pogreška tijekom praćenja (prosječno kvadratno odstupanje – sumiraju se kvadrati razlika prave pozicije pravokutnika i izračunate pozicije algoritmom). Još su imali jedno zapažanje: pri jako velikim brzinama (gravitacijska sila koja djeluje na lopticu koja pada s kreveta) Mean-shift se pokazao kao neuspješan algoritam, dok je Kalman i dalje funkcionirao uspješno. Na slici 29. vidimo kako na slijedu slika bez šuma Mean-shift algoritam tijekom pada loptice s kreveta griješi, dok

se kod Kalmanovog filtra i dalje zadržava na objektu bez obzira na povećanu brzinu što je zbog drugačije vrste algoritma praćenja.



Figure 2. MSA detection of ball



Figure 3. Kalman detection of ball

Slika 29. – prikaz slijeda slika na kojima koristimo Mean-shift i Kalmanov filter te označen model - <http://www.mecs-press.org/ijieeb/ijieeb-v5-n5/IJIEEB-V5-N5-3.pdf> - Ravi Kumar Jatoth

6. Zaključak

Praćenje objekata je dobilo vrlo značajnu pozornost zadnjih godina. Generalno, to je kombinacija obrade računalnih i video slika, raspoznavanja uzoraka te umjetne inteligencije. Sva ova područja zajedno su povezana za detekciju objekata koji se kreću (odnosno praćenje) na slikama (izvađenih iz videa). Objekt se označava, analiziraju se karakteristike lokacije te se zatim izvršava praćenje u stvarnom vremenu. Detaljni redoslijed ovih operacija može biti različit za različite algoritme.

U ovom radu je detaljno opisan Mean-shift algoritam (koji koristi znanja iz područja obrade video slika) te njegova primjena u praćenju objekta na slijedu slika (odnosno video sekvenci). Algoritam koristi podatak o bojama objekta, koje se modeliraju histogramom. Nakon toga se računa distribucija, funkcija koja opisuje vjerojatnost položaja objekta na tom mjestu. Nakon toga se obavlja Mean-shift, odnosno određuje se lokalni ekstrem, tj. najvjerojatniji položaj našeg objekta. Kako bismo došli do ekstrema, više puta izvršavamo pomak za Mean-shift vektor. Postupno se pomičemo u smjeru gradijenta distribucije. Na kraju konvergiramo u lokalnom maksimumu. Dokazali smo da je gradijent distribucije jednak Mean-shift vektoru, te smo time obranili i njegovo korištenje u ovoj situaciji. Govorio sam i o mnogim problemima pri korištenju Mean-shifta, govoreći o kontekstu u kojem ga koristimo, te kako to popraviti uz, naravno, nove informacije o videu na kojem radimo praćenje. Na temelju tog novog znanja o videu sam napravio uspješan multi-tracker.

Uz ovu metodu, vrlo slična metoda je Cam-shift algoritam kojega sam ukratko opisao i koji je nadogradnja na Mean-shift, odnosno funkcionira gotovo na isti način osim što izmjenjuje veličinu prozora i njegovu rotaciju ovisno o objektu. Postoji još jako mnogo metoda koje uspješno obavljaju praćenje, ali ja sam se orijentirao na ovo takozvano Kernelovo praćenje, gdje se koristi jezgrena funkcija pri izradi distribucije. Postoji još "praćenje točaka" koje se dijeli na determinističke i statističke metode, praćenje silueta koje se također dijeli na podskupove raznih metoda. Mean-shift kao algoritam iz područja obrade video slika je bio jedan od ranijih metoda, te je ovaj rad možda jedan dobar uvod u takva područja koja rade s praćenjem objekata.

7. LITERATURA

http://web.eecs.umich.edu/~silvio/teaching/EECS598/papers/kernel_based_object_tracking.pdf

Comaniciu et al

http://crcv.ucf.edu/projects/MeanShift/MeanShift_ppt.pdf

Alper Yilmaz

<http://www.cse.psu.edu/~rtc12/CSE598G/introMeanShift.pdf> - Afshin

Deghan

<https://www.slideshare.net/sandtouch/meanshift-tracking-presentation> -

Leow Wee Kheng – Department of Computer Science, National University of Singapore

[http://crcv.ucf.edu/courses/CAP5415/Fall2012/Lecture-11-](http://crcv.ucf.edu/courses/CAP5415/Fall2012/Lecture-11-MeanShiftTracking.pdf)

[MeanShiftTracking.pdf](http://crcv.ucf.edu/courses/CAP5415/Fall2012/Lecture-11-MeanShiftTracking.pdf) - Dr. Mubarak Shah, Fall 2012., Center for Research in Computer Vision, University of Central Florida

https://en.wikipedia.org/wiki/Mean_shift - Wikipedia, The Free

Encyclopedia, Date of last revision: 10.03.2017., 03:43 pm

https://en.wikipedia.org/wiki/Probability_distribution - Wikipedia, The Free Encyclopedia

https://en.wikipedia.org/wiki/Nonparametric_statistics - Wikipedia, The Free Encyclopedia

https://en.wikipedia.org/wiki/Probability_density_function - Wikipedia, The Free Encyclopedia

[https://en.wikipedia.org/wiki/Kernel_\(statistics\)](https://en.wikipedia.org/wiki/Kernel_(statistics)) - Wikipedia, The Free Encyclopedia

http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/TUZEL1/MeanShift.pdf - University of Edinburgh, Bob Fisher

http://home.ku.edu.tr/mehyilmaz/public_html/mean-shift/00400568.pdf - Y. Cheng 08.08.1995.

http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_calculation/histogram_calculation.html - OpenCV 2.4.13.2 Documentation

http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_comparison/histogram_comparison.html - OpenCV 2.14.13.2 Documentation

http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/back_projection/back_projection.html - OpenCV 2.14.13.2 Documentation

https://en.wikipedia.org/wiki/Kernel_density_estimation - Wikipedia, The Free Encyclopedia

http://docs.opencv.org/3.2.0/db/df8/tutorial_py_meanshift.html - OpenCV 3.2.0 Open Source Computer Vision

<http://www.mecs-press.org/ijieeb/ijieeb-v5-n5/IJIEEB-V5-N5-3.pdf> -

Published Online November 2013 in MECS (<http://www.mecs-press.org/>), Ravi Kumar Jatoth, Sampad Shubhra, Ejaz Al

8. Naslov, sažetak i ključne riječi

Naslov

Praćenje igrača u snimkama nogometnih utakmica

Sažetak

Danas postoje razne metode praćenja objekata koje na razne načine mogu brzo i efikasno pomoći treneru i stručnom stožeru analizirati igrače unutar utakmice na temelju čega mogu doći do raznih strategija kako igrati protiv drugih timova, primjerice koji igrači bi bili prikladni za igru, koji ne, te sam način igre. Sve se to dobiva na temelju puta kojeg prolazi igrač, prostora u kojem se kreće te svega ostalog što se daje zaključiti iz praćenja igrača. U ovom radu opisuje se klasično praćenje objekata koristeći algoritam Mean-shift. Kako bismo primijenili Mean-shift moramo koristiti neku metodu obrade slike kako bismo dobili dobru distribuciju s kojom radi Mean-shift. Modelira se histogram na temelju kojega se opisuje vjerojatnost da li se igrač nalazi na tom mjestu u terenu. Algoritam je testiran na sekvencama nogometnih utakmica, a radi boljeg praćenja i pokazivanja što sve poboljšava praćenje, implementacija je primjenjiva na specifičan skup video-sekvenci.

Ključne riječi: Mean-shift, algoritam, distribucija, procjena gustoće Kernelom, praćenje, objekti, backprojection, regija interesa, histogram, boje, HSV sustav boja, RGB sustav boja, jezgrena funkcija, funkcija gustoće vjerojatnosti, neparametarska statistika, video, sekvenca, usporedba histograma, centar mase, Mean-shift vektor, Cam-shift, lokalni optimum, konvergencija, cluster, svojstveni vektor, svojstveni prostor, košara, Bhattacharyya koeficijent, OpenCV biblioteka, pikseli, model, kandidat, nogometna utakmica.

Title

Tracking of players in football game video

Abstract

Today there are many different object tracking methods which can be helpful to coach and his expert team for efficient and fast analysis of players in the game. They can decide which strategy to use versus opponent teams, for instance, which players would be suitable for playing, which not, or how to play the whole game generally. We get all of that based on player's path, space where he moves and all other things which can be deducted from players tracking. In this thesis is described classic object tracking using Mean-shift algorithm. In order to apply Mean-shift, we must use some specific image processing method, so we could get good density distribution, which Mean-shift use as input. We are modelling histogram which is used for describing probability, is player placed on a specific dot in football field. Mean-shift is tested on video-sequences of football games. For better tracking and indicating what does improve tracking, given implementation is effective on specific set of video-sequences.

Keywords: Mean-shift, algorithm, density distribution, Kernel-density estimation, tracking, objects, backprojection, region of interest, histogram, colors, HSV color space, RGB color space, kernel function, probability density function, nonparametric statistics, video, sequence, frame, histogram comparison, centroid, Mean-shift vector, Cam-shift, mode, convergence, cluster, feature vector, feature space, bin, Bhattacharyya coefficient, OpenCV library, pixels, model, candidate, football game.