***UNIVERZITET U SARAJEVU***

***ELEKTROTEHNIČKI FAKULTET***

***AUTOMATIKA I ELEKTRONIKA***

***Predmet: Mobilna robotika***

# Seminarski rad

*Detekcija ivica, linija i krugova na slici – robotska vizija*

***Profesor: prof. Dr. Jasmin Velagić***

S***TUDENT: Emina Hasanović***

***Sarajevo, Septembar 2019.***

Sadržaj

# 1. Uvod

U ovom seminarskom radu bit će prikazana detekcija ivica, linija i krugova na slici. Pomenuti problem bit će urađen u ROS-u *(eng. Robotic Operating System)* o kojem će detaljnije biti u nastavku ovog dokumenta. Ovaj problem je jedan od početnih problema u robotskoj viziji, jako je aktuelan i zanimljiv. Rad napisanog algoritma bit će demonstriran koristeći web kameru laptopa.

Algoritmi koji će biti korišteni su: Canny Edge detection, Hough Transform for Line and Circle detection (Canny-ev algoritam za detekciju ivica I Hafova transformacija za detekciju linija I krugova).

Kriteriji funkcionalnosti algoritma su dobra detekcija, dobra lokalizacija i minimalan višestruki odziv. Dobra detekcija kaže da bi algoritam trebao obilježiti što je moguće više stvarnih rubova koji zaista postoje na posmatranoj slici, dobra lokalizacija je uvjet da su obilježeni rubovi što je moguće manje udaljeni od stvarnih rubova na slici, a minimalan višestruki odziv govori da bi rub trebao biti obilježen najviše jednom sa što manjom prisutnošću šuma.

# 2. Robotska vizija

Sa stajališta robotike, područja računarske ili robotske vizije istražuju oblasti računarskog predstavljanja okoline na temelju svjetlosnih signala i proučavaju tehnike umjetne inteligencije koje obavljaju zadatke zaključivanja ili planiranja na temelju dobivenog prikaza okoline.

Glavni cilj sistema robotske vizije je oponašati sposobnosti i značajke ljudskog vizualnog sistema.

Kako je vizija najmoćniji način opažanja okoline slijedi da vizualni robotski sistemi osiguravaju veliki broj informacija o robotskom okruženju na temelju kojih se može ostvariti inteligentna interakcija u dinamičkoj sredini.

Pri tome treba voditi računa da je obrada podataka i gradnja mape okruženja na temelju vizualnih informacija značajno složenija od istih sa nevizualnim senzorima.

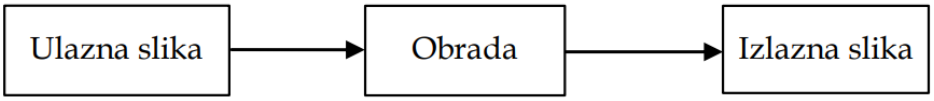
**Temeljni procesi u robotskoj viziji: obrada i analiza slike**.

## 2.1. Obrada i analiza slike

Obrada i analiza slike predstavljaju važne procese kojima se transformira slika u prihvatljiv oblik i izdvajaju, odnosno prepoznaju odgovarajuće značajke slike koje se mogu koristiti u različitim procesima unutar navigacijskog sistema mobilnog robota.

Digitalna obrada slike je proces podvrgavanja numeričkih reprezentacija objekata u slici seriji operacija s ciljem postizanja željenog rezultata.

Ulaz i izlaz u procesu obrade slike predstavljaju sliku, dok se sam proces obrade sastoji od poboljšavanja i obnavljanja slike – slika 2.1..




Slika 2.1. Blok dijagram obrade slike

Digitalna obrada slike obuhvaća sljedeće procese:

- **Operacije na slici** (konvolucija, Fourierova transformacija, Z transformacija,...).
- Transformacije slike (linearne, ortogonalne, diskretne).
- **Metode poboljšanja slike** (operacije na tački, prostorne operacije, upotreba transformacija, tehnike pseudokoloriranja, filtriranje – linearno i homomorfno,...).
- **Obnavljanje slike** (modeli degradacije slike, Inverzni i Wienerov filter, pseudoinverzija matrica, ...).
- **Rekonstrukciju slike iz projekcija** (Radonova transformacija, operator povratne projekcije, projekcijski teorem, inverzna Radonova transformacija, Hilbertova transformacija, Fourierova metoda rekonstrukcije, ...).
- **Kompresiju slike** koja pokušava smanjiti broj bitova potrebnih za pohranu slike bez ili sa gubitkom informacija.

Izlaz iz sistema za analizu nije slika, već numerički ili simbolički opis njenog sadržaja.

**U sistemu za analizu slike postoje tri grupe operacija izdvajanje značajki, segmentacija i opis scene.**

Postupci u procesu analize slike:

- **Raspoznavanje uzoraka** (pravilo najbližeg susjeda, Bayesovo pravilo, algoritmi za grupiranje,…).
- **Izdvajanje (ekstrakcija) značajki slike** (prostorne osobine, osobine u domeni transformirane slike, detekcija rubova, detekcija granica).

- **Segmentacija slike** (amplitudna, obilježavanje komponenti, granice objekata, unutarnjost objekta, grupiranje, ekspertni sistemi, neuronske mreže,… ).
- **Matematička morfologija** (binarna i siva).
- **Analiza tekstura, oblika i pokreta.**
- **Registracija slike** (klasifikacija metoda za registraciju, pregled geometrijskih transformacija i algoritama).
- **Stereovizija** (kalibracija kamere, stereovizija, problem korespondentnih tačaka).

Većina vizualnih robotskih sistema koriste **kamere** za opažanje prostora. Opažanje područja je izuzetno važno u mobilnoj robotici za izbjegavanje prepreka pri obavljanju nekog zadatka. Međutim, kod vizualnog sistema (kamera) izražen je problem gubitka **informacija o dubini.** Ako se mogu predvidjeti informacije o dimenzijama objekta, ili pak o njegovoj boji i refleksiji, tada se mogu izvući informacije o dubini. Svojstva slike ne ovise samo o promjenama u sceni već i o parametrima kamere.

## CCD i CMOS kamere

*CCD (Charged Coupled Device) kamera*

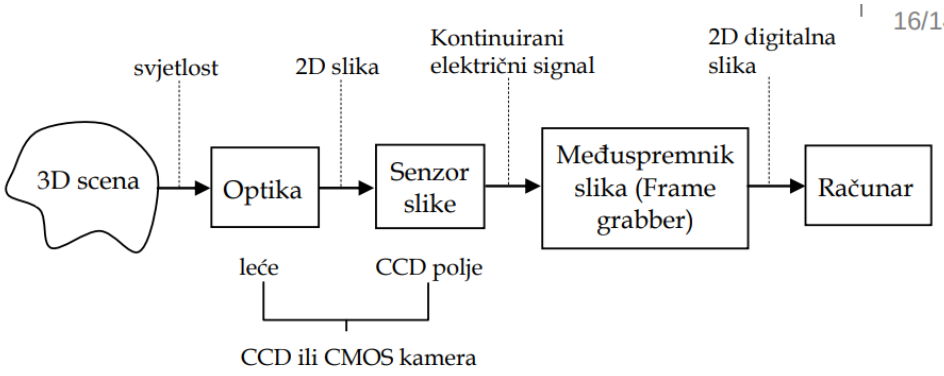
Fotoni svjetlosti udaraju u svaki piksel, oslobađaju elektrone koji su obuhvaćeni električkim poljem i zaustavljaju se na pikselu (fotoelektrički efekt). Tokom vremena svaki piksel će akumulirati različit iznos energije na temelju ukupnog broja fotona koji su "udarili" u njega. Nakon što se period integracije kompletira, relativan iznos energije piksela se pohranjuje i čita – proces kolekcije.

*CMOS kamera*

CMOS čip se značajno razlikuje od CCD čipa. On također ima polje piksela, ali su oni locirani paralelno i svaki piksel koristi nekoliko tranzistora za svoje lociranje. Proces integracije (akumuliranje energije u pikselima) je isti kao kod CCD čipa. U fazi kolekcije CMOS koristi drukčiji pristup, gdje se svaki piksel mjeri i pojačava se njegov signal, pri čemu se to odvija paralelno za svaki piksel u polju. Rezultantne vrijednosti piksela se prenose do njihovih destinacija.

Obje tehnologije sistema vizije, CCD i CMOS, generiraju digitalne signale koji se mogu direktno iskoristiti u robotici. Čip u kameri posjeduje paralelne digitalne ulazne i izlazne pinove preko kojih se prenose diskretne vrijednosti sadržaja piksela. Neki moduli vizualnog sistema koriste ove digitalne signale i obrađuju ih u stvarnom vremenu. Za ovu svrhu koristi se čip koji predstavlja međuspremnik slika (frame grabber) između digitalnih izlaza kamere i digitalnih ulaza u računar. Ovakvi čipovi "hvataju" kompletnu sliku i omogućuju pristup pikselima, obično u pojedinačnom poretku.

Sklopovlje sistema vizije prikazano je na slici 2.2.




Slika 2.2. Sklopovlje sistema vizije

Ovdje nećemo pričati o neki glavnim parametrima kamere, kako se kamera klibriše i kako se uzorkuje slike, naime, fukus je stavljen na samo obradu gotove slike.

### Značajke slike

Osnova mnogih vizualnih zadataka u robotskoj viziji je usporedba dvaju ili više pogleda istog objekta dobivenih sa dvije/više kamera ili dobivenih iz jedne kamere sa različitih pozicija, pri čemu je jedan od njih referentni pogled. Često je korisno izabrati takve apstrakcije slika koje pojednostavljuju usporedbu. Pojedinačni pikseli su samo indirektno povezani sa intenzitetom (boja) objekta u sceni. Iznosi piksela su rezultat složene interakcije između osvjetljenosti prostora, položaja kamere, prisustva drugih objekata u prostoru i refleksijskih svojstava objekata.

Vrijednosti pojedinačnih piksela ne koriste se direktno u većini zadataka robotske vizije. Zbog toga je potrebno načiniti njihovo preobradbu da se istaknu korisne strukture u slici. U nastavku se obrađuju sljedeće značajke slike:

- **boja i sjena,**
- **ograničenja osvijetljenosti slike (optički tok)**
- **korelacija**
- **izdavajanje značajki**
- **izdvajanje rubova**

Umjesto razmatranja cjelokupne slike neki pristupi koriste heuristike za identificiranje pojedinih tačaka na slici koje su važne za određeni zadatak i kasnije korištenje tih tačaka za prikazivanje slike. Korneri, naprimjer, često su veoma važni za vizualne zadatke i za njihovo identificiranje se koriste ranije izgrađeni detektori kornera. Opća klasa takvih operatora je poznata pod imenom detektori značajki ili operatori od interesa. U praksi ovi detektori trebaju identificirati lokacije koje su stabilne unutar malih promjena u slici i malih promjena u tački pogleda. Oni također sažimaju sliku u obliku brojnih podesivih značajki.

Postoje razni algoritmi za detekciju različitih značajki ali u ovom seminarskom radu bit će obrađeni Canny operator za detekciju ivica i Hough transformacija za detekciju linija i krugova na slici, pa će oni samo biti u nastavku detaljnije obrađeni.

### Canny operator

Nastao 1986 na MIT-u.

I dalje ima superiorne karakteristike u pogledu detekcije ivica.

Kani je formulisao tri uslova koja bi idealni detektor trebalo da ispuni:

- ✓ Mali procenat lažnih detekcija, visok procenat pravih detekcija
- ✓ Dobra lokalizacija
- ✓ Jedinični odziv na ivicu (sve ivice su širine 1 piksel)

Ova metoda koristi višerazinski algoritam za detekciju rubova u slici. Za razliku od ostalih metoda za detekciju i izdvajanje rubova, Cannyjeva metoda koristi dvije različite vrijednosti pragova za detektiranje jakih i slabih rubova, te uključuje slabe rubove samo u slučaju kada su povezani sa jakim rubovima. Na ovaj način omogućeno je kvalitetnije izdvajanje rubova iz slike u prisustvu šuma.

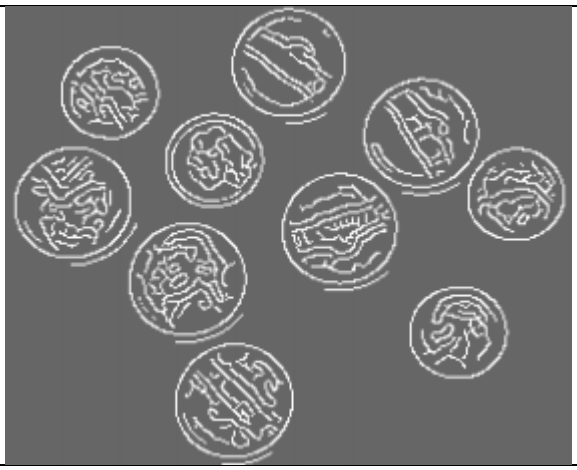
Cannyjeva metoda predstavlja "optimalnu" metodu za detekciju, odnosno izdvajanje rubova. Optimalna funkcija Cannyjevog detektora opisana je zbrojem eksponencijalnih članova.

Cannyjev algoritam može se razmatrati kao filter koji ulaznu crno-bijelu sliku obrađuje tako da na izlaznoj slici postoje samo rubovi ili ne-rubovi.

**Detekcija rubova odvija se u četiri faze: izglađivanje (usrednjavanje) slike, određivanje gradijenta intenziteta slike, stanjivanje rubova i uporedba s pragom.**

Primjer primjene Canny operatora na nekoj slici prikazan je na slici 2.3.

a) Orginalna slika



b) Canny

Slika 2.3. Primjena Canny operatora na slici

### Hough transformacija

Houghova transformacija je robusna tehnika procjenjivanja parametara temeljena na načelu glasanja. Patentirao ju je Paul Hough 1962. kao metodu određivanja orijentacije linija. U računalnom vidu, Houghovom transformacijom pronalaze se parametri nesavršenih primjeraka zadane klase oblika. Isprva je korištena za detekciju ravnih linija, a kasnije je proširena i na neke kompleksnije parametarske oblike poput kružnica i elipsa. Generalizirana Houghova transformacija, koju je 1979. opisao D. H. Ballard, generalizirana je za detekciju proizvoljnih parametarskih oblika.

Pravu liniju, odnosno pravac, potrebno je opisati u obliku minimalnog broja parametara.

Standardni zapis $v = au + b$ je problematičan u slučaju vertikalnog pravca kada je $a = \infty$. Umjesto toga može se koristiti $(\rho, \theta)$ parametrizacija:

$$u\sin\theta + v\cos\theta = \rho \tag{2.1.}$$

odnosno:

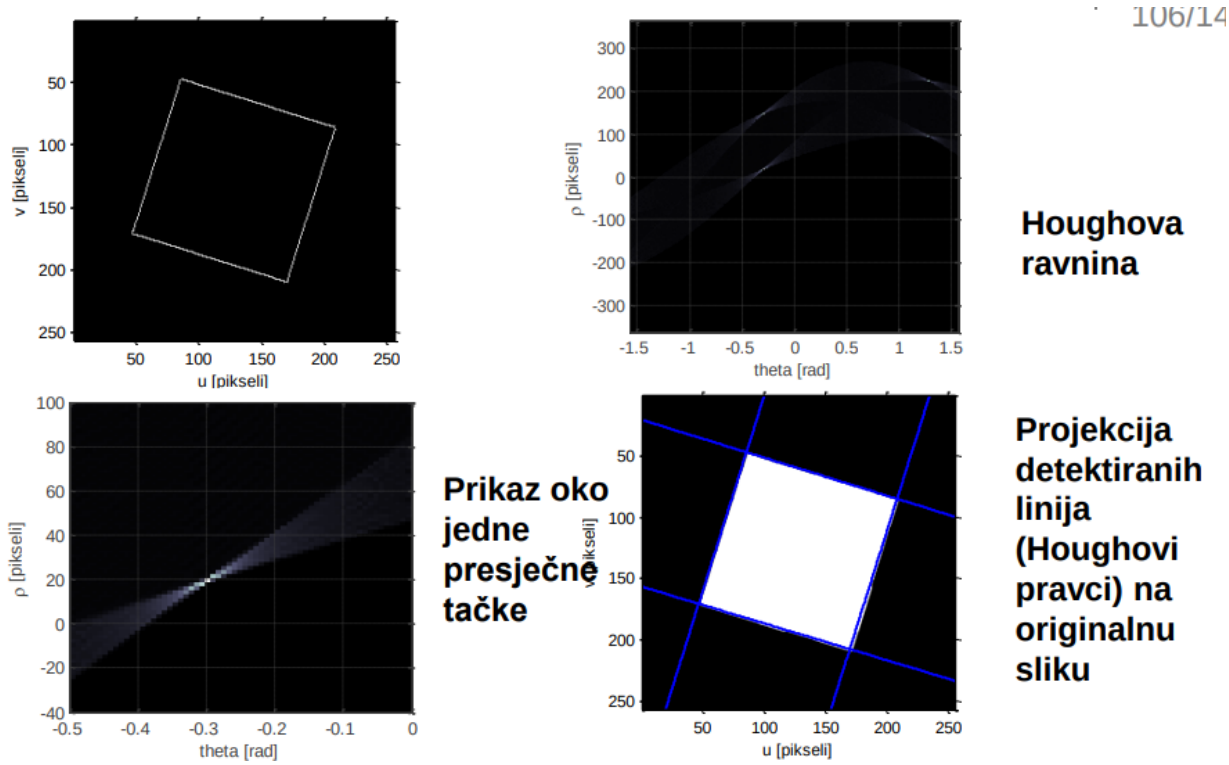
$$v = -u\tan\theta + \frac{\rho}{\cos\theta} \tag{2.2.}$$

$\Theta = [-\pi/2, \pi/2)$ je ugao nagiba i $\rho$ udaljenost pravca od ishodišta, za koji vrijedi $\rho = [\rho min, \rho max]$.

Horizontalni pravac ima $\theta = 0$ i vertikalni $\theta = -\pi/2$. Svaki pravac može se promatrati kao tačka $(\rho, \theta)$ u dvodimenzionalnom prostoru svih mogućih linija. Houghova transformacija preslikava

pravac u tačku u koordinatnom sistemu ρθ. Familija pravaca koja polazi kroz jednu tačku bi se preslikala u skup tačaka koje leže na sinusoidi. U praksi se razmatraju pravci iz konačnog skupa.

Primjer primjene Hough transformacije na sliku je dat a slici 2.4.




Slika 2.4. Primjer Hough transformacije na slici

U ovom poglavlju je prikazana osnovna teorijska podloga zadatka obrade slike i teorijska podloga algoritama koji će biti korišteni, dok u sljedećem poglavlju bit će spomenuto programsko okruženje koje će biti korišteno. [1]

# 3. ROS (eng. Robotic Operating System)

Danas se u nastavi i istraživanju koristi mnogo hardverskih i softverskih komponenti za izgradnju mobilnih robota. Većina tih komponenti je zamišljena i implementirana kao zatvoren sistem bez mogućnosti integracije u neke veće sisteme. Poseban problem nastaje kada se pokušavaju integrirati različite komponente koje obavljaju isti posao: CPU, kontroleri za motore, senzore itd. Robotic Operating System (u nastavku teksta ROS) pokušava da riješi taj problem. Zamisljen kao projekat otvorenog koda i specifikacija pod BSD licencom, privukao je ogromnu društvenu zajednicu u njegovo razvijanje i korištenje. Ovaj sistem nudi skup biblioteka i alata koji pomažu programerima da jednostavno

razvijaju softver za robotske sisteme. On pruža hardversku apstrakciju, drivere za uređaje, biblioteke za manipulaciju podacima, vizualizaciju, prosljeđivanje poruka, kao i jednostavno upravljanje paketima i modulima ROS-a.

ROS pokreće sve vrste robota: od malih mobilnih robota sa diferencijalnim pogonom do robotskih manipulatora ili automobila. Neki od najčešće korištenih robotskih platformi su:

- PIXHAWK, Skybotix CoaX Helicopter (helikopteri),
- Meka, Qbo, Mini-PR2, Lego NXT (setovi za ucenje),
- Shadow Dextrous Hand, Robotino, Penn Quadrotors, Washington University's B21r and Videre ERRATICs (mobilni roboti),
- Care-O-bot 3, BOSCH RTC, EL-E and Cody, Kawada HRP2-V (mobilni robotski manipulatori),
- STAIR 1, Aldebaran Nao, Junior (humanoidni roboti).

Obzirom na svoju heterogenost i funkcionalnosti koje podsjećaju na prave operativne sisteme ROS je našao primjene u raznim oblastima:

- Robotska vizija
- Navigacija
- Upravljanje
- Mobilna robotika
- Planiranje kretanja itd.

Kako je model razvoja ovog softvera otvorenog koda to je osiguran stalni razvoj u budućnosti i sve veća prihvaćenost kako u naučnim institucijama tako i u industriji. ROS nije operativni sistem! To znaci da ne dolazi kao distribucija nekog operativnog sistema, nego se instalira na već postojeći operativni sistem. Za sada je potpuno podržan Ubuntu Linux u svim verzijama, dok su dostupne eksperimentalne varijante za neke operativne sisteme koji neće biti nabrajani ovdje. [2]

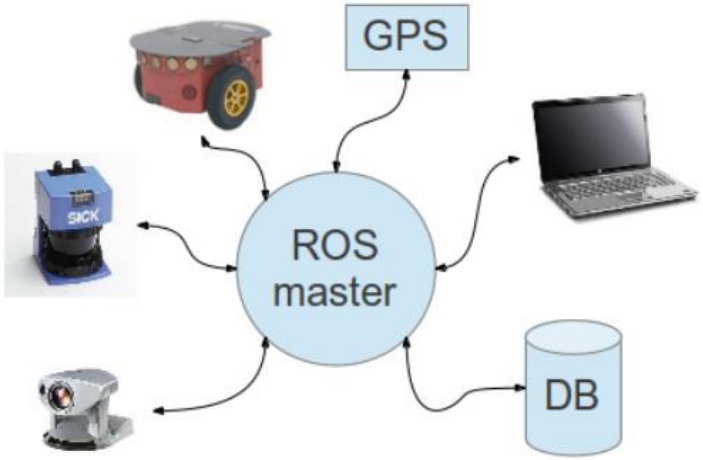
## 3.1. Arhitektura ROS-a

Ciljevi koji su definirani prilikom dizajna ROS-a su da on bude prije svega skalabilan. To znači da se bez puno posla jedan kod može primijeniti na što više istih ili različitih robotskih sistema. Ukratko, ti ciljevi su:

- Peer-to-peer komunikacija
- Baziran na alatima
- Višejezičan (u smislu korištenih programskih jezika)
- Besplatan i otvorenog koda.

**Peer-to-peer** Sistem razvijen korištenjem ROS-a se sastoji iz više procesa, pa čak i različitih računara na mreži u peer-to-peer topologiji. Ovaj pristup je drugačiji od centraliziranog pristupa

u tome sto master ima funkciju isključivo da usmjerava poruke na prave klijente. U slučaju robotskog sistema ti klijenti su aktuatori, senzori i kontroleri.




Slika 3.1. Primjer arhitekture robotskog sistema sa ROS-om

**Višejezičnost** Mnogi programeri imaju svoje ”najdraže” programske jezike. Zbog toga je ROS dizajniran da bude neutralan po pitanju korištenog programskog jezika. Do sada je potpuno implementirana podrška za jezike C++, Python, Octave i LISP, dok su za mnoge druge jezike razvijanja u toku. Sam ROS je implementiran u C programskom jeziku sto osigurava brzinu i veoma dobro upravljanje hardverskim resursima. Prosljeđivanje poruka je definirano XML-RPC protokolom za koji postoji implementacija u skoro svim poznatim programskim jezicima. Zbog toga, ali i zbog zamisljene distribuirane filozofije sistem moguće je da klijenti izvršavaju kod napisan u potpuno različitim programskim jezicima.

**Baziran na alatima.** Zbog jednostavnijeg upravljanja nadogradnje kompleksnog sistema kao sto je ROS, dizajn podsjeća na mikrokernelsku filozofiju koja se susreće kod operativnih sistema. To znači da se veliki broj malih alata koristi za izgrađivanje komponenti sistema: od postavljanja parametara do vizualizacije podataka.

**Besplatan i otvorenog koda** ROS je moguće besplatno preuzeti sa interneta. Sav kod je javno dostupan i moguće ga je preuzeti i nastaviti razvijati.

U ovom poglavlju opisana je osnovna struktura ROS-a i osnovne informacije o ROS-u. Detaljnije neće biti obrađen ROS iako je da obradu ovakvog zadatka ui njemu potrebno mnogo veće znanje od ovog prethodnom, ali kako je ROS open source sve značajno možete pronaći na oficijanoj stranici ROS-a ROS.org i pomoć korisnicima z ROS ROS wiki. [2]

## 3.2. Korišteni ROS paketi i biblioteke

Da bi se ovaj problem obradio potrebno je uključiti odgovarajuće biblioteke ROS-a. Glavna biblioteke koja će biti korištena je **OpenCV** biblioteka. Ova biblioteka je upravo namijenjena za obradu slike, obradu videa i slično.

### OpenCV

OpenCV je computer vision biblioteka otvorenog koda. Njen fokus su aplikacije koje rade s podacima u stvarnom vremenu (real-time aplikacije).
Jedan od glavnih ciljeva OpenCV-a je pružanje jednostavne infrastrukture za rad sa computer vision-om i omogućuje jednostavan i brz razvoj jednostavnih ali i složenijih aplikacija. Biblioteka sadrži više od 500 funkcija koje se protežu kroz gotovo sva područja računalne vizije i strojnog učenja (machine learning).
Naravno, obavljanje velikog broja operacija i kompliciranih algoritama nad podacima u stvarnom vremenu zahtjeva optimiziran kod pisan na nižoj razini. Upravo zato je OpenCV originalno pisan u programskom jeziku C. Službeni repozitorij se može pronaći na github stranici projekta. Osim korištenja C jezika za razvoj openCV aplikacija, razvijena su sučelja prema mnogo drugih jezika više razine kao npr. Java, Python, Ruby itd.

OpenCV ima podršku i za algoritme koje je potrebno koristiti u seminarskom radu, Cannny and Hough Transform. Kako izgleda njihova primjena na nekoj slici može se vidjeti u oficijelnoj dokumentaciji OpenCV-a.

U ovom seminarskom radu koristit će se OpenCV C++, tj. cijeli kod će biti C++ kod. Logo OpenCV-a je prikazan na slicic 3.2. [3]


Slika 3.2. Logo OpenCV-a

Package koji su potrebni da bi se realizovao dati zadatak su:

**USB_cam package**

Predstavlja ROS drajver za V4L USB kamere.

Čvorovi packaga su:

**usb_cam_node** koji ostvaruje konekciju sa USB kamerom u ovom slučaju sa web kamerom laptopa.

Objavljene teme su:

**<camera_name>/image**

Parametri su:

~video_device (string, default: "/dev/video0")

~image_width (integer, default: 640)

~image_height (integer, default: 480)

~pixel_format (string, default: "mjpeg")

...jpeg, yuyv, uyvy

~io_method (string, default: "mmap")

... mmap, read, userptr

~camera_frame_id (string, default: "head_camera")

~framerate (integer, default: 30)

~contrast (integer, default: 32)

(0-255)

~brightness (integer, default: 32)

(0-255)

~saturation (integer, default: 32)

(0-255)

~sharpness (integer, default: 22)

(0-255)

~autofocus (boolean, default: false)

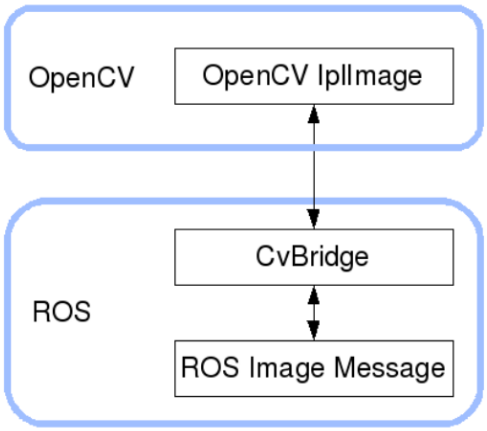
~focus (integer, default: 51)

~camera_info_url (string, default: )

~camera_name (string, default: head_camera) [4]

**Cv_bridge package**

Paket koji vrši konverziju između ROS Image poruka i OpenCV Image poruka. Na način koji je prikazan na slici 3.3.




Slika 3.3. Konverzija između OpenCV i ROS Image poruka

Na ROS wiki moguće je naći i odgovarajuće tutorijale kako napisati kod za čvor koji uprvo radi konverziju između OpenCV i ROS-a. Cilj je spomenuti samo osnovne detalje o package-u a detaljno upoznavanje s package-om je ostavljeno čitaocu ovog seminarskog rada. [4]

**Image_transport package**

Ovaj paket se koristi kada je god upitanju objavljivanje slike na neku temu ili preplaćivanje na temu. Omogućava podršku za transport slika u kompresiranom formatu.

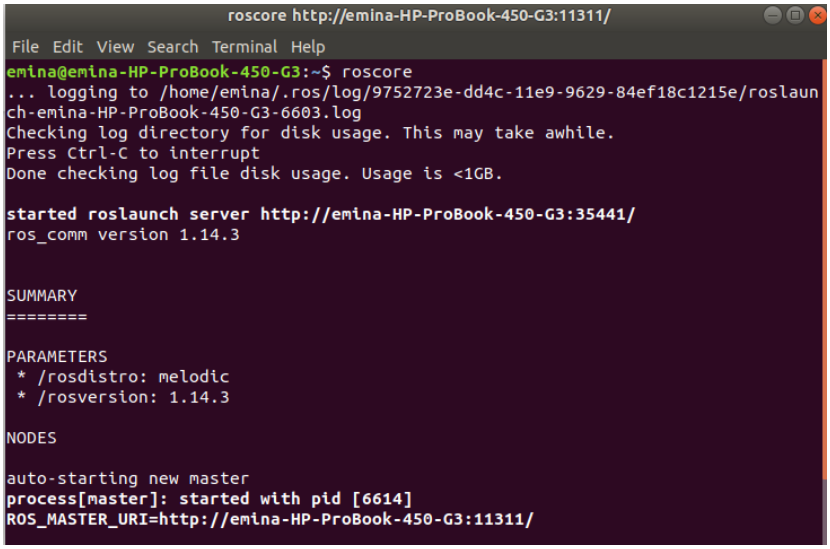
Kako napisati Publisher i Subscriber slika koristeći ovaj paket može se pronaći na ROS wiki.

Sada kada smo obradili osnovne pakete koji su nam potrebni da realiziramo dati problem u ROS-u pristupano pisanju koda za čvor koji će osluškivati temu <camera_name>/image na koju objavljuje slike usb_cam_node paketa usb_cam i vršit će obradu slika te obrađene slike objaviti na novu temu. Implementacija će biti data u narednom poglavlju. [4]

# 4. Implementacija u ROS-u

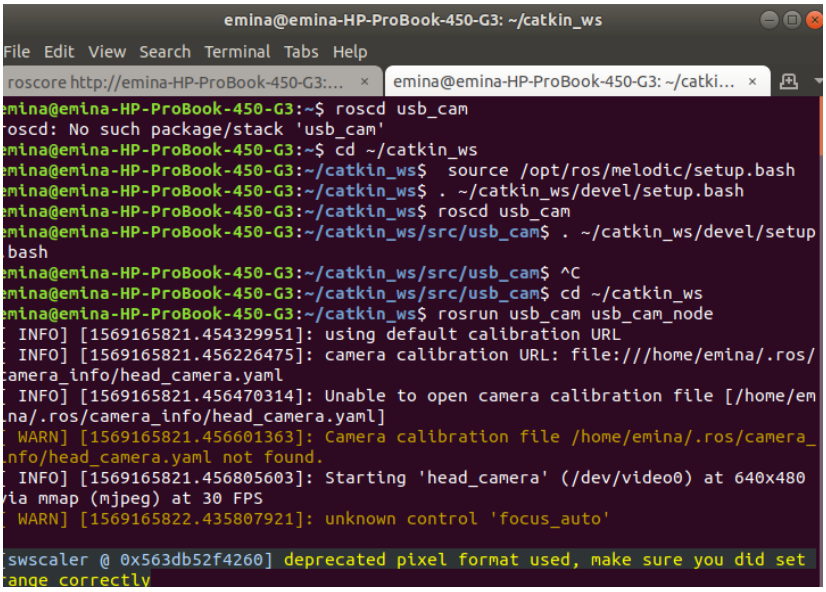
Usb_cam package-a koji je naveden i objašnjen u prethodnom poglavlju predstavlja drajver za web kameru i čiji čvor objavljuje slike na odgovarajuću temu. Pokretanje ovog već implementiranog čvora će biti prikazano u nastavku i kako on objavljuje slike.

Prvo je pokrenut ROS master – slika 4.1.:



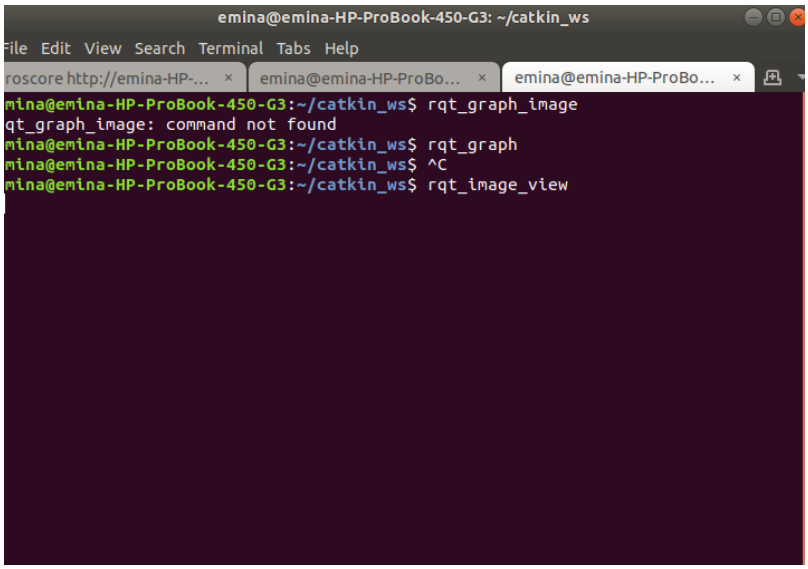

Slika 4.1. Pokretanje ROS mastera

Pokretanje usb_cam_node-a, slika 4.2.:



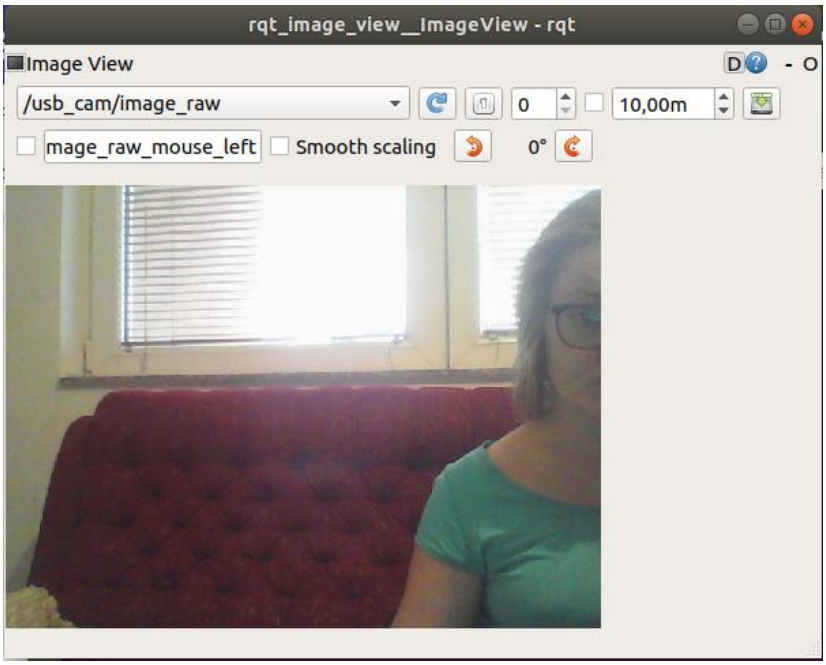

Slika 4.2. Pokretanje usb_cam_node-a

Pokretanje rqt_image_view-a kako bi prikazali trenutni snimak web kamere – slika 4.3.:



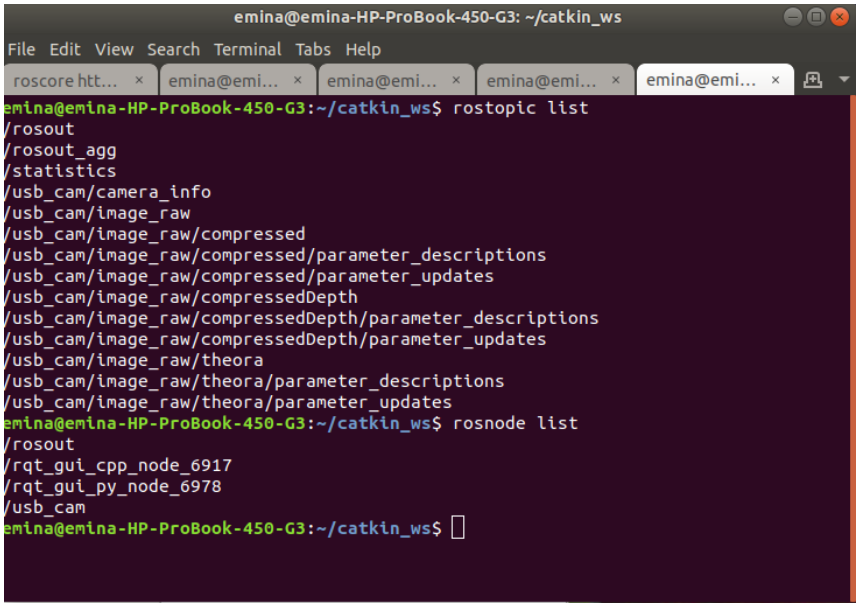

Slika 4.3. Pokretanje rqt_image_view-a

Kada samo pokrenuli rqt_image_view, na njemu vidimo sljedeće, web kamera je upaljena i prikazuje trenutno stanje ispred sebe- slika 4.4.




Slika 4.4. Live prikaz sa web kamere

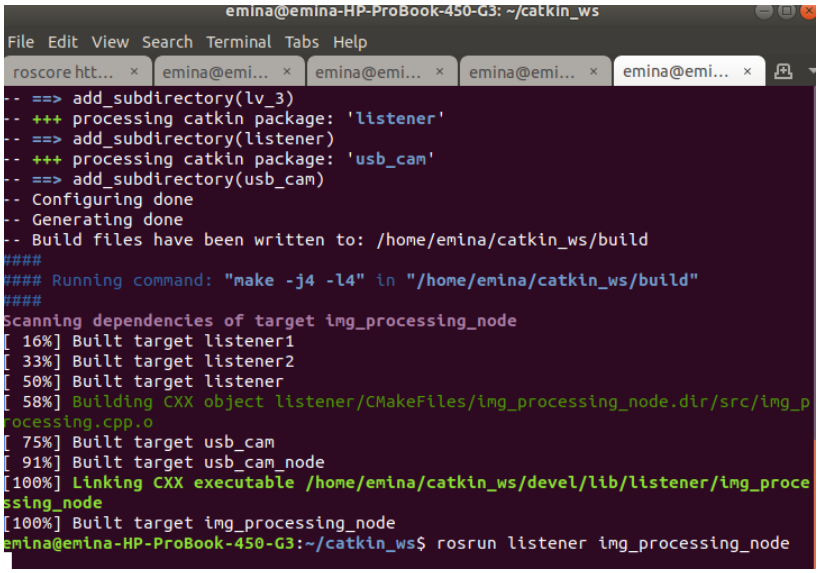
Provjerit ćemo i rostopic list i rostopic node, da vidimo koje su to teme trenutno aktivne i čvorovi, naravno očekujemo da su prisutne teme koje objavljuje usb_cam_node – slika 4.5.




Slika 4.5. Trenutno objavljene teme i prisutni čvorovi

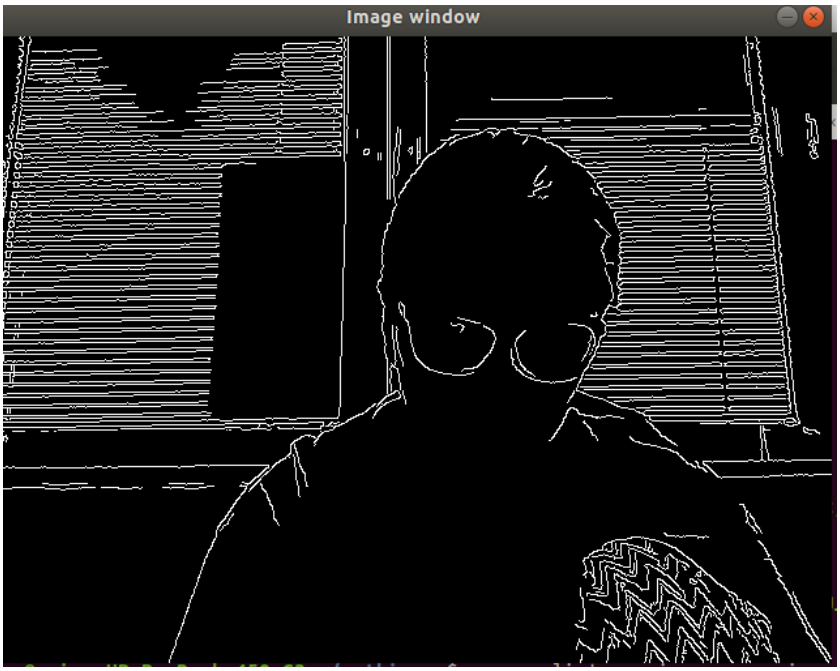
Sada je potrebno pokrenuti čvor koji osluškuje ovu temu, kod je napisan u C++ programskom jeziku i korišteno je dosadašnje znanje iz ovog programskog jezika i packag-i i biblioteke koje su pomenute u prethodnom poglavlju.

Čvor koji je nazvan img_processing_node se nalazi u ručno kreiranom package-u listener. Pokretanje čvora je prikazano na slici 4.6.



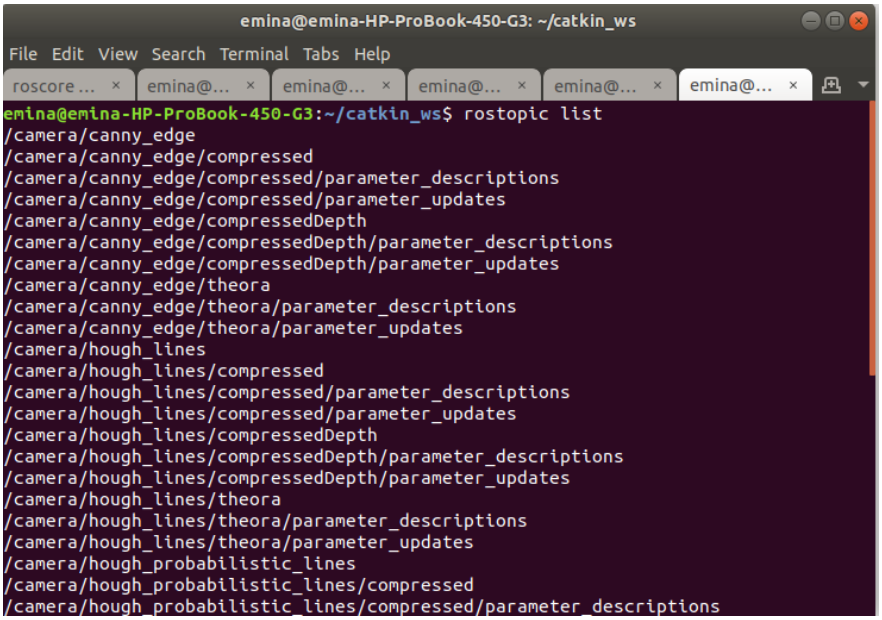

Slika 4.6. Pokretanje čvora img_processing_node

Nakon ovog pokazuje nam se prijenos s kamere ali ovaj put sa detektovanim ivica-ma ispred sebe jer smo prvo implementirali algoritam Canny za detekciju ivica – slika 4.7.

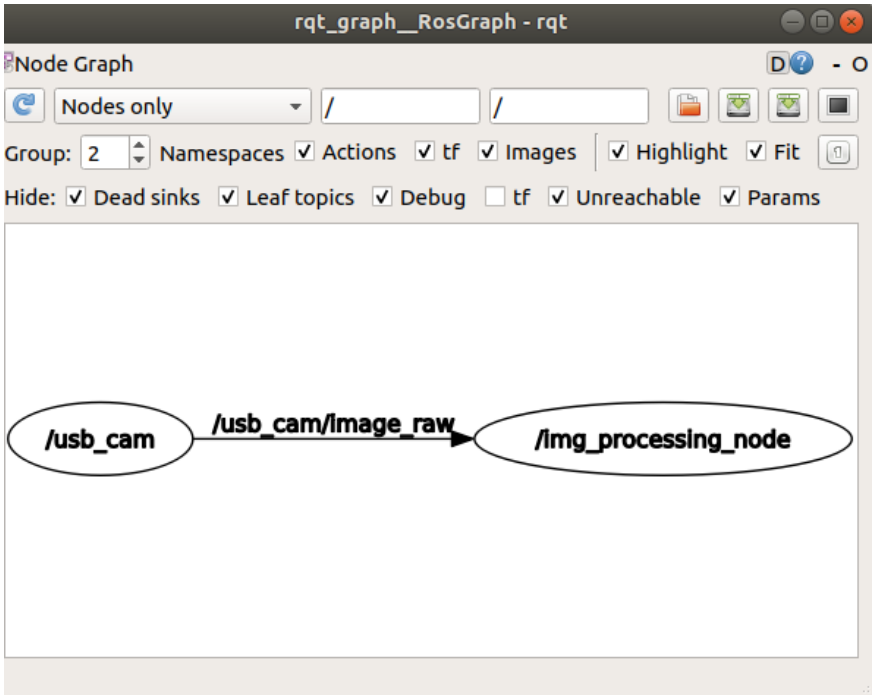



Slika 4.7. Live prenos sa web kamere sa detektovanim ivicama

Sada možemo provjeriti i teme koje su prisutne. Vidimo na slici 4.8. da sada čvor img_processing_node objavljuje nove teme i vidimo da on ostvaruje komunikaciju sa usb_cam_node što možemo vidjeti na rqt graphu – slika 4.9.

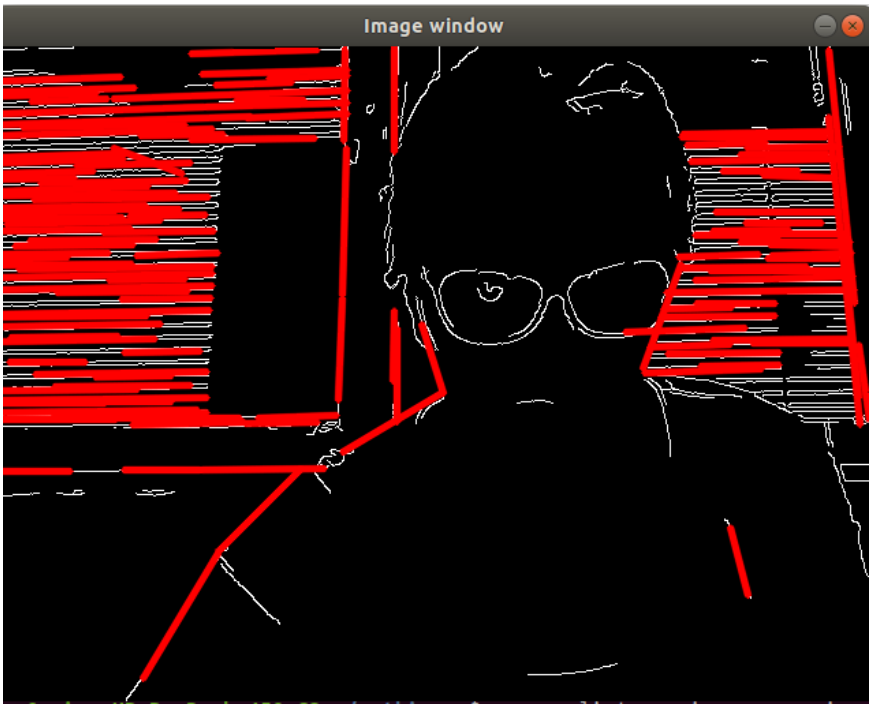



Slika 4.8. Prisutne teme nakon pokretanja i čvora img_processing_node

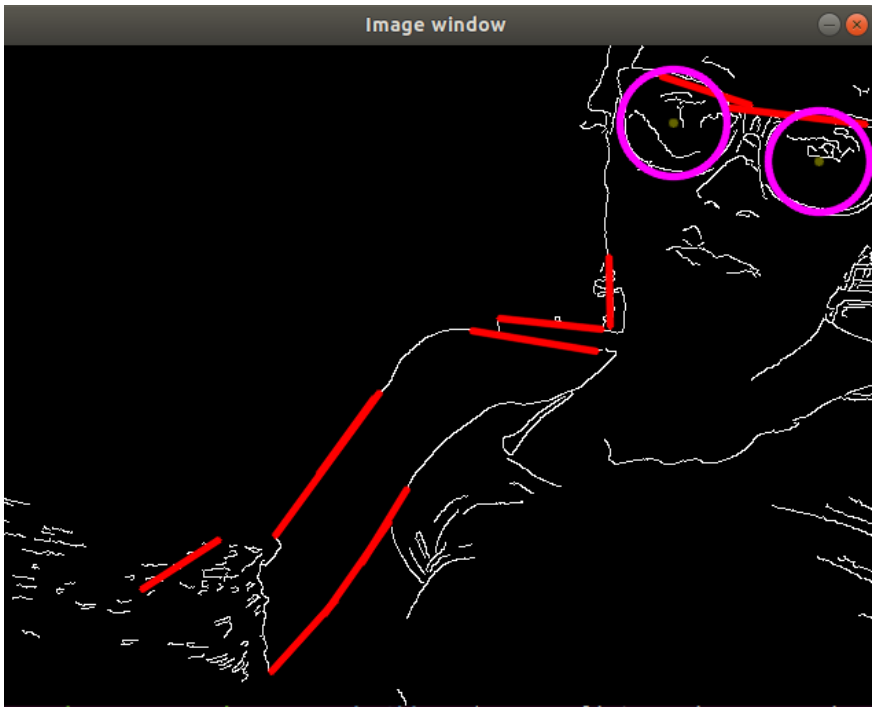



Slika 4.9. Rqt_graph

U nastavku će biti prikazana u Hough transformacija slike za detekciju ivica i krugova i kako izgleda live prijenos sa web kamere (slika 4.10 i 4.11). Koristit će se isti čvor samo će kod malo biti prepravljen, a ostalo je sve isto kao i u prethodnom koraku, što se tiče stanja rostopic liste, rqt grapha i pokretanja čvora. Bitno je napomenuti da čvor stalno objavljuje nove teme koje sadrže slike sa canny edge detekcijom (tema: /camera/canny edge), hough linije (tema: /camera/hough_lines), hough probablističke linije (tema: /camera/hough_probabilistic_lines), hough krugovi (tema: /camera/hough_circles).




Slika. 4.10. Detekcija samo linija




Slika 4.11. Detekcija krugova i linija

Sa prethodnih slika možemo uočiti da ovi algoritmi jako dobro detektuju i ivice i linija i krugove, iako se radi o live prenosu sa web kamere i nizu frame-ova ova detekcija je jako dobra.

Korišteni kod je dat kao prilog ovom izvještaju, tj. cijeli package listener sa popratnim package.xml i CMakeLIsts.txt fajlovima.

# Zaključak

Algoritmi koji su korišteni u ovom seminarskom radu su jako korisni u mobilnoj robotici. Svaki mobilni robot koji se kreće može imati senzor kameru na osnovu koje se može implementirati vizualna odometrija ili vizualni SLAM (eng. Simultaneous localization and mapping), a ovi algoritmi igraju ključnu ulogu pri implementaciji ovih zadataka.

# Reference

1. Jasmin Velagić Mobilna robotika, 2019.

2. Robotic Operating System priručnik za korištenje na predmetu Mobilna robotika

3. Oficijelna OPENCV stranica https://opencv.org/

4. Oficijelna ROS stranica: https://www.ros.org/ i http://wiki.ros.org/