Aplikacija proširene stvarnosti za pretvorbu cijena u hrvatsku kunu na uređajima s operacijskim sustavom Android

Kuzle, Karlo

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1627

APLIKACIJA PROŠIRENE STVARNOSTI ZA PRETVORBU CIJENA U HRVATSKU KUNU NA UREĐAJIMA S OPERACIJSKIM SUSTAVOM ANDROID

Karlo Kuzle

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1627

APLIKACIJA PROŠIRENE STVARNOSTI ZA PRETVORBU CIJENA U HRVATSKU KUNU NA UREĐAJIMA S OPERACIJSKIM SUSTAVOM ANDROID

Karlo Kuzle

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1627

Pristupnik:	**Karlo Kuzle (0036541081)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Krešimir Pripužić

Zadatak: **Aplikacija proširene stvarnosti za pretvorbu cijena u hrvatsku kunu na uređajima s operacijskim sustavom Android**

Opis zadatka:

U ovom radu potrebno je osmisliti, izvesti u programskom jeziku Kotlin te testirati aplikaciju proširene stvarnosti za konverziju cijena iz valute euro u hrvatsku kunu na uređajima s operacijskim sustavom Android. Ova aplikacija treba koristiti kameru uređaja na način da u stvarnom vremenu prepoznaje cijene prikazane u valuti euro, odmah ih konvertira u iznos u hrvatskoj kuni te ih primjenom metoda proširene stvarnosti prikaže na zaslonu uređaja. U teoretskom dijelu rada je potrebno predstaviti koncept aplikacija temeljenih na proširenoj stvarnosti te opisati softverske knjižnice koje se koriste u praktičnom dijelu rada.

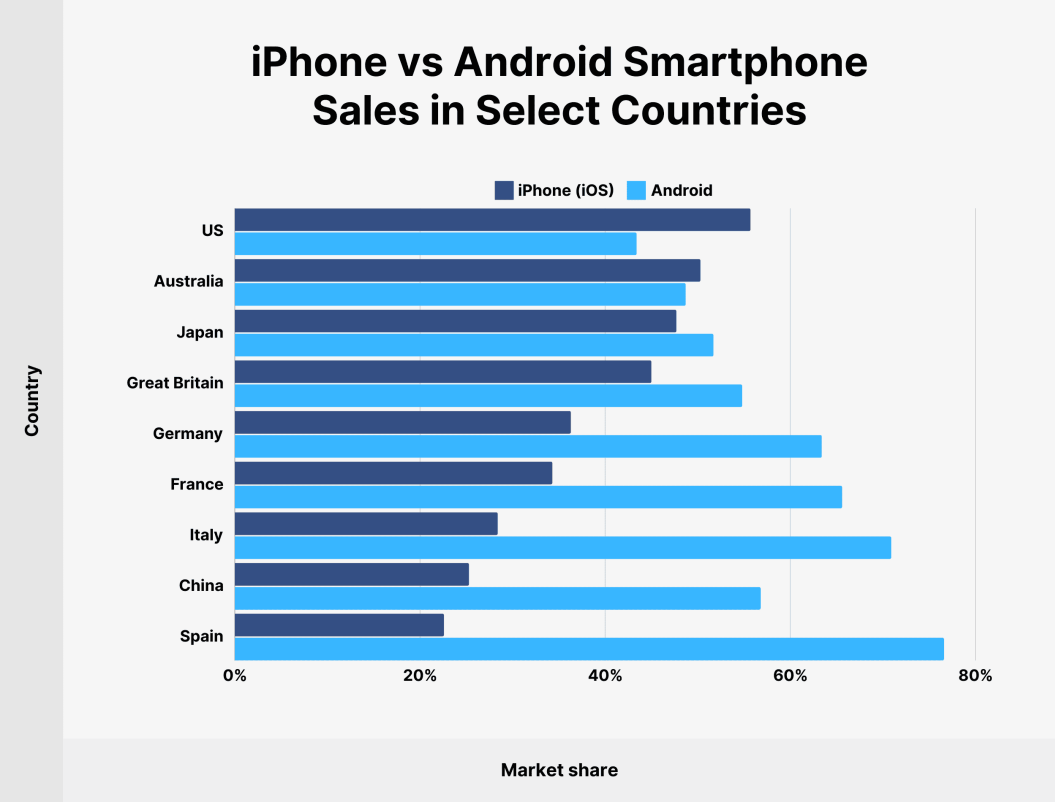
Rok za predaju rada: 14. lipnja 2024.

Sadržaj

1. Uvod

1.1. Android

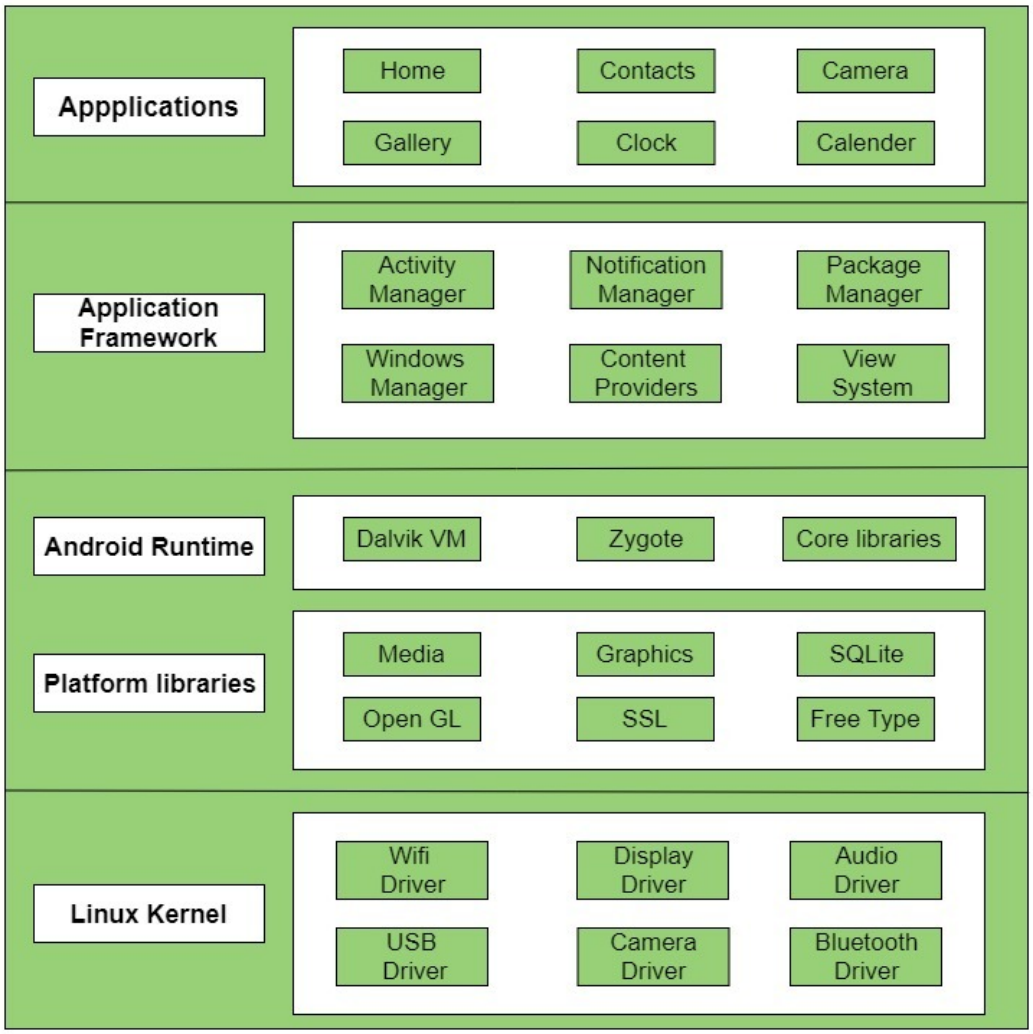
U svijetu telefona i drugih prijenosnih uređaja poput tableta dominiraju dva mobilna operacijska sustava, Android i iOS. Android zauzima oko 70% globalnog tržišta, dok je iOS dominantan u Sjedinjenim Američkim Državama (SAD) i time zastupa oko 28% globalnog tržišta.



Slika 1.1. Odnos prodaje iPhone i Android mobilnih uređaja u vodećim državama

Operacijski sustav Android razvila je tvrtka Google koja ga trenutno i održava. Arhitekturu operacijskog sustava Android dijelimo na pet slojeva. Najniži sloj, Linuxova ljuska upravlja sigurnošću, memorijom, procesima i mrežom. Sloj knjižnica u program-

skom jeziku C/C++ pruža različite funkcionalnosti poput grafike (OpenGL), baze podataka (SQLite) ili pregledavanja interneta (WebKit). Sloj Android Runtime sadrži osnovne knjižnice za pokretanje Androidovih aplikacija. Radni okviri za aplikacije pružaju visok nivo sučelja za programiranje aplikacija (engl. application programming interface, API). Na najvišem sloju aplikacija nalaze se sve instalirane aplikacije na uređaju.

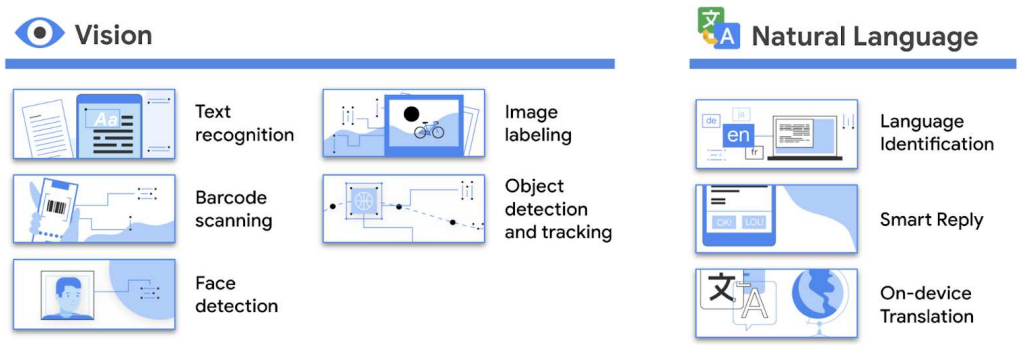


Slika 1.2. Arhitektura operacijskog sustava Android

Andoridove aplikacije sastoje se od aktivnosti. Svaka aktivnost ima svoj životni ciklus. Logika Androidove aplikacije kodirana je u programskim jezicima Java ili Kotlin, dok je korisničko sučelje definirano u tekstualnom formatu XML (engl. Extensible Markup Language). Sve sposobnosti pametnog uređaja dostupne su Androidovoj aplikaciji uz dotične dozvole i komponente. Koristeći kameru, mikrofon ili bilo koji drugi senzor pametnog uređaja, aplikacija korisniku omogućava iskustvo proširene stvarnosti.

1.2. Proširena stvarnost

Za razliku od virtualne stvarnosti (engl. virtual reality, VR) u kojoj su događaji potpuno nestvarni i nevezani uz korisnikovu okolinu, proširena stvarnost (engl. augmented reality, AR) je integracija digitalnih informacija s korisnikovim okruženjem u stvarnom vremenu [3]. Proširena stvarnost na pametnim telefonima najčešće je ostvarena korištenjem kamere ili obradom prirodnog jezika. Korisnici su kroz pogled kamere u interakciji s okolinom. Proširena stvarnost najčešće koristi alate temeljene na umjetnoj inteligenciji i strojnom učenju. Neki od najpoznatijih alata su skeniranje barkoda, prepoznavanje lica, detekcija mreže lica, označavanje slika i objekata na slikama, prepoznavanje teksta, detekcija poza, skeniranje dokumenata i prijevod i prepoznavanje jezika. Svi ovi alati ostvareni su kao API u Googleovom programskom paketu MLKit namijenjenom za mobilne aplikacije.

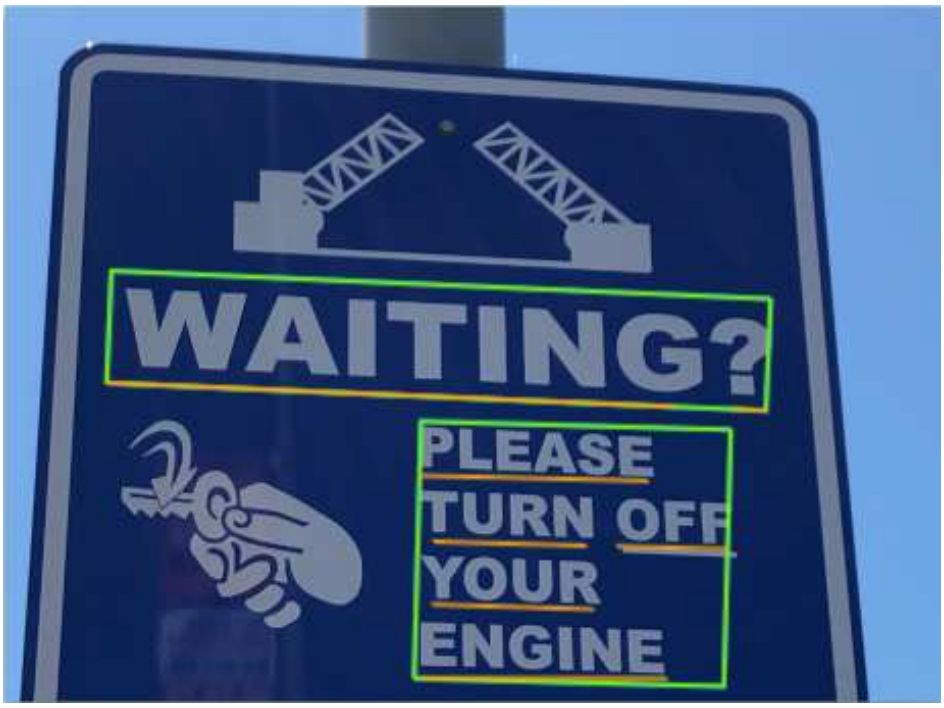


Slika 1.3. Google ML Kit

Alati raspoznavanja teksta koriste OCR (engl. optical character recognition) tehnologiju.

1.3. OCR

Dok je za ljude prepoznavanje teksta i simbola trivijalno, pokazuje se da je isti zadatak vrlo složen ako ga obavlja računalo. OCR tehnologije osnivaju se na prepoznavanju obrazaca i algoritmima umjetne inteligencije. OCR dijelimo na dva glavna procesa, detekciju teksta i raspoznavanje teksta. Detekcija teksta također je složen proces jer ovisi o boji, veličini, obliku, smjeru, duljini i korištenom pismu. Dodatni problemi na koje OCR nailazi su loša rezolucija slike, sjene, preklapanje i neujednačenost teksta. Svi ovi čimbenici otežavaju detekciju teksta.



Slika 1.4. Detekcija teksta

Detekciju teksta dijelimo na metode zasnovane na regresiji i na metode zasnovane na segmentaciji. Metode zasnovane na regresiji dijele elemente slike u dvije klase: tekst i pozadinu. Pri detekciji nepravilnog teksta korištene su metode zasnovane na segmentaciji slike. Za svaki piksel na slici određeno je pripada li dijelu teksta. Nakon klasifikacije svih piksela, stvorena je obuhvatna krivulja oko teksta na slici. Nakon detekcije teksta, zadaća raspoznavanja teksta je pretvoriti informacije iz slike u informacije u tekstu.

Raspoznavanje teksta je podijeljeno u tri koraka: predobradu slike, segmentaciju znakova i prepoznavanje znakova. Prepoznavanje znakova ostvareno je metodom usporedbe matrica ili ekstrakcijom značajki i klasifikacijom. Metoda usporedbe matrica uspoređuje segmentirani znak s preddefiniranim uzorcima, dok metoda ekstrakcije značajki i klasifikacije koristi neuronske mreže i algoritam K-najbližih susjeda. Proces završava ispravljanjem grešaka i rekonstrukcijom teksta.

Android aplikacije proširene stvarnosti koristi alate iz Googleovog programskog paketa ML Kit kako bi u stvarnom vremenu pretvarala cijene iz eura u hrvatsku kunu.

Slika 1.5. Raspoznavanje teksta

2. Programska podrška

2.1. Odabir softverskih knjižnica

Prvi korak u izradi aplikacije proširene stvarnosti na uređajima s operacijskim sustavom Android je odabir valjanih i podržanih knjižnica. Kako bi aplikacija ostvarila korištenje kamere potrebno je implementirati neku od knjižnica iz Androidovog radnog okviru Camera API.

Dok je knjižnica Camera zastarila i više nije podržana, preostale podržane knjižnice su Camera2 i CameraX. Knjižnica Camera2 je API niže razine i nije kompatibilna na uređajima s Android 5 verzijom i nižim verzijama, dok je CameraX jednostavnija za korištenje i bolje podržana. CameraX pruža samo predefinirane obrasce korištenja dok Camera2 pruža izravan pristup sposobnostima hardvera. Za potrebe izrade aplikacije proširene stvarnosti koristit ćemo knjižnicu CameraX.

Najpoznatije knjižnice za potrebe OCR tehnologije u razvoju aplikacija s operacijskim sustavom Android su Tesseract i Google ML Kit. Google ML Kit je vrlo dobro integriran s knjižnicom CameraX, stoga će nam služiti za potrebe prepoznavanja teksta.

2.2. CameraX[1]

Obrasci uporabe podržani u knjižnici CameraX su: pregled kamerom, analiza slike, snimanje slike i snimanje videozapisa. CameraX dopušta istovremeno korištenje više obrazaca uporabe. Za korištenje jednostavnih sposobnosti knjižnice CameraX namijenjen je skup razreda `CameraController`, dok skup razreda `CameraProvider` ostvaruje složenije sposobnosti.

Korištenjem metoda `set` konfiguriramo obrasce uporabe, dok naredbom `build` pot-

vrđujemo željene konfiguracije. Glavno svojstvo sučelja `CameraProvider` je dodjeljivanje životnog ciklusa kameri metodom `bindToLifecycle`. Element koji u Android radnom okviru ima dodijeljen životni ciklus prolazi kroz niz stanja od stvaranja (metoda `onCreate`) do uništenja (metoda `onDestroy`) elementa ili aktivnosti. Pojedine obrasce uporabe vežemo uz životni ciklus aktivnosti. Ukoliko simultano koristimo više obrazaca uporabe, prije vezanja uz životni ciklus, potrebno je sve nevažne obrasce upotrebe odvezati metodom `unbindAll`.

2.2.1. Pregled kamere

CameraX za prikaz pregleda kamere na ekranu uređaja koristi razred `PreviewView`, a za pokretanje obrasca uporabe pregled kamere koristimo razred `Preview.Builder`.

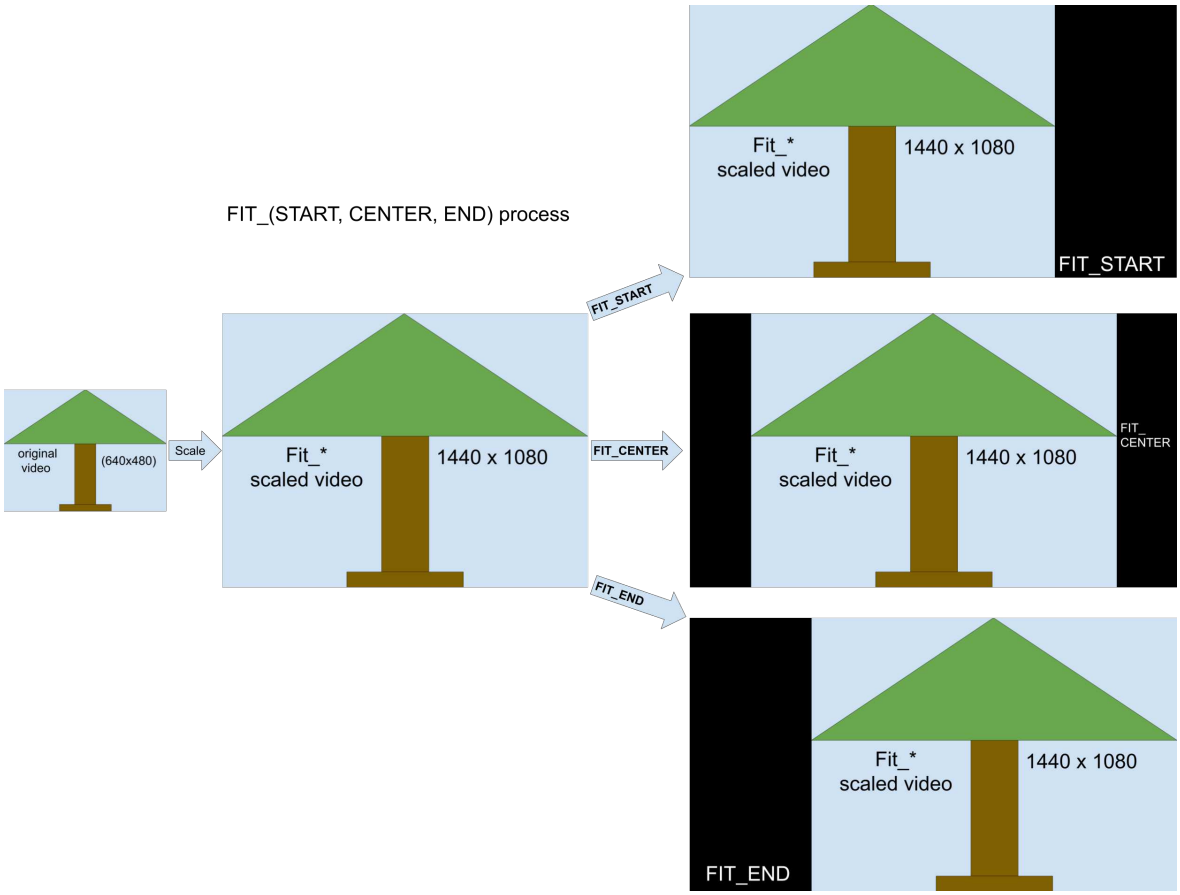
Razred `PreviewView` ima dva načina rada: Performance i Compatible. Dok je Performance osnovni način rada, Compatible način rada je prikladniji za skaliranje i rotaciju slike u pregledu. Razred `PreviewView` pruža mogućnost skaliranja videa za prikaz kako bi bio prikladan formatu prikaza (engl. *aspect ratio*) na uređaju. Podržane su opcije FIT_CENTER, FIT_START i FIT_END (slika 2.1) i FILL_CENTER, FIL_START i FILL_END (slika 2.2) opcije skaliranja slike.

2.2.2. Analiza slike

Analiza slike u klasi `CameraX` ostvarena je implementacijom sučelja pod nazivom `ImageAnalysis.Analyzer` i metode *analyze* koja se pokreće na svakoj slici. Objekt `ImageAnalysis` gradi se pomoću klase `ImageAnalysis.Builder` nad kojim je moguće namjestiti parametre izlazne slike. Neki od tih parametara su format izlazne slike, format prikaza i rotacija.

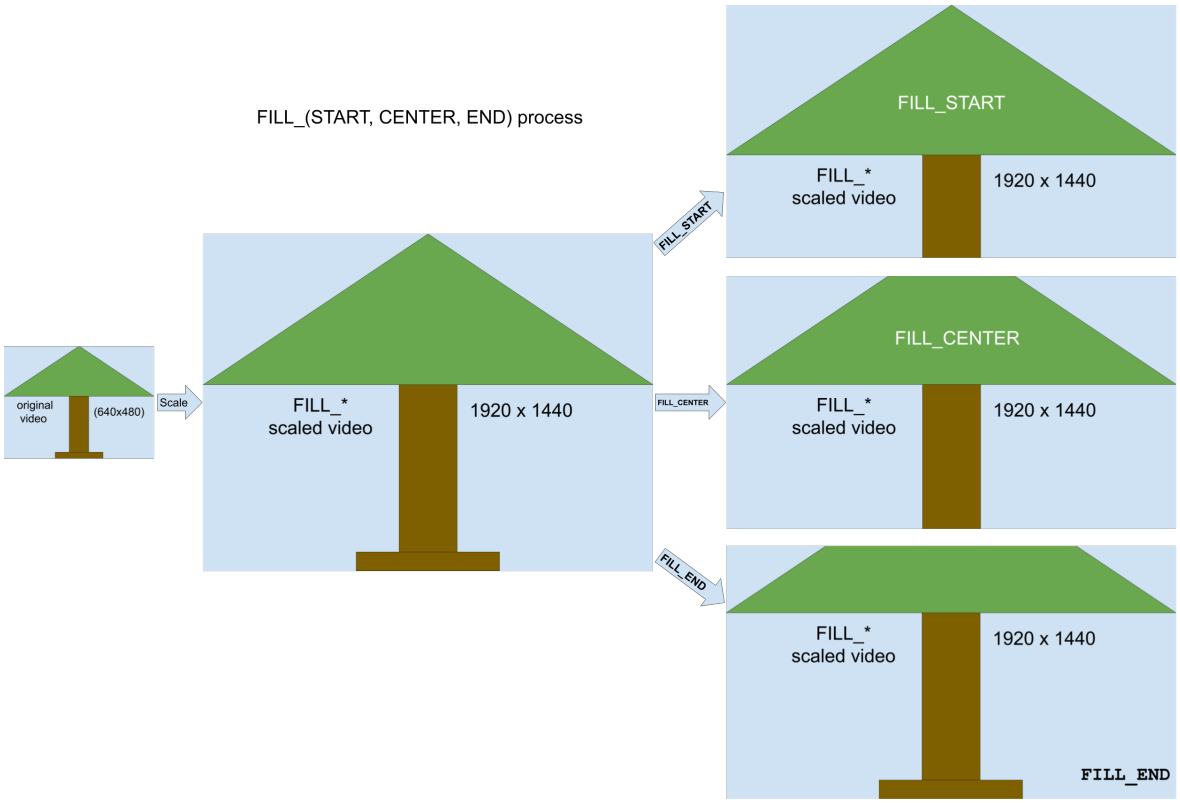
Podržani formati izlazne slike u knjižnici CameraX su YUV_420_888 i RGBA_8888. YUV_420_888 koristi tri kanala kako bi prikazao boju (Y je luminacija, dok U i V daju informacije o nijansi i zasićenosti boje). RGBA_8888 je format slike koji sprema 8 bitova po kanalu. R predstavlja crvenu, G zelenu, B plavu boju, dok je A prozirnost. Na slici 2.3 vidimo razliku između YUV i RGB formata.

Osim pregleda kamere i analize slike, CameraX kao obrasce uporabe omogućava sni-

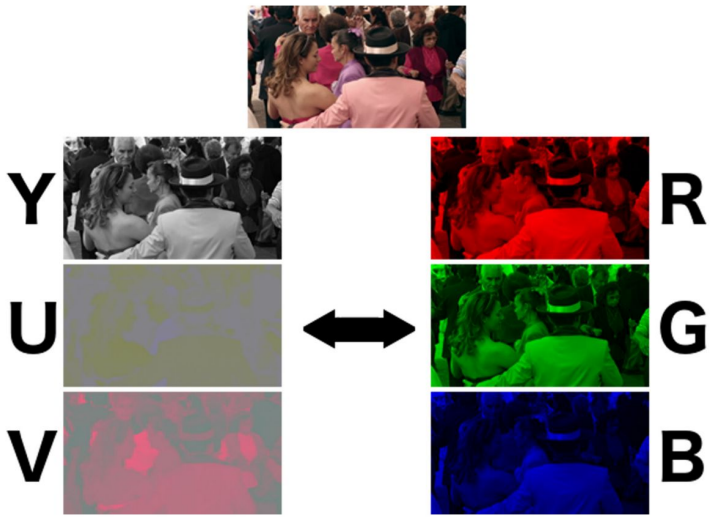


Slika 2.1. Fit opcije

manje slike te izradu videozapisa.



Slika 2.2. Opcije Fill



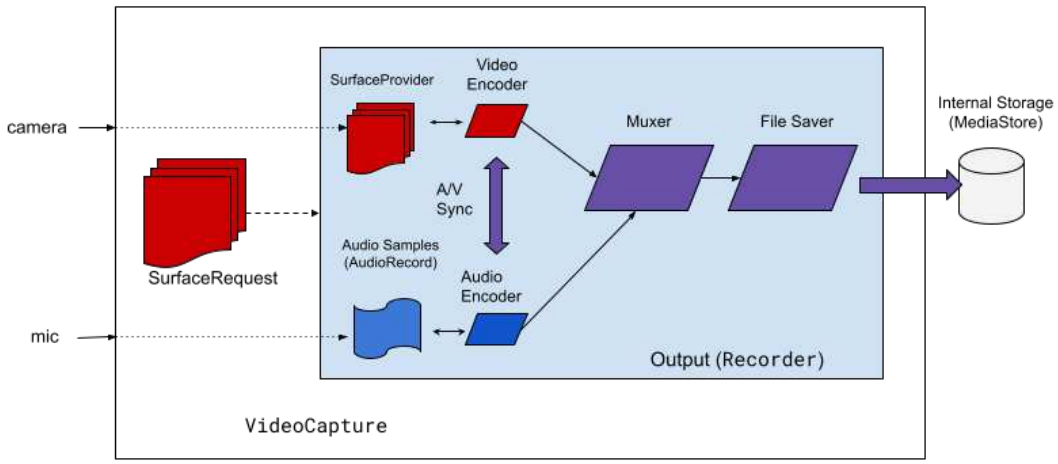
Slika 2.3. Formati slike YUV i RGB

2.2.3. Snimanje slike

Snimanje slike onsovano je na `ImageCapture` elementu koji ima način rada za smanjenje odazivnog vremena i način rada za maksimalnu kvalitetu slike. Glavna fukncionalnost snimanja slike ostvarena je metodom `takePicture` koja podržava JPEG format izlazne slike.

2.2.4. Snimanje videozapisa

Snimanje videozapisa u CameraX knjižnici podjeljeno je na tok slike i tok zvuka koji su nakon kompresije spojeni i zapisani na disk (slika 2.4).



Slika 2.4. Snimanje videozapisa

`Recorder` objekt je osnova snimanja videozapisa. Pomoću objekta `QualitySelector` odabiremo kvalitetu videozapisa (4K ultra HD, full HD, HD ili SD). Zatim stvaramo `VideoCapture` objekt i njega vežemo uz životni ciklus aktivnosti. Metodom `start` počinje snimanje, dok metodama `pause`, `resume` i `stop` kontroliramo snimanje.

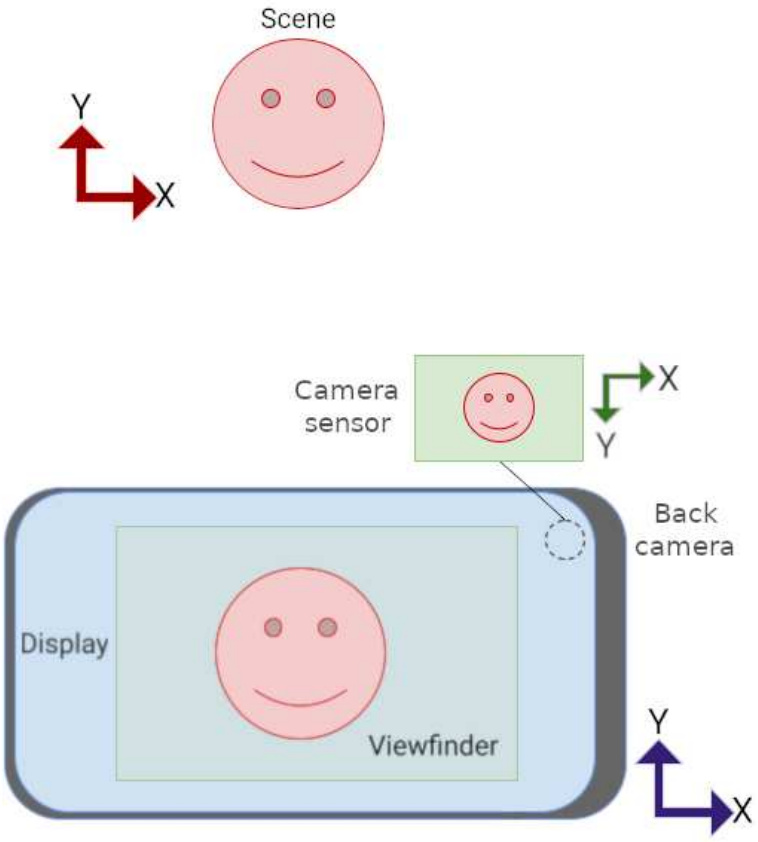
2.2.5. Rotacija

Pri snimanju videozapisa, snimanju slike ili analizi slike, rotacija kamere je važan čimbenik. Ovisno o rotaciji uređaja aplikacija treba odrediti je li željena snimka trebala biti u formatu portreta ili pejzaža. Također važno je osigurati da je slika uspravna bez obzira kako je uređaj kojim snimamo rotiran. U svrhu rješavanja tog problema pomažu nam dvije informacije: rotacija senzora i rotacija slike.

Rotacija senzora je fiksna vrijednost koja nam govori koliko je stupnjeva senzor kamere rotiran u odnosu na vrh uređaja u prirodnom položaju. Rotacija senzora može biti 0, 90, 180 ili 270 stupnjeva. Stražnje kamere mobilnih uređaja tipično imaju senzor stražnje kamere rotiran 90 stupnjeva.

Rotacija slike je potrebna rotacija u smjeru kazaljke na satu kako bi slika bila pravilno orijentirana, bez obzira na orijentaciju uređaja u trenutku snimanja.

Kako bismo osigurali da je slika uvijek pravilno prikazana, treba u obzir uzeti i rotaciju slike i senzora (slika 2.5).



Slika 2.5. Primjer rotacije

Rezultat analize ili snimanja slike je `ImageProxy` objekt koji osim same slike još sadrži neke informacije o slici.

Iz `ImageProxy` objekta možemo dobiti atribut `rotationDegrees` koji nam govori koliko stupnjeva u smjeru kazaljke na satu slika treba biti rotirana kako bi bila uspravna na ekranu uređaja.

Knjižnica CameraX pruža pruža niz raznolikih načina upravljanja kamerom uređaja, jednostavna je za korištenje i efikasno automatizira mnoge zadatke koji bi Android razvojnim programerima zadavali probleme. Zato je upravo CameraX često temelj Android aplikacija proširene stvarnosti.

2.3. Google ML Kit[2]

Pomoću knjižnice Google ML Kit, podatke prikupljene kamerom pretvorit ćemo u nama korisne informacije. Glavne alate ML Kita dijelimo na alate vizualne analize i alate obrade prirodnog jezika. Glavni alati vizualne analize prevođenje teksta, prepoznavanje lica, prepoznavanje poza tijela, prepoznavanje bar koda, označavanje slika, prepoznavanje i praćenje predmeta te mnogi drugi. U alate obrade prirodnog jezika spadaju prevođenje teksta, prepoznavanje jezika, pametno odgovaranje na poruke i ekstrakcija entiteta.

Alati ML Kita ne zahtijevaju internetsku vezu da bi funkcionirali, jednostavni su za upotrebu, besplatni su i vrlo su prilagodljivi potrebama programera.

U sklopu izrade aplikacije za pretvaranje eura u kunu korišten je Text recognition v2 alat.

2.3.1. Text recognition v2

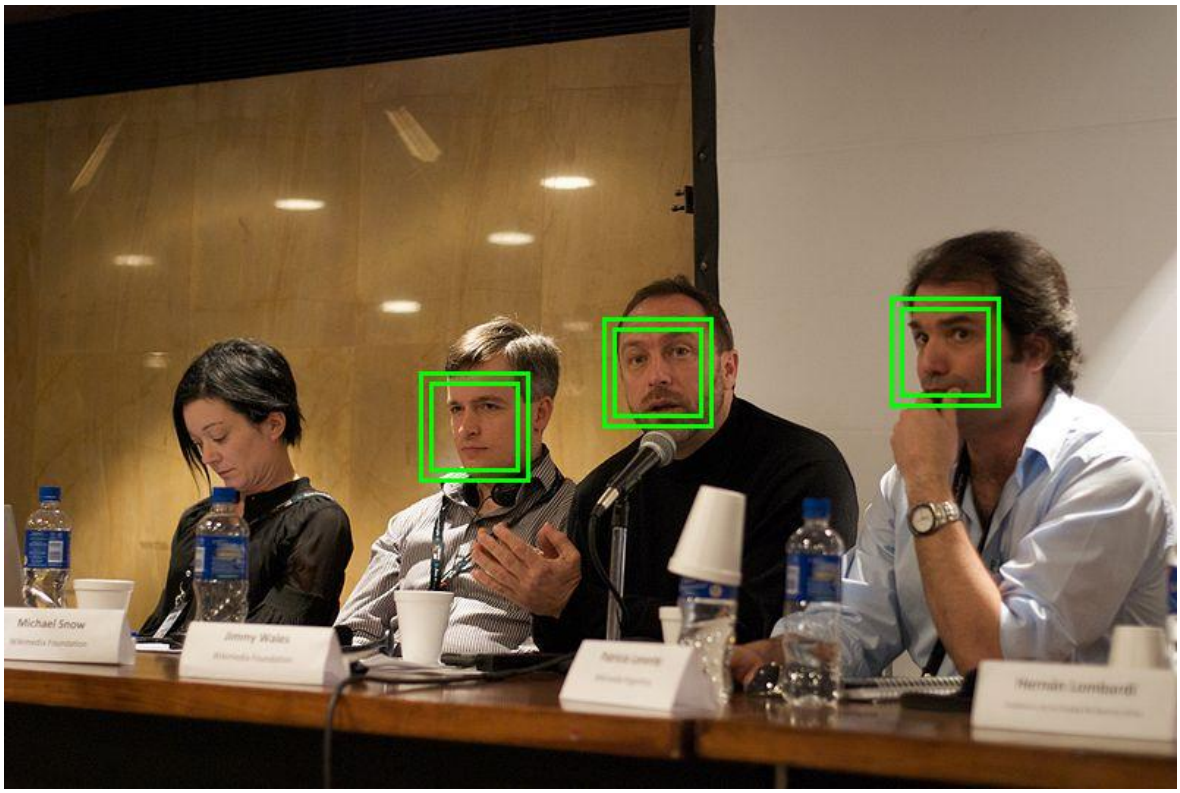
Glavna sposobnost alata Text recognition v2 je prepoznavanje strukturiranog i nestrukturiranog teksta u stvarnom vremenu. Alat prepoznaje jezik teksta te raspoznaje sve simbole i elemente teksta.

Blokovi teksta podijeljeni su na pojedinačne linije, dok je se svaka linije sastoji od elemenata teksta. Svaki element teksta je skup simbola i znakova.

Stvaranje instance `TextRecognition` počinje pozivom metode `getClient` s opcijama prevođenja teksta. Potrebno je pripremiti ulaznu sliku. Metoda `process` prima objekt tipa `InputImage` koji je moguće dobiti iz objekta tipa `media.Image`, `ByteBuffer` ili `Bitmap`. Nakon metode `process` objektu se dodaje slušač (engl. *listener*) na uspjeh prepoznavanja teksta. U tijelu slušača tekst iz slike se procesuira i spaja u `String`. Procesuiranje teksta sastoji se od iteracije po blokovima, linijama i elementima slike.

2.3.2. Face detection

Face detection alat (slika 2.6) prepoznaje crte lica, konturu lica, grimase. Alat je sposoban prepoznati i pratiti nečije lice u videozapisu.



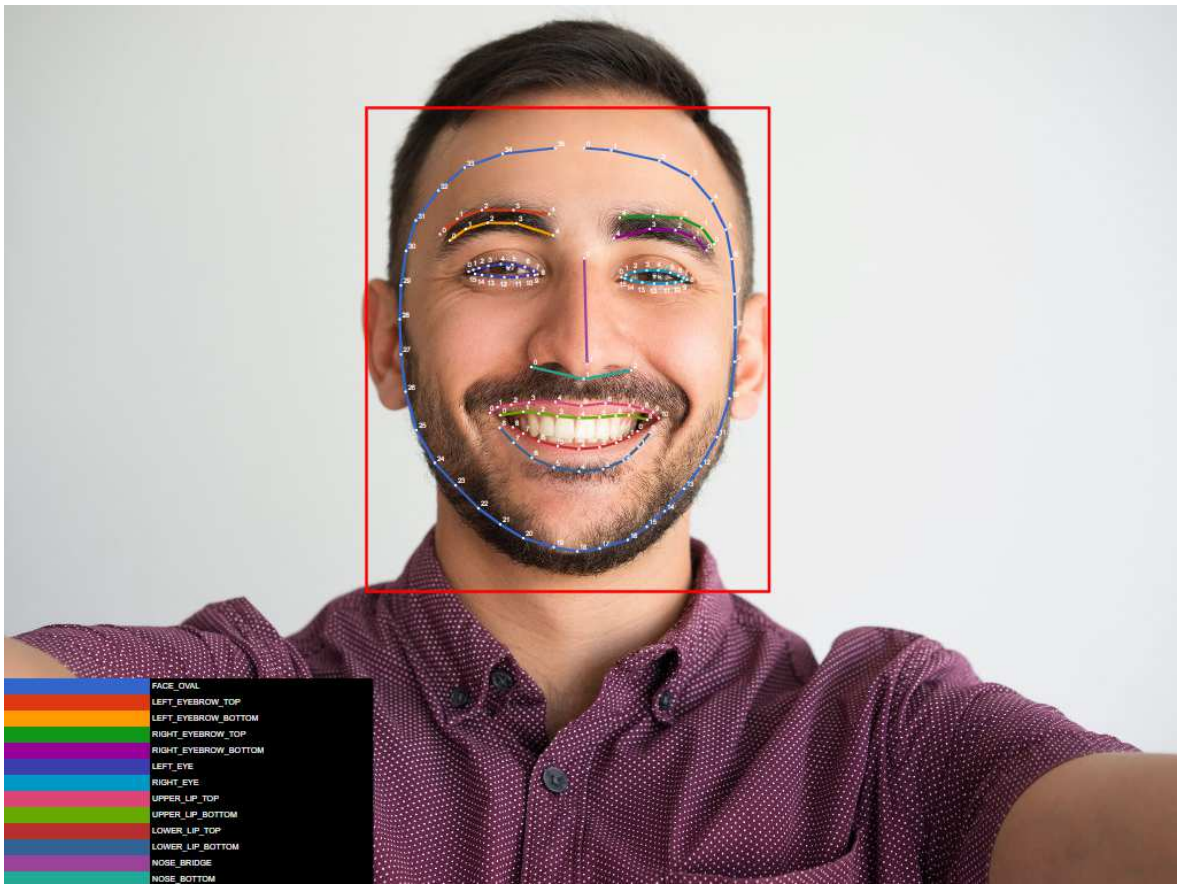
Slika 2.6. Prepoznavanje lica

Na slikama manje rezolucije prepoznavanje lice je brže, dok je na slikama veće rezolucije preciznije.

ML Kit pruža načine rada za performanse, prepoznavanje konture lica i klasifikaciju lica po grimasama i emocijama. Također je moguće postaviti minimalnu veličinu lica. Ako omogućimo praćenje lica u videozapisu, svakom licu dodijeljen je identifikacijski broj. U načinu rada za prepoznavanje konture lica samo jedno lice u slici je prepoznato, stoga nije preporučeno simultano prepoznavanje konture lica i praćenje lica.

Implementacija se osniva na `FaceDetector` objektu koji pruža metodu `process` na koju nadovezujemo slušače uspjeha ili neuspjeha.

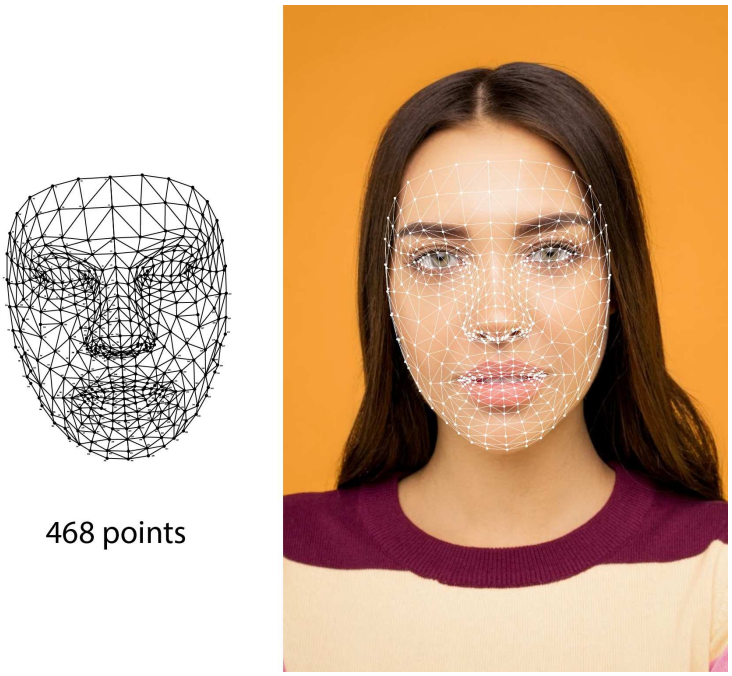
Ako je korišten način rada za prepoznavanje konture lica, nakon što je lice prepoznato, moguće je dohvatiti koordinate svih elemenata lica (uho, nos, lijevo oko itd.) pomoću `FaceLandmark` ili `FaceContour` elemenata.



Slika 2.7. Konture lica

2.3.3. Face mesh detection

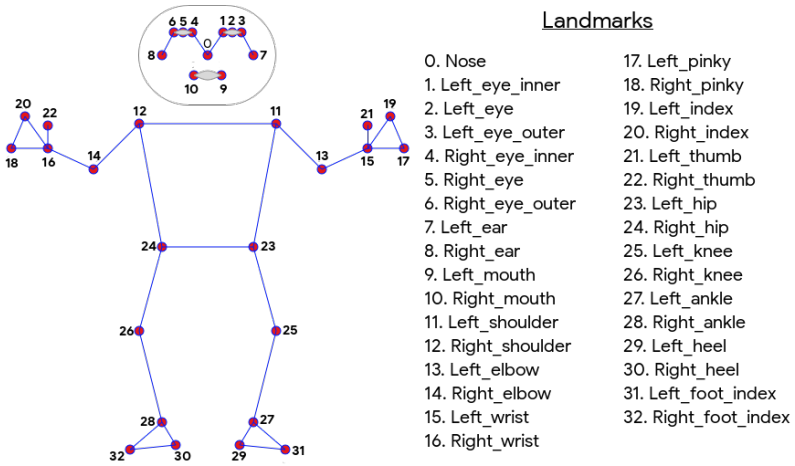
Razlikujemo Face detection alat od Face mesh detection alata. Face detection alat samo prepoznaje da se radi o licu koje može pratiti i na njemu odrediti neke značajke svih lica, dok Face mesh detection alat za prepoznato lice stvara mrežu (slika 2.8) od 468 točaka povezanih bridovima u prostoru. Pomoću mreže lica, alat je sposoban prepoznati o kojoj se osobi na slici ili videozapisu radi.



Slika 2.8. Mreža lica

2.3.4. Pose detection

Detekcija poze još je jedan od ML Kit alata namijenjenih za aplikacije proširene stvarnosti. Na slici ili snimci prepoznato je ljudsko tijelo. Ključnim dijelovima tijela dodijeljene su 33 točke (slika 2.9) u prostoru pomoću kojih određujemo pozu tijela (slika 2.14).



Slika 2.9. Ključne točke tijela

Nakon što su točke tijela prepoznate, tijelo je moguće pratiti kroz sve pokrete na videozapisu ili slici. Na pozi tijela moguće je odrediti koordinate dijelova tijela. Korištenjem tih vrijednosti određujemo interpretaciju poze. Interpretirajući različite poze moguće je pratiti pravilnost vježbi ili bilo kojih fizičkih aktivnosti koje uključuju ljudsko tijelo.

Slika 2.10. Različite poze

2.3.5. Barcode scanning

Barkod ili crtični kod je način označavanja proizvoda ili predmeta koji je strojno lako čitljiv. Crne linije na barkodu upijaju svjetlost dok bijele linije reflektiraju svjetlost. Barkod čitač pretvara različite razine reflektirane svjetlosti u niz nula i jedinica.

Barcode scanner alat sposoban je čitati barkodove u linearnom (slika 2.11) i 2D formatu (slika 2.12).

Slika 2.11. Linearni barkodovi

2D barkodovi mogu sadržavati mnogo više informacija od linearnih barkodova te imaju ugrađen sustav za automatsko ispravljanje grešaka. Najpoznatiji barkod u 2D formatu je QR kod.

Slika 2.12. 2D barkodovi

Alat dozvoljava prepoznavanje barkoda u proizvoljnom formatu, no ako specificiramo željeni format barkoda brzina prepoznavanja bit će značajno veća. Rotacija barkoda i ekstrakcija podataka obrađeni su automatski.

2.3.6. Image labeling

Alat označavanja prepoznaje entitete u slikama. ML Kit pruža svoj istrenirani model za klasifikaciju preko 400 entiteta. Dostupna je i opcija uvođenja svog modela u alat označavanja. Alat svakoj slici pridjeljuje niz oznaka koje opisuju tu sliku. Uz svaku oznaku dostupna je i *Confidence* vrijednost koja nam u postotku govori koliko je alat siguran u točnost oznake.

2.3.7. Object detection and tracking

ML Kit knjižnica sadrži još jedan alat za prepoznavanje objekata u slici (slika 2.13). Object detection and tracking alat prepoznatim entitetima određuje točnu lokaciju u slici i dodjeljuje im ograničavajući okvir (engl. *bounding box*). Alat također entitetima dodjeljuje klasu.

Glavna razlika Image labeling i Object detection alata je u preciznosti klasifikacije objekata. Alat za detekciju entitetima u slici dodjeljuje samo opću kategoriju, dok alat za označavanje ne određuje lokaciju entiteta u slici.

Slika 2.13. Detekcija entiteta u slici

2.3.8. Alati obrade prirodnog jezika

Prepoznavanje jezika

Alat prepoznavanja jezika za ulazni tekst na latinici određuje o kojem se jeziku radi.

Prevođenje

Alat za prevođenje prevodi zadani tekst na drugi jezik. Za korištenje alata za prevođenje nužno je znati jezik izvornog teksta (engl. *source language*) i jezik u koji želimo prevoditi (engl. *target language*).

Ako ne znamo o kojem se jeziku izvornog teksta radi potrebno je prethodno koristiti alat za prepoznavanje jezika.

Implementacija se ostvaruje kroz objekt `TranslatorOptions` na kojemu konfiguriramo opcije jezika te metodu `translate` koju pozivamo nad `Translator` objektom.

Smart reply

Alat za pametno odgovaranje na poruke na osnovi priloženog razgovora između dva govornika pruža odgovor na svaku nadolazeću poruku. Prethodan razgovor potreban je alatu kako bi osmislio relevantan i dosljedan odgovor. Model predlaže tri odgovora te korisnik odabire njemu najprikladniji.

Entity extraction

Alat za ekstrakciju entiteta u ulaznom tekstu pronalazi entitete za koje korisnik može definirati uobičajene akcije. Tipični entiteti u tekstu su: adresa, datum, vrijeme, e-mail adresa, bankovne informacije, novčane valute, telefonski brojevi hiperveze te mnogi drugi. Korisnik može definirati akcije poput klika na hiperveze, pretrage adrese, poziv na telefonski broj i slično.

3. Implementacija aplikacije

Aplikacija proširene stvarnosti za pretvorbu cijena u hrvatsku kunu koristi knjižnicu CameraX kako bi implementirala pregled kamere i analize niza nadolazećih slika.

Aplikacija je podijeljena na klase `MainActivity`, `CameraAdapter`, `MyImageAnalyzer` i `MyTextRecognizer`.

Klasa `CameraAdapter` sadrži kod za pokretanje kamere te inicijalizaciju `ImageAnalyzer` objekta. Kao implementaciju sučelja `CameraProvider` koristimo razred `ProcessCameraProvider`. Metodom `ProcessCameraProvider.getInstance` dohvaćamo instancu klase `ProcessCameraProvider` koji upravlja životnim ciklusom, veže obrasce uporabe te osigurava da su sve operacije izvedene na siguran način na dretvama, što je važno za istovremeni pristup hardveru kamere.

```
fun startCamera(context: Context, lifecycleOwner: LifecycleOwner, surfaceProvider: Preview.SurfaceProvider){
    val cameraProviderFuture = ProcessCameraProvider.getInstance(context)
    var runnable = Runnable {
        val preview = Preview.Builder() Preview.Builder
            .build() Preview
            .also { it.setSurfaceProvider(surfaceProvider) }
        with(cameraProviderFuture.get()) { this: ProcessCameraProvider!
            unbindAll() // unbind previous use cases
            bindToLifecycle(
                lifecycleOwner,
                CameraSelector.DEFAULT_BACK_CAMERA,
                preview,
                imageAnalzyer
            )
        }
    }
    cameraProviderFuture.addListener(runnable, ContextCompat.getMainExecutor(context))
}
```

Slika 3.1. Pokretanje kamere

Objektu tipa `ImageAnalyzer` pridjeljuje se objekt `ImageAnalyzerExecutor` koji osigurava da se analiza slike odvija u odvojenoj dretvi. Osim pregleda kamere, analiza slike također je vezana uz životni ciklus aktivnosti (slika 3.2).

```
private val imageAnalyzerExecutor: ExecutorService by lazy { Executors.newSingleThreadExecutor() }
private val imageAnalyzer by lazy {
    ImageAnalysis.Builder() ImageAnalysis.Builder
        .setTargetAspectRatio(AspectRatio.RATIO_16_9)
        .setOutputImageFormat(ImageAnalysis.OUTPUT_IMAGE_FORMAT_YUV_420_888)
        .build() ImageAnalysis
        .also { it: ImageAnalysis
            it.setAnalyzer(
                imageAnalyzerExecutor,
                MyImageAnalyzer(onTextFound)
            )
        }
}
```

Slika 3.2. Image Analyzer

Za potrebe aplikacije koristit ćemo YUV_420_888 format slike, kako bi jednostavno pretvorili sadržaj slike u `Bitmap` objekt čiju veličinu možemo proizvoljno mijenjati.

Klasa `MyImageAnalyzer` sadrži metode `convertImageToBitmap` i `analyze`. Metoda `analyze` pozvana je na svakoj slici iz pregleda kamere. Također, u metodi `analyze`, korištenjem `convertImageToBitmap` slika u YUV formatu pretvorena je u `Bitmap` objekt (slika 3.3) nad kojim izvodimo rezanje (engl. *crop*). Iz slike režemo dio na kojemu ćemo raditi analizu i na kojemu će biti pravokutnik fokusa.

```
private fun convertImageToBitmap(image: Image): Bitmap {
    val yBuffer = image.planes[0].buffer // y
    val vuBuffer = image.planes[2].buffer // vu
    val ySize = yBuffer.remaining()
    val vuSize = vuBuffer.remaining()
    val nv21 = ByteArray( size: ySize + vuSize)
    yBuffer.get(nv21, offset: 0, ySize)
    vuBuffer.get(nv21, ySize, vuSize)
    val yuvImage = YuvImage(nv21, ImageFormat.NV21, image.width, image.height, strides: null)
    val outStream = ByteArrayOutputStream()
    yuvImage.compressToJpeg(Rect( left: 0, top: 0, yuvImage.width, yuvImage.height), quality: 50, outStream)
    val imageBytes = outStream.toByteArray()
    return BitmapFactory.decodeByteArray(imageBytes, offset: 0, imageBytes.size, opts: null)
}
```

Slika 3.3. Pretvorba slike u YUV formatu u `Bitmap` objekt

Na kraju metode `analyze` pozivamo metodu `recognizeImageText` iz klase `MyTextRecognizer`. `MyTextRecognizer` koristi ML Kit knjižnicu i alat prepoznavanja teksta. Metoda `recognizeImageText` gradi `TextRecognition` objekt, obrađuje tekst u slici i poziva metodu `processTextFromImage` (slika 3.4). `processTextFromImage` strukturira tekst pronađen u slici i pretvara ga u `String` format.

```
fun recognizeImageText(inputImage: InputImage, rotationDegrees: Int, onResult: (Boolean) -> Unit) {
    TextRecognition.getClient(TextRecognizerOptions.DEFAULT_OPTIONS) TextRecognizer
        .process(inputImage) Task<Text!>
        .addOnSuccessListener { recognizedText ->
            processTextFromImage(recognizedText)
            onResult(true)
        }
}
```

Slika 3.4. Ekstrakcija teksta

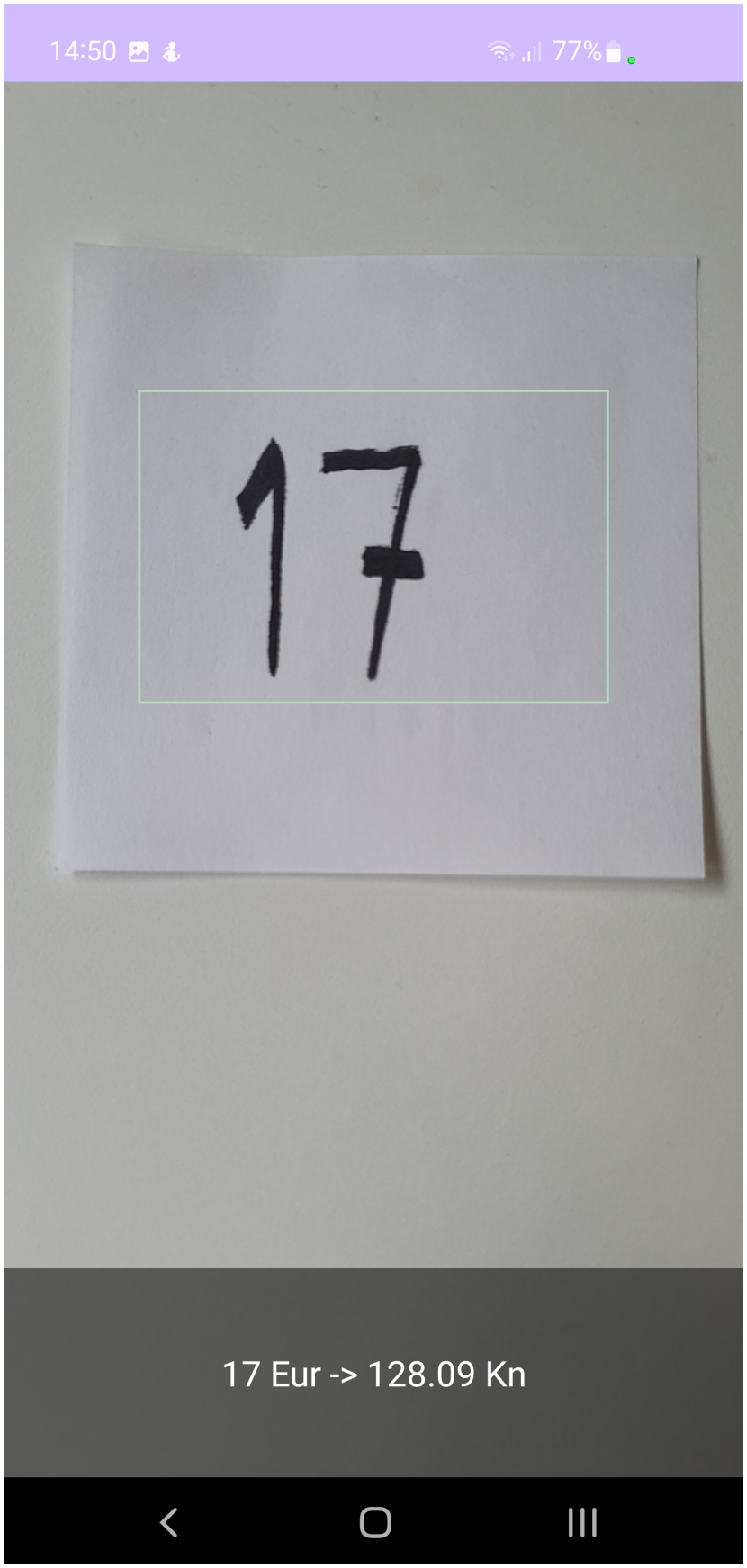
U glavnoj aktivnosti aplikacije (klasa `MainActivity`) logika aplikacije povezana je s XML elementima. Metoda `drawFocusRect` na zaslonu crta pravokutnik fokusa.

Kako bi aplikacija koristila kameru, nužno je da se od korisnika uređaja zatraži dozvola za korištenje hardvera uređaja koji će koristiti. Korištenjem pomoćnog razreda `ContextCompat` provjeravamo jesu li ispunjene sve dozvole za pokretanje aplikacije.

```
private val isAllPermissionsGranted get() = REQUIRED_PERMISSIONS.all { it: String
    ContextCompat.checkSelfPermission( context: this, it) == PackageManager.PERMISSION_GRANTED
}
```

Slika 3.5. Provjera dozvola

Stvaranjem objekta `CameraAdapter` glavna aktivnost povezana je s dijelom aplikacije namijenjenim za obradu i prikaz slike. Pozivom metode `startCamera` nad objektom `CameraAdapter` pokrećemo kameru. U lambda izrazu objekta `CameraAdapter` dobivamo niz znakova koji je izvučen iz slike. Iz niza znakova u `String` formatu regularnim izrazima probrane su samostalne znamenke te znamenke s valutom. Obrađene znamenke i valute prikazuju se na zaslonu uređaja.



Slika 3.6. Demonstracija funkcionalnosti

4. Zaključak

Aplikacije proširene stvarnosti omogućuju preklapanje digitalnih informacija s fizičkim svijetom.

Fizički svijet predočen je pomoću raznih senzora uređaja, a najčešće je to kamera. Knjižnice poput CameraX ili Camera2 dozvoljavaju aplikaciji pristup kameri uređaja i pružaju visoku razinu prilagodljivosti i optimizacije performansi. Digitalni aspekt aplikacija proširene stvarnosti omogućen je korištenjem knjižnica poput Google ML Kit koje koriste napredne algoritme strojnog učenja za implementaciju OCR funkcionalnosti u aplikaciju.

Kroz praktični dio rada implementirali smo obrasce uporabe pregled kamere i analizu slike korištenjem knjižnice CameraX. Implementacijom Text recognizer v2 alata iz Google ML Kit knjižnice ostvarili smo OCR funkcionalnost i proširili prikazanu stvarnost.

Razni alati ML Kit-a poput prepoznavanja lica, poza, predmeta i bar kodova demonstriraju raznolikost i ekspresivnost aplikacija proširene stvarnosti.

Budući razvoj i poboljšanja u tehnologijama proširene stvarnosti pomiču granice inovativnih aplikacija te unapređuju korisničko iskustvo.

Literatura

[1] Android, https://developer.android.com/media/camera/camerax, [mrežno; stranica posjećena: lipanj 2024.].

[2] Google, https://developers.google.com/ml-kit, [mrežno; stranica posjećena: lipanj 2024.].

[3] D. i. P. Krevelen, "A survey of augmented reality technologies, applications and limitations", *The International Journal of Virtual Reality*, sv. 9, br. 2, str. 1–20, 2010.

Sažetak

Aplikacija proširene stvarnosti za pretvorbu cijena u hrvatsku kunu na uređajima s operacijskim sustavom Android

Karlo Kuzle

Proširena stvarnost obogaćuje svijet digitalnim sadržajem te korisnicima pruža interaktivno iskustvo. Aplikacije proširene stvarnosti na uređajima s operacijskim sustavom Android najčešće kombiniraju funkcionalnost kamere uređaja s raznim OCR tehnologijama. Funkcionalnosti kamere implementirane su knjižnicama poput CameraX, dok alati OCR tehnologije donose knjžnice poput Google ML Kita.

Ključne riječi: Android; proširena stvarnost; CameraX; Google ML Kit

Abstract

An augmented reality application for converting prices into Croatian kuna on Android devices.

Karlo Kuzle

Augmented reality enriches the world with digital content, providing users with an interactive experience. Augmented reality applications on devices with the Android operating system most commonly combine the functionality of the device's camera with various OCR technologies. The camera functionalities are implemented through libraries such as CameraX, while OCR technology tools are brought by libraries such as Google ML Kit.

Keywords: Android; augmented reality; CameraX; Google ML Kit