eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Informatika Ingeniaritzako Gradua

Konputazioa

Gradu Amaierako Proiektua

Digitalizazio Automatikoa: Iruditik Testura

Egilea

Pello Arrue Aldalur

informatika fakultatea
facultad de informática

2017

Laburpena

Hurrengo orrietan, paperean inprimaturik dauden testuen digitalizazio automatikoa hobetzen saiatuko da. Hobekuntza hori hiru modutara egin daiteke: Irudiak aurreprozesatuta, OCR[1] [OCR, 2017] algoritmoak hobetuta edo behin digitalizatuta dagoen testua hiztegien bidez zuzenduta. Proiektuan zehar, irudien aurreprozesatzean sakonduko da gehien.

Hasteko, hobekuntza hauek burutzeko beharrezkoak diren aurrekari eta teknologiak azalduko dira. Ondoren, itxura eta hizkuntza desberdineko testu-irudiekin esperimentazioa burutuko da. Segidan, proiektuan egin den aplikazioaren garapenaren azalpenak.

Bukatzeko, proiektuan zehar lortutako ondorioak azalduko dira, baita, proiektuari etorkizunean egin ahal zaizkion hobekuntzak ere.

[1] Optical Character Recognition

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

1. KAPITULUA

Sarrera

Gaur egun testu digitala indar handia hartzen ari da. Horrenbestez gero eta interes handiagoa dago paperean inprimatuak dauden testuak digitalizatzean. Proiektu hau horretan zentratuko da, testuen irudiak izanik, modu automatiko batean, irudi horiek testu digital bihurtzean alegia. Hau burutzeko OCR [1] teknika inplementatuta daukan *Tesseract* aplikazio libreaz baliatuko da. Horrela software librearen ideologia jarraituz, jadanik inplementatuta dagoena aprobetxatuko da, eta honen gainean hobekuntzak egiten saiatu.

Proiektu honetan bi teknika erabili dira aplikazioari hobekuntzak egiteko: Irudiak aurreprozesatzea eta hiztegien bidez testu digitala zuzentzea.

Irudien aurreprozesatzeari dagokionez, proiektu honen zatirik handiena hartzen duen teknika da. Bertan, irudiei eraldaketak aplikatzen zaizkie, adibidez, ertzak nabarmendu, irudia normalizatu... Aplikazioak testua digitalizatzean akats gutxiago egiteko helburuarekin.

Hiztegiei dagokionez berriz, aplikazioak behin testua digitalizatu duenean, hau zuzentzea da helburua. Horretarako, hiztegi digital batez baliatzen da, horrela, hitza hiztegian aurkitzen bada zuzentzat hartuz eta hitza hiztegian ez badago, hiztegiak berak egiten duen iradokizunagatik aldatuz.

[1] Optical character recognition

Motibazioa

Paperean soilik idatzita dauden milioika testu digitalizatzea ez da lan erraza. Testu horiek digitalizatzeko bi modu daude, pertsonek testuan ikusten dutena ordenagailuan tekleatzen joatea edo adimen artifizialeko teknikez baliatzea. Lehenengo teknikak akats gutxiago sortzen baditu ere, oso makala da. Pentsa zenbat pertsona eta denbora beharko liratekeen adibidez, 500 orrialdeko 100 liburu digitalizatzeko. Adimen artifizialeko tekniken bidez, lan guzti hori konputagailuak burutzen du. Testua askoz azkarrago digitalizatzea lortzen bada ere, lortutako testua ez da guztiz zuzena izaten. Proiektu honetan adimen artifizialeko teknika horiek ezagutu, ikertu eta hobetzen saiatuko da.

Proiekturen zati handi bat irudien aurreprozesatzean oinarrituko dela esan da. Baina digitalizatu nahi diren testu-irudi denentzat aurreprozesatze egoki bat aurkitu daitekeen edo bakoitzak aurreprozesatze propioa behar duen begiratu beharko da. Irudien eraldaketak beraien artean konbinatzeko aukera ere badagoenez, konbinaketa egokiak aurkitzeko bilaketa heuristiko bat ere garatu da. Hori guztia esperimentatu eta ondorio egokiak lortzea izango da helburu nagusia.

Testua aurreprozesatuta izanda ere, digitalizazioa ez da perfektua izango ordea. Horrenbestez hiztegi digitalak baliatuz testua zuzentzea da bigarren helburua.

Bukatzeko, aurreko paragrafoetan aipaturiko guztia aplikazio informatiko batean bilduko da.

2. KAPITULUA

Aurrekariak eta Teknologia

Atal honetan proiektuaren helburuak bete ahal izateko eginiko azterketa azalduko da.

Hasteko, proiekturen helburu nagusia irudien tratamendu egoki bat lortzea denez, *Irudien Tratamendua* izeneko azpiatala dago. Bertan, irudien zenbait transformazio burutzeko erabiltzen diren eredu matematikoak gordeko dira. Baita, irudien aurreprozesatze egoki bat lortzeko teknikak ere.

Ondoren, aplikazioak irudien aurreprozesatzea zein hiztegien bidezko zuzenketa egiteko behar duen softwarea esplikatuko da, *Aplikaziorako Softwarea* izeneko azpiatalean.

Bukatzeko, aplikazioak funtzionatu ahal izateko oinarrizko softwarea azalduko da. Baita aurrekarietan aipatu diren programa guztiek dituzten lizentzia desberdinak eta hauek instalatzeko argibideak ere.

Irudien Tratamendua

Irudien tratamendua, oso atal garrantzitsua da konputagailu bidezko ikusmenean. Atal honetan, irudiei eraldaketak aplikatzen zaizkie, adibidez, ertzak nabarmendu, irudia normalizatu, etab. ondoren adimen artifizialeko teknikak erabiltzerakoan, emaitzak hobetzeko helburuarekin.

Irudien Transformazioak atalean, irudiei aplikatu ahal zaizkien eraldaketa ezberdinak azalduko dira. Ondoren, irudiekin lan egiteak kostu konputazional handia duenez, bilaketa

heuristikoaren nondik norakoak azalduko dira. Bilaketa mota hau, irudien aurreprozesatze egoki bat lortzeko erabiliko da proiektuan.

Irudien Transformazioak

Atal honetan, irudien tratamendurako erabiliko diren eraldaketa ezberdinen azalpenak eta hauek gauzatzeko eredu matematikoak azalduko dira.

Kontrastea Aldatzea

Kontrastea, irudi baten intentsitate handieneko pixelaren eta instentsitate txikieneko pixelaren arteko aldea bezala definitu dezakegu.

$$kontrastea = max(pixelenIntentxitatea) - min(pixelenIntentsiatea)$$

Beraz, kontrastea handitu edo txikitzeko, nahikoa da pixel intentsitateen arteko aldea handitzea edo txikitzea.[Kontrastea, 2017]

Normalizazioa

Normalizazioa, kontrastea aldatzeko modu partikular bat da. Transformazio honen helburua, pertsonen zentzumenen bidez irudia hobeto edo errazago ikustea da.

Normalizazioaren bidez, n dimentsioko gris-eskaladun irudia $I : \{\mathbb{X} \subseteq \mathbb{R}^2\} \rightarrow \{Min, .., Max\}$ intentsitate balioak (Min,Max) tartean dituena, $I_N : \{\mathbb{X} \subseteq \mathbb{R}^2\} \rightarrow \{newMin, .., newMax\}$ irudian eraldatuko da, intentsitate balioak (newMin, newMax) tartera pasatuko direlarik.

Gris-eskaladun irudi baten normalizazio lineala ondorengo formularen bidez burutzen da

$$I_N = (I - Min)\frac{newMax - newMin}{Max - Min} + newMin$$

Normalizazioa ez lineala ere izan daiteke, hau I eta I_N -ren artean antzekotasun linealik ez dagoenean gertatzen da. Ondorengo formula honen bidez burutzen da

$$I_N = (newMax - newMin)\frac{1}{1 + e^{\frac{I-\beta}{\alpha}}} + newMin$$

non, α sarrerako intentsitateen tartea den eta β tarte hori non dagoen zentratuta.[Normalizazioa, 2017]

Negate

Negate, irudi bateko koloreak aurkako koloreagatik aldatzean datza. Adibidez pixel bat beltza bada zuriagatik aldatzea eta zuria bada beltzagatik.

Irudia RGB balioen bidez definituta badago, pixel bakoitzaren balio berriak honela kalkulatzen dira:

$$R_{new} = 255 - R$$
$$G_{new} = 255 - G$$
$$B_{new} = 255 - B$$

Ertzen Detekzioa

Ertzen detekzioan bi teknika nagusi bereiz daitezke: [Mugarza, 2016]

- **Canny:** Teknika honen oinarria, gris eskalan dauden irudien pixelen intentsitate aldaketa antzematea da. Hau da, ondoz ondoko pixel multzo txiki bat badugu, adibidez, 3x3 ingurune bat osatzen duten pixel multzoa, eta multzo horretako pixelek gris intentsitate antzekoa badute, irudiaren objektu bereko pixelak direla pentsatzea logikoa litzateke. Bestalde, pixel multzo horretan gris intentsitate aldaketak badaude logikoa da puntu horiek ertz batekoak direla pentsatzea, hots, hautatutako ingurunean bi objektu desberdineko pixelak daudela pentsatzea.

- **Autokorrelazio Espaziala:** Teknika honetan autokorrelazio espazialaren bidez ertzak bilatuko dira. Izan bedi $\Omega = \{\omega_i \mid i = 1,..,n\}$ espazialki egituratutako multzoa, kasu honetan irudi baten pixelak, orduan, autokorrelazio espaziala Ω maparen antolaketa eredua aztertzean datza. Puntu batean aztertutako egituraren magnitudearen erlatiboki altua (baxua) bada, kasu honetan pixelen gris intentsitatearen balioa izango zena, eta bere inguruneko magnitudeen balioak ere altuak (baxuak) badira, orduan autokorrelazioa positiboa izango litzateke. Aldiz, aztertutako kokapen zehatz batean magnitudearen balioa erlatiboki altua (baxua) bada eta magnitude horren balio baxuz (altuz) inguratuta badago, autokorrelazioa negatiboa dagoela esan daiteke.

Gaussian-Blur

Gaussian-Blur, irudi bat funtzio Gauss-tarraren bidez lausotzearen emaitza da. Normalean, irudiaren zarata murrizteko erabilia.

Honakoa da bi dimentsioko funtzioa Gauss-tarraren ekuazioa:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

non, x ardatz horizontalaren jatorritik dagoen distantzia eta x ardatz bertikalaren jatorritik dagoen distantzia diren. σ berriz, banaketa Gauss-tarraren desbideratze estandarra da. Ekuazio honetan lortutako balioak transformazio matrize bat eraikitzeko erabiltzen dira, ondoren irudi originalari aplikatuko zaiona.[Gaussian-Blur, 2017]

Local Adaptive Thresholding

Thresholding teknika irudia binarizatzeko erabili oi da. Horrela atalase baten azpitik dauden pixelak eta atalasearen goitik dauden pixelak sailkatuz. *Thresholding* teknikan, atalasea(g) irudi guztiarentzat berdina izango litzateke.

$$T_{global}(g) = \begin{cases} 0 \quad baldin \quad g < t \\ \\ 1 \quad baldin \quad g \geqslant t \end{cases}$$

Local Adaptive Thresholding teknikan berriz, irudia zatika prozesatzen da, zati bakoitzean bertako atalasea kalkulatuz. Atalse hori zati horretako pixelen batezbestekoa, mediana edo pixel maximoaren eta minimoaren arteko batezbestekoa izan daiteke. [Lat, 2003]

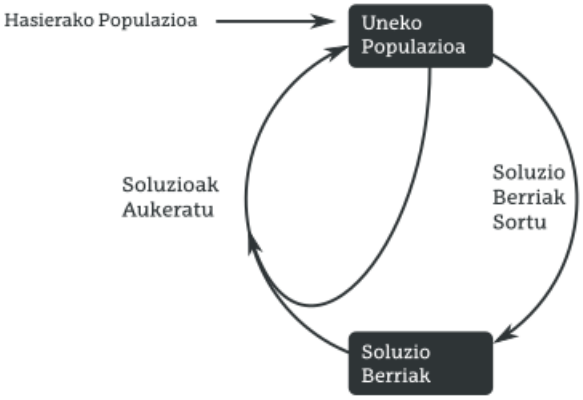
Bilaketa Heuristikoa

Proiektu honetan Bilaketa Heuristikoa erabiliko da irudien transformazioen sekuentzia bat lortzeko.

Hasiera batean, 8 irudi transformazioko sekuentziekin probak egitea pentsatu zen. Baina horrek arazo bat du. Pentsa hamar transformazio desberdin daudela eta zortzi transformazioko sekuentziak osatu nahi badira, $10^8 = 100000000$ irudi desberdinekin egin beharko dela proba. Hori egiteko denbora asko behar denez, bilaketa heuristiko bat egitea erabaki

zen. Horrela transformazioen sekuentziarik hoberena lortuko dela ziurtatzen ez bada ere, soluzio on bat aurkituko da, denbora askoz gutxiago batean.

Bilaketa heuristikoa oso gai zabala bada ere, algoritmo ebolutibo bat erabiltzea erabaki da. Algoritmo hauek oso erabiliak dira. Alde batetik, emaitza onak lortzen dituztela ikusi da eta bestetik beraien inplementazioa nahiko erraza da. 2.1 irudian ikus daiteke eskema orokorra [Borja Calvo, 2017]



2.1 Irudia: Algoritmo Ebolutioboaren Eskema

Algoritmo Genetikoa

Algoritmo genetikoak algoritmo ebolutiboen adibide ezagunenak dira. Naturan gertatzen den espezieen eboluzioa imitatuz, algoritmo genetikoen osagaiak naturako fenomeno horren harira definitzen dira:

- Espezie bateko indibiduoak = Problemaren soluzioak
- Indibiduoen egokitasuna (fitnessa, alegia) = Soluzioaren ebaluazioa
- Espeziearen populazioa = Soluzio multzoa/populazioa
- Ugalketa = Soluzio berrien sorkuntza

Algoritmo genetikoetan soluzio berriak sortzeko estrategiak diseinatzean naturako indibiduoen ugalketa-prozesuan oinarritzen da. Ugalketa prozesuaren xedea zenbait indibiduo emanda (bi gehienetan), indibiduo berriak sortzea da. Ohikoena prozesu hori bi pausotan banatzea da: soluzioak gurutzatzea eta mutatzea. Lehenaren helburua guraso so-

luzioek dituzten ezaugarriak (geneak) soluzio berriei pasatzea da, espezieen gurutzaketan jasotzen den bezala. Bigarrenarena, berriz, sortutako soluzio berriei, semeei, ezaugarri berriak eranstea da. 2.2 irudian ikusi daiteke algoritmo genetikoen eskema orokorra. [Populazioan Oinarritutako Algoritmoak, 2017]

2.2 Irudia: Algoritmo Genetikoen Eskema Orokorra

Algoritmo genetikoak bost pausotan banatu daitezke:

- Populazioaren hasieraketa

  Nahiz eta askotan garrantzi gutxi eman, hasierako populazioa da algoritmoaren abiapuntua eta, hortaz, beraren sorkuntza oso pauso garrantzitsua da, eragin handia izaten baitu lortutako azken emaitzan. Algoritmoen xedea soluzio onak topatzea denez, pentsa daiteke hasierako populazio on batek soluzio onez osatuta egon behar duela. Alabaina, soluzioen dibertsitatea haien kalitatea bezain garrantzitsua da. Populazioa antzekoak diren soluzioez osatuta badago, orduan horren eboluzioa oso zaila izango da, eta algoritmoak azkarregi konbergitu dezake optimoa ez den soluzio batera.

  Hortaz, hasierako populazioa sortzean bi alderdi izan behar dira kontuan: kalitatea eta dibertsitatea.

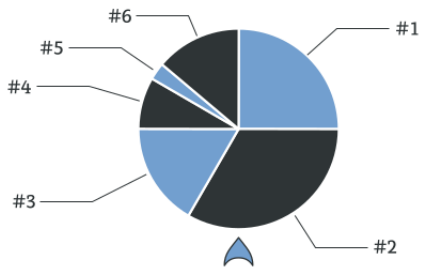
- Hautespena

  Algoritmo ebolutiboetan soluzioen hautespena izango da seguruenik urratsik garrantzitsuena, horrek kontrolatzen baitu populazioaren eboluzioa. Orokorrean, po-

pulazioan dauden soluziorik onenak hautatzea da gehien erabiltzen den hautespen-irizpidea hautespen *elitista* da. Alabaina, soluzio onak aukeratzea garrantzitsua bada ere, dibertsitatea mantentzearren, tarteka soluzio ez hain onak sartzea ere komenigarria izaten da. Teknika hori zuzenean aplikatu daitekeen arren, badaude aukeraketa metodo egokiago batzuk soluzio txarrak estrategia probabilistikoak erabiliz aukeratzen dituztenak.

Erruleta-hautespena[1], deritzon estrategian soluzioak erruleta batean kokatzen dira. Soluzio bakoitzari bere ebaluazioarekiko proportzionala den erruletaren zati bat esleituko zaio. Hori horrela, 2.3 irudian ikus daitekeen bezala, erruleta jaurtitzen den bakoitzean indibiduo bat hautatzen da. Hautatua izateko probabilitatea erruleta zatiaren tamainarekiko eta, hortaz, indibiduoen ebaluazioarekiko proportzionala da. Indibiduo bat baino gehiago aukeratu behar baldin badira, behar adina erruleta-jaurtiketa egin daitezke.

Indibiduo	Ebaluzaioa
#1	899
#2	1204
#3	598
#4	313
#5	95
#6	500



2.3 Irudia: Erruleta-hautespena

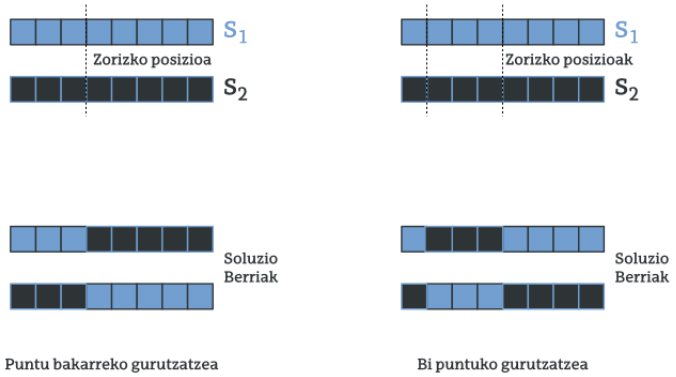
- Gurasoen gurutzaketa

  Bi soluzio (edo gehiago) gurutzatzen direnean euren propietateak sortutako soluzio berriei transmititzea da helburua. Optimizazio arloan, soluzioen arteko gurutzaketak gurutzaketa-operadore[2]-en bidez egiten dira. Operadore horiek soluzioen kodeketarekin dihardute, eta, beraz, gurutzaketa operadore zehatz bat hautatzean soluzioak nola adierazten diren aintzat hartu beharko da.

  Badaude kodeketa klasikoekin erabil daitezkeen zenbait oinarrizko gurutzaketa-operadore. Ezagunena puntu bakarreko gurutzaketa[3] deritzona da. Demagun solu-

[1] Roulette Wheel selection, ingelesez
[2] Crossover, ingelesez
[3] One-point crossover, ingelesez

zioak bektoreen bidez kodetzen direla. Bi soluzio/guraso, s_1 eta s_2 izanik, operadore horrek bi soluzio berri/seme sortzen ditu (2.4 irudian operadore horien adibide bat ilustratzen da). Horretarako, lehenik eta behin, zorizko posizio bat, i, aukeratu behar da. Hau egin ahala, lehenengo soluzio berria s_1 soluziotik lehenengo i elementuak eta s_2 soluziotik gainontzekoak ($i+1$-tik aurrerakoak) kopiatuz sortuko da. Era berean, bigarren soluzio berria s_2-tik lehenengo i elementuak eta s_1-etik $i+1$ posiziotik aurrerako elementuak kopiatuz sortuko da. 2.4 irudiaren ezkerraldean, puntu bakarreko gurutzaketa operazioaren aplikazioaren adibide bat ikus daiteke. Horrez gain, eskuinaldean operadore hori nola orokortu daitekeen erakusten da, puntu bakar bat erabili beharrean bi, hiru, etab. puntu erabiliz.



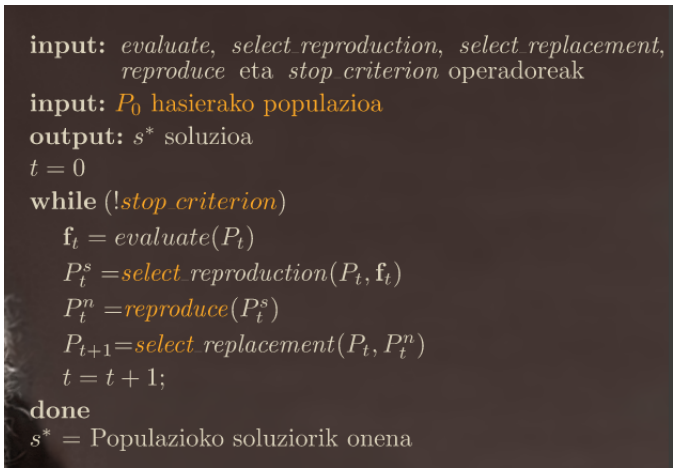
2.4 Irudia: Gurutzaketa

- Mutazioa

  Esan bezala, populazioak eboluzionatu ahal izateko soluzioak desberdinak izatea berebizikoa da. Hori dela eta, behin gurutzaketa-operadorearen bidez soluzio berriak lortzen direnean, zorizko aldaketak eragin ohi dira mutazio-operadorearen bidez.

  Mutazio-operadorea bi parametroren bidez definitzen da. Batetik, mutazioaren magnitudea. Honek mutazioa aplikatzen denean aldaketa zenbatekoa izango den zehazten du. Bestetik, mutazio-operadorea era probabilistikoan aplikatzen da, ez zaie soluzio guztiei aplikatzen. Hortaz, mutazio-operadoreari lotutako bigarren parametroa mutazio-probabilitatea izango da.

  2.5 irudian ikus daiteke algoritmo genetikoaren sasikodea.



2.5 Irudia: Algoritmo Genetikoaren Sasikodea

Aplikaziorako Softwarea

Azpiatal honetan irudiak transformatzeko, OCR egiteko eta hiztegien bidez testu digitalak zuzentzeko beharrezkoa den sowftwarea agertuko da.

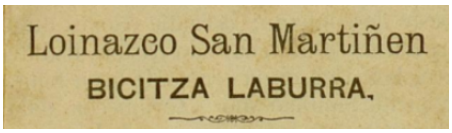
Image Magic

ImageMagick, hainbat utilitate dituen software librea da. Irudiak erakutsi, manipulatu eta transformatzeko gaitasuna duena. Tresna hau Apacheren lizentziapean (ikus 2.3.1) banatzen da.
Programa honek ez du interfaze grafiko propiorik eta beraz komando lerroa erabili behar da irudiak eraldatzeko. Zenbait erabiltzailerentzat hori traba handia bada ere, proiektu honetarako oso egokia da, horrela atazak erraz automatizatu baitaitezke.[Image Magick, 2017]

Aplikazio honen bidez lortu dira 2.1.1 atalean azaldutako eraldaketa batzuk. Bakoitzak zer egiten duen hobeto ikusteko irudi zati bat erabiliko da transformazioen adibide moduan, 2.6, 2.7, 2.8, 2.9 eta 2.10 irudietan ikusi dezakegun bezala[4]. Komando lerroko *convert* komandoaren bidez burutzen dira eraldaketa horiek.

[4]Originala ez da transformazio bat, hauek argazki originalarekiko duten aldea ikusteko jarri da

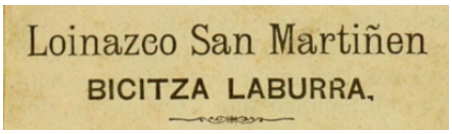
- **Originala**



2.6 Irudia: Originala

- **Contrast:** Ondorengo komandoaren bidez irudi originalaren kontrastea handitzea lortzen da. Ikus 2.1.1

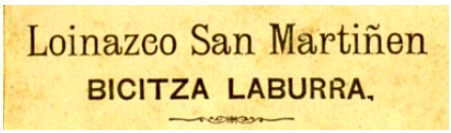
```
convert -contrast original.png contrast.png
```



2.7 Irudia: Contrast

- **Normalize:** Kasu honetan ere irudiari kontrastea igoko zaio, baina, intentsitate balioak luzatuz koloreen balio guztiak eskuratzeko. Ikus 2.1.1

```
convert -normalize original.png normalize.png
```



2.8 Irudia: Normalize

- **Negate:** 2.1.1 atalean azaltzen dena egiten du ondorengo komandoak.

```
convert -negate original.png negate.png
```

2.9 Irudia: Negate

- **Edge** 2.1.1 ataleko Canny azpiatalean azaldukoa burutzen da. Komando honek *radius* parametroa du, honen arabeza ertzaren detekzioa aldatuz.

```
convert -edge 2 original.png edge2.png
convert -edge 4 original.png edge4.png
```

(a) Edge 2

(b) Edge 4

2.10 Irudia: Edge

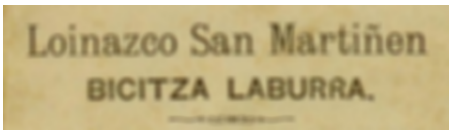
- **Lat:** 2.1.1 atalean azaldutakoa burutzen da. Bertan aipatu bezala, irudia zatika prozesatzen da, ondorioz, komandoari zatiak zenbatekoak izango diren esan behar zaio, 3x3 pixelekoak kasu honetan.

```
convert -lat 3x3 original.png lat.png
```

2.11 Irudia: Lat

- **Gaussian-Blur:** 2.1.1 atalean azaldutakoa burutzen da. Bertan aipatu bezala, irudia zatika prozesatzen da, ondorioz, komandoari zatiak zenbatekoak izango diren esan behar zaio, 3x3 pixelekoak kasu honetan.

```
convert -gaussian-blur 3x3 original.png gaussian-blur.png
```



2.12 Irudia: Gaussian-Blur

Index Values

Index Values irudi bateko ertzen detekziorako programa bat da. 2.1.1 atalean azaltzen da ertzak detektatzeko programa honen funtzionamendua. Gehiago sakondu nahi izanez gero ikus [Mugarza, 2016]

Hauek izan dira probetan erabili diren transformazioak, 2.13 eta 2.14 irudietan ikusi daitezke bakoitzaren adibideak. Irudiak eraldatzeko *demo* programa exekutagarria beharrezkoa da.

- **Moran:** Transformazio honek *radius* parametroa du, honen arabera ertzaren detekzioa aldatuz. Ondorengo irudiaetan ikus dezakegu hala ere ez dagoela alde handirik.

```
./demo origi.png 1 M
./demo origi.png 3 M
```

(a) Moran 1

(b) Moran 3

2.13 Irudia: Moran

- **Edu:** Transformazio honek *radius* parametroa du, honen arabera ertzaren detekzioa aldatuz. Ondorengo irudiaetan ikus dezakegu hala ere ez dagoela alde handirik.

```
./demo origi.png 1 E1
./demo origi.png 3 E1
```

(a) Edu 1

(b) Edu 3

2.14 Irudia: Edu

OpenCV

OpenCV ikusmen artifizialeko liburutegi librea da, hasiera batean Intel [5] -ek garatua. Bere lehen bertsiotik (1999) hainbat aplikaziotan erabili izan da teknologia hau, besteak beste, mugimendua detektatzeko dituzten segurtasun-sistemetan. Hain arrakasta handia izatearen eragilea bere BSD lizentzia (ikus 2.3.1) izan da. [OpenCV, 2017]

Index Values-ek liburutegi hau erabiltzeren arrazoi nagusienetakoa argazkiekin lan egiteko ematen dituen erraztasunak dira, dituen *Mat* egiturak direla eta. Egitura hauek argazkiak matrize moduan gordetzen dituzte, balioak eta balioen helbidea gordez. Balioen helbide hau erabiliz kalkuluen abiadura asko azkartu daiteke. Gainera, *Mat* egiturek errenkadaka edota zutabeka erraztasun handiz lan egitea ahalbidetzen dute.

Tesseract

Tesseract OCR egiteko aplikazio librea da. Bere sortzailea Hewlett Packard izen zen 1985ean. Hainbat urtez software pribatiboa izan arren, inolako garapenik ez zuela ikusita, 2005ean bere kodea eskuragarri utzi zuen. Gaur egun, aplikazio honen garapen nagusia Google-k burutzen du eta apache 2.0 lizentziapean (ikus 2.3.1) banatzen da.

Hainbat hizkuntzetarako dago garatua, besteak beste, euskararako. Proiektu honetan, euskara, gaztelera eta inglesa erabiliko dira. [Tesseract, 2017]

Ondorengo adibidean, aurrez ikusita 2.6 irudiari OCR nola egin eta zer nolako emaitza lortzen den ikusiko da.

- Komandoa:

```
tesseract -l eus origi.png originalaOCR
```

[5] Estatu Batuetako multinazional eta enpresa teknologikoa

- originalaOCR.txt fitxategiaren edukia:

```
Loinazco San Martiñen
! BlCITZA LABURRA.
```

Hiztegiak

Nahiz eta irudiak aurreprozesatu, oso zaila da *Tesseract* aplikazioak testu guztia zuzen digitalizatzea. Ondorioz, beharrezkoa da, jadanik digitalizatuta dagoen testua prozesatzea, hiztegi digitalen bidez hau zuzenduz. Tesseract aplikazioak ere egiten du prozesu hau, baina atal honetan lortu nahi dena, beraren teknika ordezkatzea edo berak egiten dituen akatsak konpontzea da.

MySpell

MySpell, ortografia zuzentzaile bat da. BSD lizentziapean (ikus 2.3.1) banatzen dena. Hainbat hizkuntzetarako dago eskuragarri, baita euskararako ere.

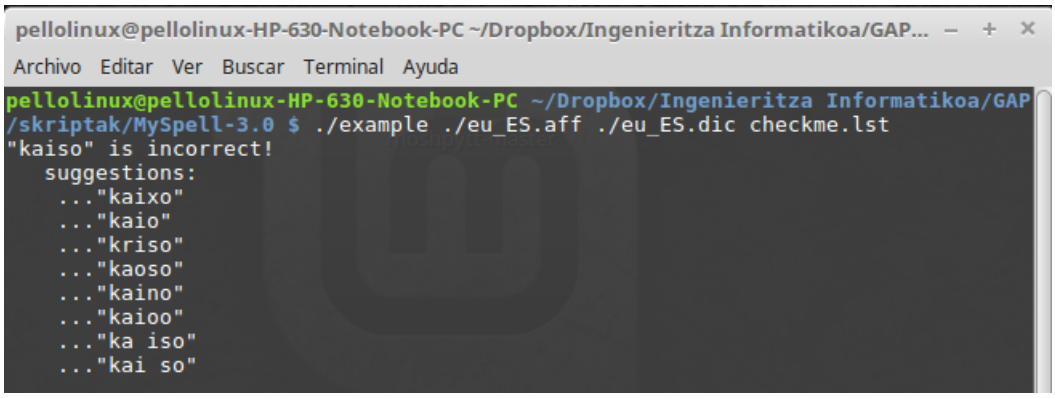
Ortografia zuzentzaile honi esker, hitz bat zuzena den edo ez erabakiko da lehenik. Hiztegiak zuzena dela badio, hau zuzentzat jo eta hurrengo hitzera salto egingo da. Hitza okerra bada ordea, antzeko hitzak itzultzen ditu hiztegiak, eta beraz baliabide honetaz baliatu daiteke hitza zuzentzeko.[MySpell, 2016]

Ondorengo adibidean 2.15 ikusi daiteke, euskarazko hiztegia erabiliz, algoritmoak 'kaixo' hitza interpretatu beharrean 'kaiso' hitza interpretatuko balu, hitza ez dela zuzena itzultzen duela. Gainera, egiten dituen iradokizunen artean 'kaixo' agertzen da. Kasu honetan hitza egoki zuzentzea lortuko litzateke.

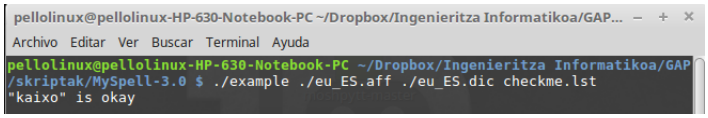
Bigarren adibide honetan 2.16 ikusi dezakegu berriz, 'agur' hitza interpretatzen badu algoritmoak, hitza zuzentzat joko duela.

Hunspell

Hunspell ortografia egiaztatzailea eta morfologia aztertzailea da, egitura morfologiko aberatsak eta karaktere sorta konplexuak erabiltzen dituzten hizkuntzentzako bereziki garatutakoa. Hasiera batean hungarierarentzako garatu zen software hau. Software librea da, GPL / LGPL / MPL lizentzia hirukoitzarekin (ikus 2.3.1) argitaratzen delarik.



2.15 Irudia: MySpell adibidea



2.16 Irudia: MySpell adibidea 2

Hunspell MySpell softwarean oinarrituta dago, eta azken honek erabiltzen dituen hiztegiekin bateragarria da. *MySpell*-ek 8 biteko ASCII karaktere sortak erabiltzen ditu, aldiz, *Hunspell*-ek, UTF-8 Unicode bezala kodetutako hiztegiak erabiltzeko gai da. [Hunspell, 2015]

Hasiera batean *MySpell* erabiltzeko asmoa bazegoen ere, *Hunspell*-ek UTF-8 Unicode bezala kodetutako hiztegiak erabiltzen dituenez, honen aldeko hautaketa egin da.

Gainera, aplikazioa *python*-en garatuko da eta *Hunspell*-ek *python*-erako liburutegia du, horrek gauzak asko erraztuz.

Oinarrizko Softwarea, Lizentziak eta Instalazioak

Proiektu honetan garatuko den aplikazioa Linux motako sistema eragiletarako garatu da. Aplikazio nagusiak behar dituen instalazioak ordea, Debian [6] sistema eragilearen eratorrietan zentratu da, horrenbestez, Debian, Ubuntu [7] eta Linux Mint [8]-erako balioko du besteak beste.

[6] Bere osotasunean software librez osatutako GNU/Linux banaketa.
[7] Idazmahairako GNU/Linux sistema eragile oso bat da, Debian banaketan oinarritutakoa
[8] Debian eta Ubuntun oinarritutako GNU/Linux banaketa da

Aplikazioa nagusia exekutatzeko python3 instalatuta izan behar da. Debian-en oinarritutako sistema eragileek hau instalatuta ekarri ohi badute ere, nahikoa da komando lerroan *apt-get install python3* exekutatzea, hau prest izateko. Bestetik, aplikazioan zehar, *python* programazio lengoaiaz prozesu batzuk hiltzeko *psutil* liburutegia erabiltzen da. Hau instalatzeko, *sudo apt-get install python3-psutil* eta *pip install psutil* exekutatu behar dira linux-eko komando lerroan.

Lizentziak

Apache lizentzia

Apache 2.0 lizentzia, Apache Software Foundation(ASF)-k sortutako software libreko lizentzia da. Libreak diren beste lizentzi guztiak bezala, erabiltzaileak edozein helburutako erabili, banatu, aldatu eta software horren bertsio berriak atera ditzake. Hala ere, copyleft lizentziarekin alderatuta, lizentzia hau duten lan eratorriak ez dira zertan apache 2.0 lizentziarekin argitaratu behar, ez eta software libre edo kode ireki moduan ere.[Apache, 2017]

BSD lizentzia

BSD lizentziak nagusiki BSD (Berkeley Software Distribution) sistemetan erabiltzen den software libre lizentzia-multzoa da. Izena lizentzia ematen duen UNIX-moduko sistema eragilearen izenetik datorkio.

Berau, Berkeleyko Californiako Unibertsitatean idatzi zen lehenengoz eta zenbait berrikuspen izan ditu ordutik hona, lizentzia eratorri horiek BSD lizentzia aldatua izena dute.

Lizentzia hauek, murriztapen gutxiago dituzte beste software libre lizentzia batzuekin alderatuz gero (esaterako, GPL). Hau da, BSD lizentzia bat esleiturik duten lanak, domeinu publikotik hurbilago daude, beste lizentzia batzuk esleituta dituzten lanak baino.[BSD, 2017]

GPL lizentzia

GNU General Public License (GNU, GPL edo GPL) (GNU Lizentzia Publiko Orokorra) *Free Software Foundation*-ek 1980ko hamarkada erdialdean sortutako lizentzia bat da, eta softwarearen erabilpen, aldaketa eta banaketa arloetara zuzenduta dago. Bere helburua, babesten duen softwarea software librea dela adieraztea da.

Badaude GPLren kide diren zenbait lizentzia ere:

- GFDL
- MGPL
- LGPL

Bere helburua, batez ere, banaketa askea, aldatzeko aukera eta software erabilera bultzatzea edo babestea da. Honen xedea lizentzia honengatik babestua dagoen software-a, "softare-libre"dela aitortzea da eta arau hauek ekiditea nahi dituzten erabiltzaileengatik babestea.[GPL, 2016]

Instalazioak

ImageMagic

GNU/Linux sistema gehienek ImageMagic defektuz instalatuta izaten dute. Ez bada ala, nahikoa da komando lerroan *apt-get install imagemagick* exekutatzea.

OpenCV

IndexValues, OpenCV erabiltzen duenez, hau instalatuta izan behar da. Instalatzeko `https://github.com/opencv/opencv` estekatik iturburu kodea jaitsi eta hau konpilatu behar da. Iturburu-kodearekin batera instalazioaren xehetasunak ere badakartza.

Tesseract

Hau instalatzeko, modurik errazena *apt* [9] komandoaz baliatzea da. Nahikoa da komando lerroan, *apt-get install tesseract-ocr* exekutatzea. Aplikazioak hiru hizkuntza [10] erabiltzen dituenez, hauek ere instalatu beharra daude. Adibidez euskara instalatzeko nahikoa da *apt-get install tesseract-ocr-eus* exekutatzea.

[9] Advanced Packaging Tool edo APT Debian proiektuak paketeak kudeatzeko garatutako tresna da. APTk GNU/Linux sistemetan programak instalatu eta desinstalatzeko lana errazten du.

[10] Euskara, Ingelesa eta Gaztelania

Tresna hau instalatzeko beste modua, iturburu kodea konpilatzea da. Horretarako https://github.com/tesseract-ocr estekatik kodea jaitsi eta hau konpilatu behar da. Aukera hau zailagoa da, askotan arazoak ematen baititu, paketeen dependentziekin besteak beste.

Hunspell

Hunspell-en liburutegia instalatu beharra dago, hau *python3*-n erabiltzeko. Instalazioa burutzeko, lehenik eta behin *python3* instalazioak erraz burutzeko paketea instalatuko dugu, *Linux*-eko komando lerroan honakoa exekutatuz: *sudo apt-get install python3-hunspell*. Behin hori eginda, berriro ere komando lerroan *pip3 install hunspell* exekutatuz, hiztegia *python*-etik eskuragarri izango genuke. Gure kasuan, hiru hizkuntzako hiztegiak erabili nahi ditugunez, azkeneko instalazio batzuk burutu behar dira. Nahiko da horretarako hiru komando hauek exekutatzea: *sudo apt-get install hunspell-en-gb*, *sudo apt-get install hunspell-es-es* eta *sudo apt-get install hunspell-eu-es*.

Azkenekoa ez litzateke instalatu beharrik egongo, *Xuxen*[11]-etik jaitsi baitira *Hunspell*-ek euskararako behar dituen fitxategiak. Hau egitearen arrazoia, *Xuxen*-eko hiztegiak, defektuzkoak baino emaitza hobea emango duela suposatu delako egin da. Ondorioz, programa nagusiarekin batera atxikituta egongo dira euskararako beharrezkoak diren bi fitxategiak.

[11] Euskaraz idatzitako testuak zuzentzen dituen aplikazio informatikoa

3. KAPITULUA

Diseinua

Atal honetan, esperimentazioaren eta garapenaren diseinua azalduko dira.

Esperimentuari dagokionez, irudiei OCR egitean asmatze-tasak kalkulatzeko ebaluazio irizpidea definituko da, baita, esperimentuan zehar burutuko diren proba desberdinak ere.

Garapenari dagokionez berriz, aplikazio nagusiaren nondik norakoak eta eskema orokorra ulertaraziko dira.

Esperimentazioaren Diseinua

Esperimentazioa burutzeko, lehenik eta behin, hizkuntza eta itxura desberdinetako irudi batzuk aukeratuko dira.

Ondoren, bi zati desberdinetan bana dezakegu esperimentazioa: irudien aurreprozesatzea eta hiztegiak. Lehendabizi irudien aurreprozesatzearen esperimentazioa burutuko da. Hiztegien esperimentuan ez gara hainbeste zentratuko eta horrenbestez aurreneko esperimentuak lortutako emaitzarik onenekin soilik egingo dira probak.

Asmatze-tasak kalkulatu eta soluziorik onenak zein diren identifikatzeko, ebaluazio-irizpide batzuk ezarri behar dira ordea.

Ebaluazio-Irizpidea

Esan bezala, testu bati OCR egin eta testu digitala lortzean, emaitza horren asmatze-tasa kalkulatu behar da. Balio hori kalkulatzeko bi testu behar dira: Testu-irudiari OCR egitean lortutako testu digitala (T1 deituko diogu) eta testu-irudian dagoen testua, erabat zuzen digitalizatuta (T2 deituko diogu).

Asmatze-tasa kalkulatzeko honako balioak eta formula erabili dira: T1-en zuzen idatzitako hitz kopurua (zuzen) eta T2-ren hitz kopurua (kopuruOsoa).

$$asmatzeTasa = \frac{100 \times zuzen}{kopuruOsoa}$$

Formula honen bidez asmatze-tasa kalkulatuko da eta ondorioz emaitzak konparatu ahal izango dira.

Irudien Aurreprozesatzea

Atal honetan irudien aurreprozesatze egoki bat lortzen saiatuko da.

Hasteko, 2.2.1 eta 2.2.2 ataletan ikusitako transformazio guztiak ebaluatuko dira. Gainera, irudi bakoitzaren pixel kopuru desberdinarekin egingo dira probak: Pixel kopuru originalarekin, 2 aldiz pixel kopuru gutxiagorekin eta 5 aldiz pixel kopuru gutxiagorekin.

Proba horietan lortzen diren 5 transformaziorik hoberenak pasatuko dira esperimentazioaren bigarren fasera. Bestetik, irudiari pixelak murrizteak zer nolako eragina duen ikusiko da.

Esperimentazioaren bigarren fasean, aurrez aukeraturiko bost transformaziorik hoberenak beraien artean konbinatuko dira, $5^2 = 25$ trasformazio desberdin lortuz. Hemen ere pixel kopuru originalarekin eta murriztuekin.

Bukatzeko, transformazioen biko konbinazioak lortu beharrean konbinazio luzeagoak lortzen saiatuko da, zortzi transformazioko sekuentziak alegia. Aurrez hautaturiko 5 transformazioez gain, *null* [1] sartzea ere interesgarria denez, 6 transformazio izango lirateke. Horrenbestez, $6^8 = 1.679.616$ konbinazio posible. Konputagailuak konbinaketa guzti horien asmatze-tasak kalkulatzeko behar duen denbora oso handia da ordea.

Arazo hori ebazteko, bilaketa heuristiko bat egitea pentsatu da, algoritmo genetiko bat

[1] Transformazio hau nulua izango da, ez dio irudiari inongo eraldaketarik eragingo

aplikatzea alegia. Horrela konbinazio guzti horietatik hoberena bilatzea ziurtatzen ez bada ere, soluzio on bat aurkituko da.

Bukatzeko, irudi guztiak desberdinak direnez, litekeena da irudi batentzat egokia den aurreprozesatzea beste irudiarentzat egokia ez izatea. Horrenbestez, irudi guztientzat nahiko ona den aurreprozesatze komun bat aurkitzen saiatuko da.

Hiztegiak

Atal honetan, bilaketa heuristikoetan lortutako emaitzarik onenak, testua hiztegien bidez zuzentzean lortutako emaitzak esperimentatuko dira.

Bi modu desberdinetara burutuko dira zuzenketak, automatikoki eta semiautomatikoki. Ondoren, taula bat osatuko da, zuzendu gabeko testua, zuzendutako bi testuekin konparatuz.

Garapenaren Diseinua

Aplikazioaren eskema orokorra hiru zati nagusik osatzen dute: Irudien aurreprozesatzea, irudiei OCR egitea eta OCR egitean lortutako testua hiztegien bidez zuzentzea.

Irudien aurreprozesatzearen kasuan, erabiltzaileak aplikazioari pasatzen dizkion argumentu kopuruaren arabera gauza desberdinak exekutatuko dira.

Argumentu kopurua bi[2] bada, defektuzko aurreprozesatzea burutuko da, hau da, esperimentazioan lortutako aurreprozesatze komuna aplikatuko zaie (ikus 4.3). Kasu honetan lehenengo argumentua OCR egitea nahi den irudien *path*[3]-a izango da. Bigarren berriz, zein hizkuntzatan idatzita dagoen irudiko testua.

Argumentu kopurua lau bada ordea programak beste bi argumentu jasoko ditu. Hirugarrena, OCR egitea nahi den irudien zati bat izango da eta laugarrena berriz azken zati horren testu digital zuzena. Bi argumentu hauen laguntzaz irudi konkretu horrentzako aurreprozesatzea egoki bat bilatzen saiatuko da sistema, algoritmo genetiko bat exekutatuz.

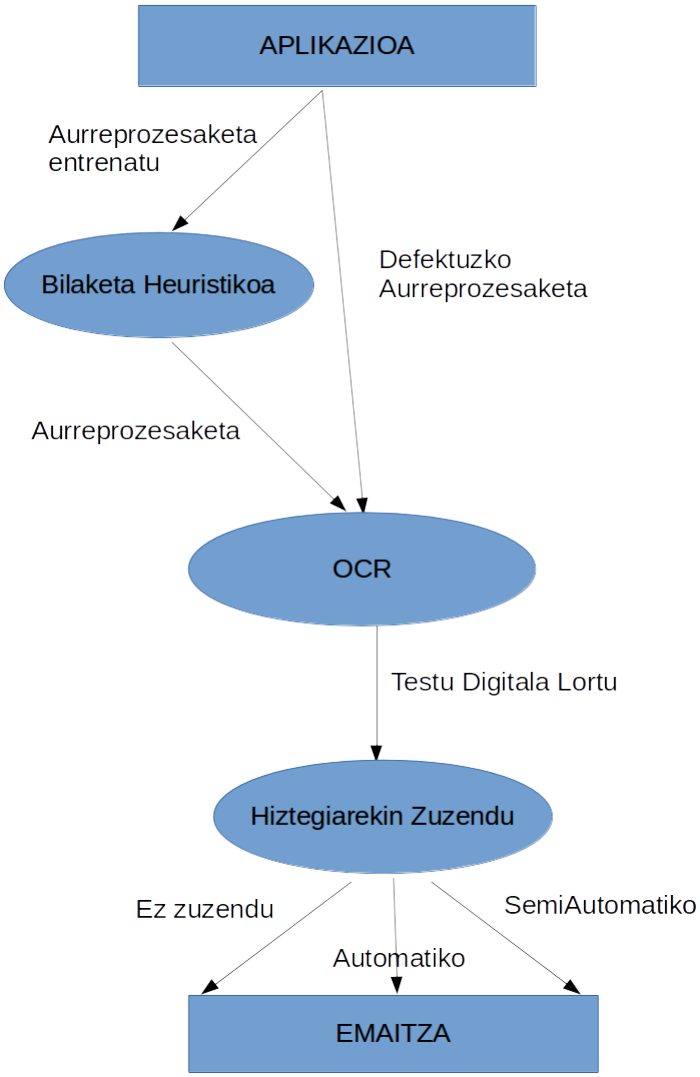
Modu batera edo bestera, behin irudiak aurreprozesatuta irudiei OCR egingo zaie, *tesseract* aplikazioaren bidez.

[2] python-en bat gehiago, 0-garrena programaren izena izaten baita
[3] fitxategiak ordenagailuan duen kokapena

Jada irudietako testua digitalizatuta dugula, honi azkeneko zuzenketak egitea faltako litzateke. Jakina baita OCR egitean testua ez dela guztiz zuzena geratzen. Horretarako *Hunspell* hiztegia erabiliko da eta erabiltzaileak hiru aukeren artean aukeratu beharko du: Testua ez zuzentzea, testua automatikoki zuzentzea edo testua semiautomatikoki zuzentzea.

Erabiltzaileak erabakitzen duena exekutatu ondoren aplikazioa amaituko da.

3.1 irudian ikusi daiteke eskema orokorra.



3.1 Irudia: Diseinuaren Eskema Orokorra

4. KAPITULUA

Esperimentazioa

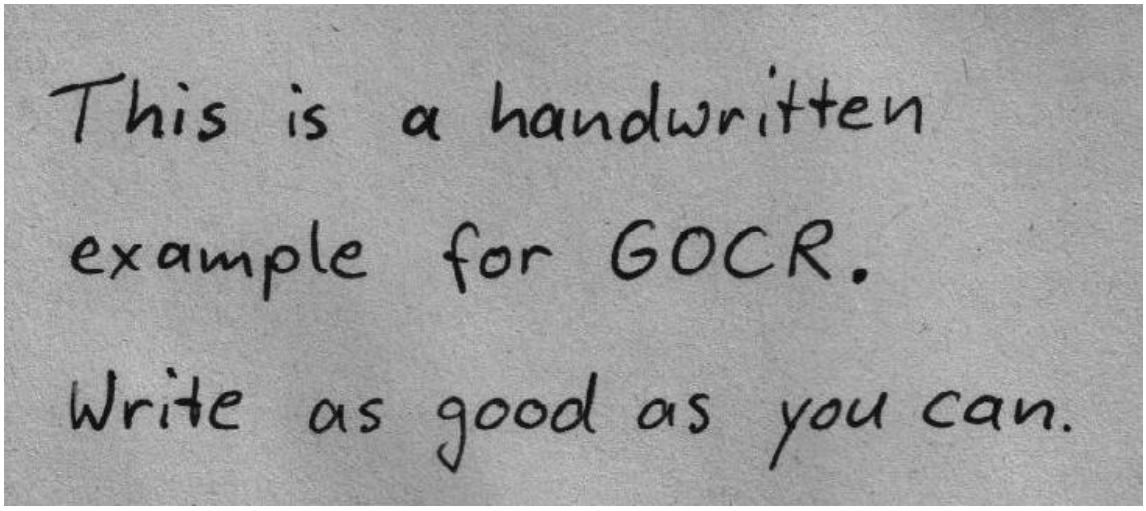
Atal hau, hiru zatitan banatu da. Hasteko *Testu-Irudien Aukeraketa* atalean esperimentua egiteko aukeratu diren irudiak eta hauek hautatzeko erabilitako irizpideak definitu dira. Ondoren, *Irudien Aurreprozesaketa* atalean irudiei transformazioak aplikatzean zer nolako emaitzak lortzen diren esperimentatutko da. Bukatzeko, *Hiztegien Eragina* atalean, aurreprozesatu ondoren lortutako testu digitalak zuzentzen saiatuko da, hiztegi digitalen laguntzaz.

Testu-Irudien Aukeraketa

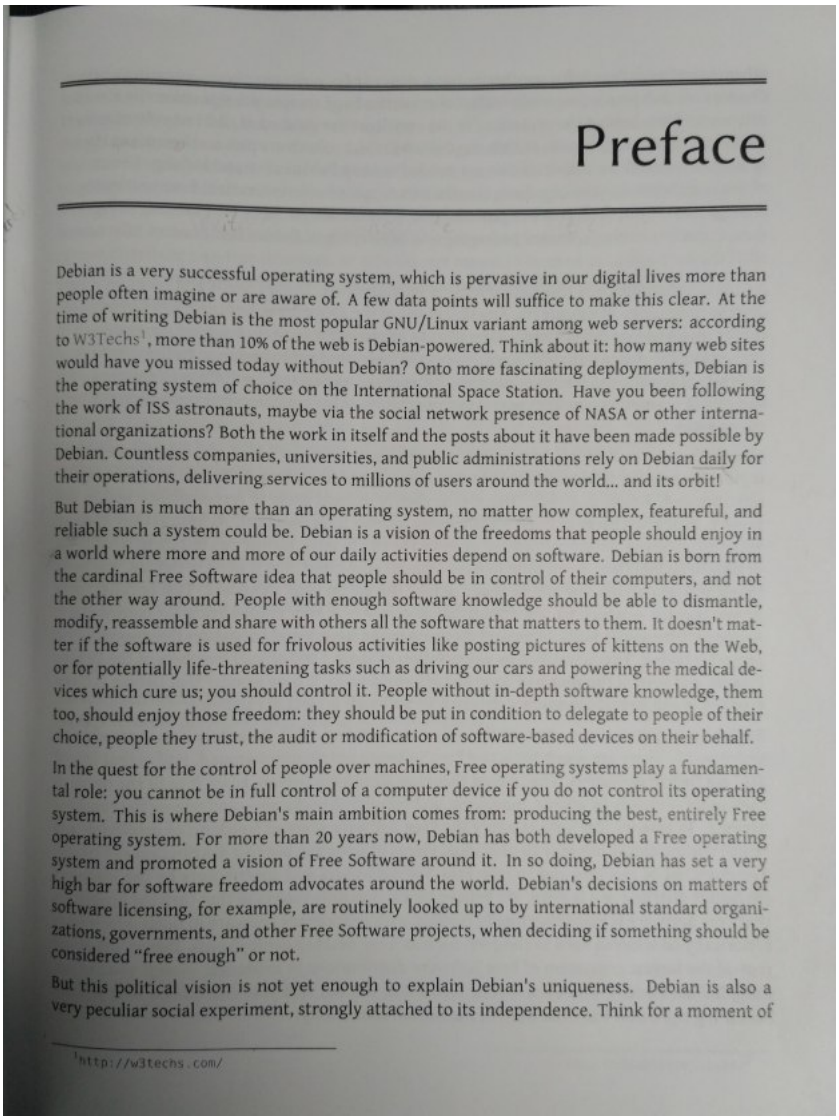
Esperimentazioaren diseinuan aipatu bezala, hizkuntza eta itxura desberdinetako testu-irudiak aukeratzen saiatu da. Hizkuntzari dagokionez, euskara, ingelesa eta gaztelania izan dira hautatuak izan direnak. Itxurari dagokionez berriz, letra oso garbia duten testuak zein hain garbiak ez dituztenak aukeratu dira.

Pixel kopuruak ere eragina izan dezakeenez, irudi berdina pixel kopuru desberdinetara eraldatu ondorengo esperimentazio bat ere burutuko da. Hain zuzen ere, irudiak defektuz dauzkan pixel kopuruarekin eta defektuzko pixel kopurua baino 2 aldiz gutxiagorekin eta 5 aldiz gutxiagorekin egingo dira probak. Jakina da, zenbat eta pixel gehiago izan, orduan eta denbora gehigo beharko duela konputagailuak irudia eraldatu eta OCR egitean. Beraz, denboran dagoen aldea eta asmatze tasetan dagoena konparatzea interesgarria izango da.

4.1, 4.2, 4.3 eta 4.4 irudiak izan dira esperimentazioa egiteko aukeratu direnak.



4.1 Irudia: Internetetik lortutako irudia; Hizkuntza ⇒ Ingelesa; 731x326 pixel



Preface

Debian is a very successful operating system, which is pervasive in our digital lives more than people often imagine or are aware of. A few data points will suffice to make this clear. At the time of writing Debian is the most popular GNU/Linux variant among web servers: according to W3Techs[1], more than 10% of the web is Debian-powered. Think about it: how many web sites would have you missed today without Debian? Onto more fascinating deployments, Debian is the operating system of choice on the International Space Station. Have you been following the work of ISS astronauts, maybe via the social network presence of NASA or other international organizations? Both the work in itself and the posts about it have been made possible by Debian. Countless companies, universities, and public administrations rely on Debian daily for their operations, delivering services to millions of users around the world... and its orbit!

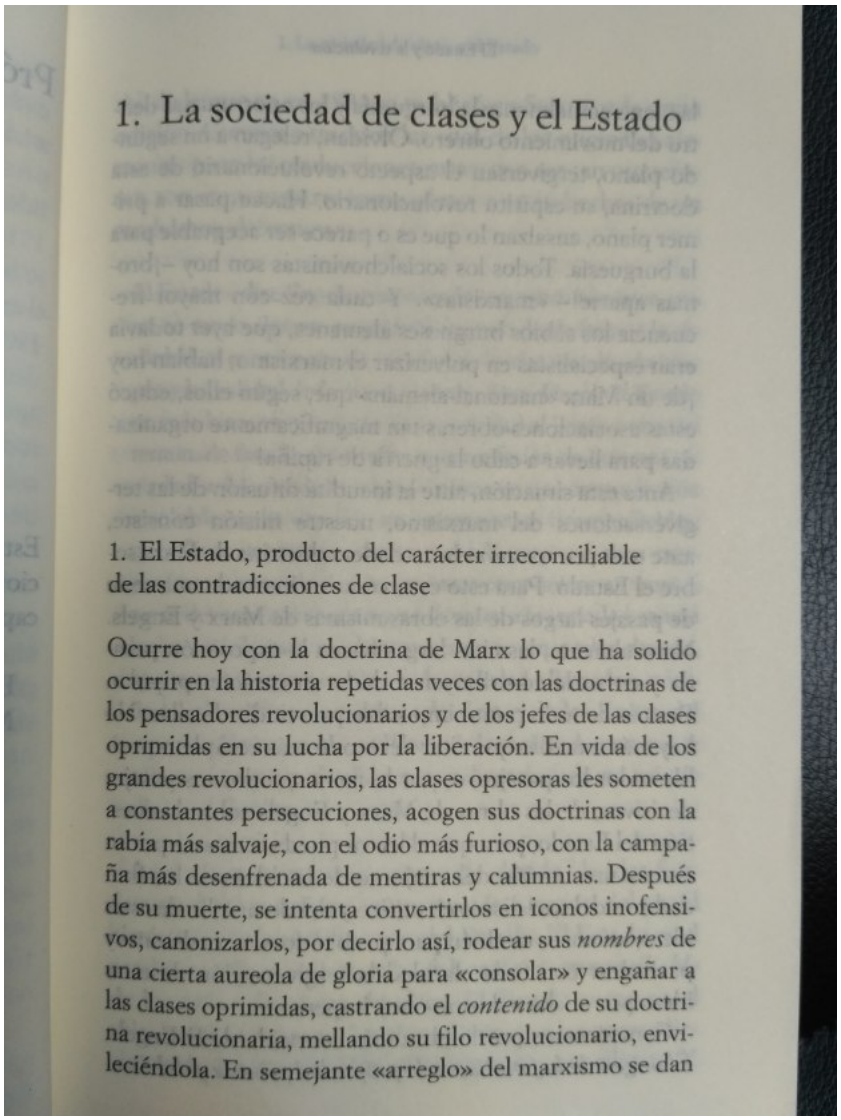
But Debian is much more than an operating system, no matter how complex, featureful, and reliable such a system could be. Debian is a vision of the freedoms that people should enjoy in a world where more and more of our daily activities depend on software. Debian is born from the cardinal Free Software idea that people should be in control of their computers, and not the other way around. People with enough software knowledge should be able to dismantle, modify, reassemble and share with others all the software that matters to them. It doesn't matter if the software is used for frivolous activities like posting pictures of kittens on the Web, or for potentially life-threatening tasks such as driving our cars and powering the medical devices which cure us; you should control it. People without in-depth software knowledge, them too, should enjoy those freedom: they should be put in condition to delegate to people of their choice, people they trust, the audit or modification of software-based devices on their behalf.

In the quest for the control of people over machines, Free operating systems play a fundamental role: you cannot be in full control of a computer device if you do not control its operating system. This is where Debian's main ambition comes from: producing the best, entirely Free operating system. For more than 20 years now, Debian has both developed a Free operating system and promoted a vision of Free Software around it. In so doing, Debian has set a very high bar for software freedom advocates around the world. Debian's decisions on matters of software licensing, for example, are routinely looked up to by international standard organizations, governments, and other Free Software projects, when deciding if something should be considered "free enough" or not.

But this political vision is not yet enough to explain Debian's uniqueness. Debian is also a very peculiar social experiment, strongly attached to its independence. Think for a moment of

[1] http://w3techs.com/

4.2 Irudia: *Debian Administration Handbook* liburuari mugikorrarekin argazkia aterata lortutako irudia; Hizkuntza ⇒ Ingelesa; 3120x4160 pixel

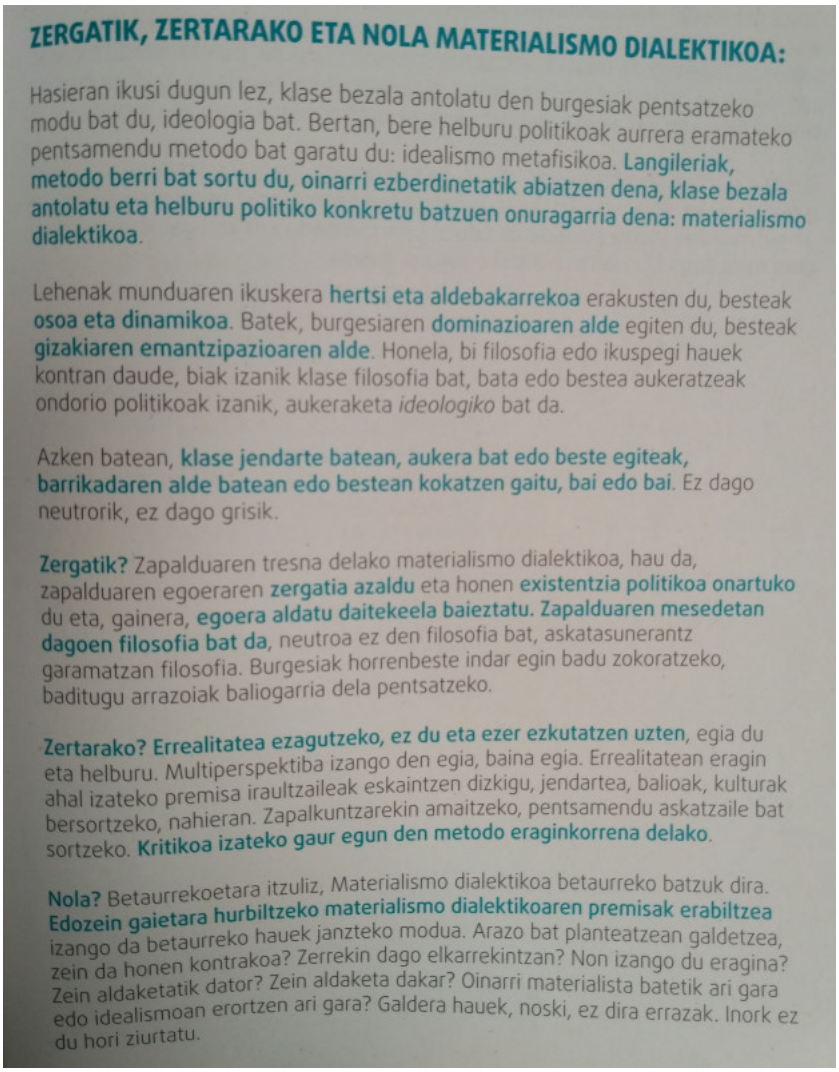


1. La sociedad de clases y el Estado

1. El Estado, producto del carácter irreconciliable de las contradicciones de clase

Ocurre hoy con la doctrina de Marx lo que ha solido ocurrir en la historia repetidas veces con las doctrinas de los pensadores revolucionarios y de los jefes de las clases oprimidas en su lucha por la liberación. En vida de los grandes revolucionarios, las clases opresoras les someten a constantes persecuciones, acogen sus doctrinas con la rabia más salvaje, con el odio más furioso, con la campaña más desenfrenada de mentiras y calumnias. Después de su muerte, se intenta convertirlos en iconos inofensivos, canonizarlos, por decirlo así, rodear sus *nombres* de una cierta aureola de gloria para «consolar» y engañar a las clases oprimidas, castrando el *contenido* de su doctrina revolucionaria, mellando su filo revolucionario, envileciéndola. En semejante «arreglo» del marxismo se dan

4.3 Irudia: *Estado y la Revolución* liburuari mugikorrarekin argazkia aterata lortutako irudia; Hizkuntza ⇒ Gaztelera; 3120x4160 pixel



ZERGATIK, ZERTARAKO ETA NOLA MATERIALISMO DIALEKTIKOA:

Hasieran ikusi dugun lez, klase bezala antolatu den burgesiak pentsatzeko modu bat du, ideologia bat. Bertan, bere helburu politikoak aurrera eramateko pentsamendu metodo bat garatu du: idealismo metafisikoa. **Langileriak, metodo berri bat sortu du, oinarri ezberdinetatik abiatzen dena, klase bezala antolatu eta helburu politiko konkretu batzuen onuragarria dena: materialismo dialektikoa.**

Lehenak munduaren ikuskera **hertsi eta aldebakarrekoa** erakusten du, besteak **osoa eta dinamikoa**. Batek, burgesiaren **dominazioaren alde** egiten du, besteak **gizakiaren emantzipazioaren alde**. Honela, bi filosofia edo ikuspegi hauek kontran daude, biak izanik klase filosofia bat, bata edo bestea aukeratzeak ondorio politikoak izanik, aukeraketa *ideologiko* bat da.

Azken batean, **klase jendarte batean, aukera bat edo beste egiteak, barrikadaren alde batean edo bestean kokatzen gaitu, bai edo bai**. Ez dago neutrorik, ez dago grisik.

Zergatik? Zapalduaren tresna delako materialismo dialektikoa, hau da, zapalduaren egoeraren **zergatia azaldu** eta honen **existentzia politikoa onartuko** du eta, gainera, **egoera aldatu daitekeela baieztatu. Zapalduaren mesedetan dagoen filosofia bat da**, neutroa ez den filosofia bat, askatasunerantz garamatzan filosofia. Burgesiak horrenbeste indar egin badu zokoratzeko, baditugu arrazoiak baliogarria dela pentsatzeko.

Zertarako? Errealitatea ezagutzeko, ez du eta ezer ezkutatzen uzten, egia du eta helburu. Multiperspektiba izango den egia, baina egia. Errealitatean eragin ahal izateko premisa iraultzaileak eskaintzen dizkigu, jendartea, balioak, kulturak bersortzeko, nahieran. Zapalkuntzarekin amaitzeko, pentsamendu askatzaile bat sortzeko. **Kritikoa izateko gaur egun den metodo eraginkorrena delako.**

Nola? Betaurrekoetara itzuliz, Materialismo dialektikoa betaurreko batzuk dira. **Edozein gaietara hurbiltzeko materialismo dialektikoaren premisak erabiltzea** izango da betaurreko hauek janzteko modua. Arazo bat planteatzean galdetzea, zein da honen kontrakoa? Zerrekin dago elkarrekintzan? Non izango du eragina? Zein aldaketatik dator? Zein aldaketa dakar? Oinarri materialista batetik ari gara edo idealismoan erortzen ari gara? Galdera hauek, noski, ez dira errazak. Inork ez du hori ziurtatu.

4.4 Irudia: *Pentsamendu Kritikoruntz* liburuari mugikorrarekin argazkia aterata lortutako irudia; Hizkuntza ⇒ Euskara; 3120x4160 pixel

Irudien Aurreprozesaketa

Trasnformazioen Hautaketa

Esan bezala, emaitzarik onenak lortzen dituzten transformazioak identifikatuko dira atal honetan. Esperimentazio hau burutzeko, irudia eraldatu, OCR egin eta ondoren testu digital zuzenarekin konparatu da, horrela asmatze tasa lortuz, 3.1.1 atalean definitu diren irizpideak jarraituz.

4.1 taulan ikusi ditzakegu emaitzak. Errenkada bakoitza irudiaren transformazio bat izanik eta zutabe bakoitza irudi desberdin bat.

Bestetik taula hiru errenkada nagusitan banatu da. Lehenengo bi zatietan, irudiaren pixel murriztuekin[1] egin dira probak eta azken zatian, pixel kopuru originalarekin. Jakina da, pixel murriztuak dituzten irudiak prozesatzeko denbora gutxiago beharko dela, baina asmatze tasak txikiagoak izango direla espero da. Denbora[2] eta asmatzeen arteko erlazioa aztertuko da horrela.

4.1 taulan ikus daitekeen bezala, ez dago transformazio bat lau irudietan asmatze-tasa altuena lortzen duena. Hala ere, garbi ikusi daiteke zein transformaziok ez duten batere laguntzen. Edge, Lat eta Edu eraldaketek asmatze-tasa nuluak lortzen dituzte eta horrenbestez hauek ez dira erabiliko. Lehen aipatu den bezala, bost transformazio esanguratsuenak aukeratu behar direnez, Contrast, Normalize, Negate, Gaussian-Blur eta Moran 3-ren aldeko hautua egin da. Taulari begiratuz, ez da inongo arazorik egon hauek aukeratzeko. Izatekotan, Moran 1 eta Moran 3-ren artean egon daiteke zalantza. Hala ere, zenbait kasutan Moran 3-k asmatze-tasa altuagoak lortu dituenez, hau aukeratzea pentsatu da.

Pixelak murrizteari dagokionez, ederki ikusi daiteke, pixelak gehiegi murriztean, denbora askoz gutxiago behar badu ere asmatze-tasetan ez duela batere laguntzen. Hala ere, pixelak erdira jeistean, asmatze-tasak handitzea eta dena lortu da, denbora gutxiago tardatuz gainera. Horrenbestez, hemendik aurrerako esperimentazioak pixel kopuru originalarekin zein, pixelak erdira murriztuta egingo dira.

[1] Lehen zatian, 5 aldiz pixel gutxiagorekin eta bigarrengoan 2 aldiz pixel gutxiagorekin
[2] Proba guztiak konputagailu berdinarekin egin dira, bere prozesadorea: Intel Core i5-4690 3.5Ghz

Irudiaren Transformazioa	1.Ir 4.1	2.Ir 4.2	3.Ir 4.3	4.Ir 4.4
Pixelak Murriztuta (5 aldiz gutxiago)				
Originala	**07,69**	28,60	55,33	**38,01**
Contrast	00,00	25,32	**62,00**	37,32
Normalize	**07,69**	**26,63**	54,60	35,61
Negate	**07,69**	32,31	56,66	36,30
Edge 2	00,00	00,00	00,66	00,34
Edge 4	00,00	01,74	10,00	04,45
Lat 3x3	00,00	03,27	00,00	00,00
Gaussian-blur 3x3	00,00	00,00	00,00	00,34
Moran 1	00,00	00,00	00,00	01,02
Moran 3	00,00	00,00	00,00	07,53
Edu 1	00,00	00,43	00,00	00,68
Edu 3	**07,69**	00,65	00,66	00,68
Denbora	2s	1m 52s	1m 11s	2m 33s
Pixelak Murriztuta (2 aldiz gutxiago)				
Originala	**69,23**	**83,18**	90,00	66,09
Contrast	**69,23**	82,75	84,00	70,20
Normalize	**69,23**	82,96	**91,31**	**72,26**
Negate	**69,23**	82,96	90,66	65,75
Edge 2	00,00	01,31	00,00	00,00
Edge 4	00,00	02,82	00,00	01,00
Lat 3x3	00,00	00,00	00,00	00,00
Gaussian-blur 3x3	07,29	46,00	64,00	22,26
Moran 1	00,00	27,29	04,66	14,38
Moran 3	00,00	19,86	06,66	67,46
Edu 1	00,00	00,00	00,00	00,00
Edu 3	00,00	00,00	01,33	00,00
Denbora	4s	2m 58s	1m 18s	2m 50s
Pixel Kopuru Originala				
Originala	61,53	82,96	**90,66**	56,16
Contrast	**76,92**	**83,62**	85,33	49,31
Normalize	69,23	81,87	88,66	53,76
Negate	61,53	82,75	86,00	56,84
Edge 2	00,00	1,52	01,33	00,00
Edge 4	00,00	00,00	00,00	00,00
Lat 3x3	00,00	00,00	00,00	00,00
Gaussian-blur 3x3	**76,92**	81,22	83,33	59,24
Moran 1	15,38	72,48	31,33	**66,09**
Moran 3	15,38	74,23	33,33	**66,09**
Edu 1	00,00	00,00	00,00	00,00
Edu 3	00,00	00,00	00,00	00,00
Denbora	7s	4m 09s	2m 46s	1m 28s

4.1 Taula: Transformazioen Asmatze Tasak

Transformazioen Konbinaketa Binaka

Esperimentazioko atal honetan, aurrez aipaturiko bost transformazioen biko konbinazioek ebaluatuko dira, $5^2 = 25$ konbinazio hain zuzen ere.

Esan bezala, ez dago aurreprozesatze bat irudi guztientzat egokia dena. Beraz, aurreprozesatze desberdinak konbinatzeak hobekuntzak ekartzea litekeena da.

4.2 taulan pixel originalak dituzten irudiekin egingo da esperimentua eta 4.3 taulan berriz, pixelak erdira murriztuta. Bi taulen errenkadak biko konbinazioak izango dira eta zutabeak berriz, lau irudiak.

Irudiaren Transformazioa	1.Ir 4.1	2.Ir 4.2	3.Ir 4.3	4.Ir 4.4
Pixel Kopuru Murriztuekin				
Originala	69,23	83,18	90,00	66,09
Contrast + Contrast	46,15	82,31	78,00	69,86
Contrast + Normalize	61,53	82,53	83,33	65,06
Contrast + Negate	69,23	82,75	84,00	66,09
Contrast + Gaussian-blur	00,00	50,43	64,00	36,30
Contrast + Moran 3	00,00	39,51	56,00	03,42
Normalize + Contrast	**76,92**	**83,62**	**92,66**	70,54
Normalize + Normalize	69,23	80,78	92,00	**72,60**
Normalize + negate	69,23	83,18	90,66	72,26
Normalize + Gaussian-blur	00,00	37,33	70,00	24,65
Normalize + Moran 3	07,69	45,85	92,66	35,95
Negate + Contrast	69,23	82.75	81,33	71,57
Negate + Normalize	69,23	82,75	90,00	**72,60**
Negate + Negate	69,23	83,18	90,00	66,09
Negate + Gaussian-blur	07,69	47,16	65,33	22,94
Negate + Moran 3	00,00	19,86	06,66	68,15
Gaussian-blur + Contrast	00,00	43,01	69,33	32,53
Gaussian-blur + Normalize	00,00	47,81	66,66	21,57
Gaussian-blur + Negate	07,69	47,16	63,33	26,71
Gaussian-blur + Gaussian-blur	00,00	05,02	34,66	05,13
Gaussian-blur + Moran 3	00,00	05,02	34,66	05,13
Moran 3 + Contrast	00,00	13,97	01,33	60,61
Moran 3 + Normalize	00,00	13,97	01,33	60,61
Moran 3 + Negate	00,00	19,86	06,66	67,46
Moran 3 + Gaussian-blur	00,00	02,83	00,00	21,91
Moran 3 + Moran 3	00,00	00,00	00,00	00,00
Denbora	10s	3m 38s	2m 01s	3m 34s

4.2 Taula: Transformazioen Konbinazioen Asmatze Tasak, pixel kopuru murriztuekin

Irudiaren Transformazioa	1.Ir 4.1	2.Ir 4.2	3.Ir 4.3	4.Ir 4.4
Pixel Kopuru Originala				
Originala	61,53	82,96	**90,66**	56,16
Contrast + Contrast	69,23	80,34	68,00	56,50
Contrast + Normalize	**76,92**	82,09	86,00	45,54
Contrast + Negate	**76,92**	**83,62**	84,66	52,39
Contrast + Gaussian-blur	**76,92**	80,34	68,00	61,78
Contrast + Moran 3	07,69	72,27	20,66	49,33
Normalize + Contrast	69,23	82,96	86,00	55.82
Normalize + Normalize	61,53	82,31	88,66	54.10
Normalize + negate	69,23	82,75	88,00	53,76
Normalize + Gaussian-blur	69,23	81,22	79,33	58,90
Normalize + Moran 3	00,00	40,39	40,00	11,98
Negate + Contrast	**76,92**	82,53	82,66	55,13
Negate + Normalize	61,53	81,00	86,00	53,08
Negate + Negate	61,53	82,32	86,00	55,82
Negate + Gaussian-blur	**76,92**	81,44	80,66	58,96
Negate + Moran 3	15,38	69,86	14,66	16,78
Gaussian-blur + Contrast	**76,92**	80,78	69,33	57,19
Gaussian-blur + Normalize	**76,92**	80,56	86,00	59,24
Gaussian-blur + Negate	**76,92**	81,44	84,66	59,58
Gaussian-blur + Gaussian-blur	38,46	78,82	68,66	49,65
Gaussian-blur + Moran 3	38,46	78,82	68,66	49,65
Moran 3 + Contrast	00,00	74,23	30,66	**78,08**
Moran 3 + Normalize	00,00	74,23	30,66	**78,08**
Moran 3 + Negate	15,38	75,32	33,33	66,09
Moran 3 + Gaussian-blur	53,84	76,20	51,33	63,69
Moran 3 + Moran 3	00,00	00,00	01,33	66,78
Denbora	12s	3m 37s	6m 55s	4m 14s

4.3 Taula: Transformazioen Konbinazioen Asmatze Tasak, pixel kopuru originalekin

4.2 eta 4.3 tauletan ikusi dezakegun bezala, transformazioen biko konbinazioak erabilita asmatze-tasak hobetzea lortu da zenbait kasutan. Hori horrela izanik, transformazioen arteko biko konbinazioak egin beharrean, zenbaki altuagoko konbinazioak egitea bururatu daiteke. Hori izango da hurrengo atalean landuko den ideia.

Gainera, pixelak erdira murriztearekin ere emaitza egokiak lortu dira. Asmatze-tasa eta denboraren arteko erlazioa hobetuz.

Aipatu bezala, hurrengo atalean kopuru altuagoko konbinazioak probatuko dira, pixel originalekin zein murriztuekin.

Transformazioen Konbinaketa Bilaketa Heuristikoen Bidez

Beraz, biko konbinazioen arteko eraldaketak egin beharrean zenbaki altuagoko konbinazioen eraldaketak egitea ideia ona dirudi.

Diseinuan aipaturiko arazoa ebazteko, bilaketa heuristiko bat egitea pentsatu da, algoritmo genetiko bat aplikatzea alegia. Horrela konbinazio guzti horietatik hoberena bilatzea ziurtatzen ez badugu ere, soluzio on bat aurkituko dugu.

Horrenbestez, irudi bakoitzarentzat taula bat osatuko da. Taula hauen errenkadetan, algoritmo genetikoa probatzeko parametro desberdinak (ikus 5.2) jarriko dira eta bi zutabeetan berriz, bata irudia pixel guztiekin eta bestea pixel murriztuekin [3]. Bestetik, algoritmo genetikoak lortu duen soluzioa, hots, transformazioen sekuentzia eta soluzio hau lortzeko pasatako denbora ere agertuko dira tauletan.

[3] Pixel murriztuak dituen irudiak, pixel kopuru erdia izango du

1.Irudia 4.1	Pixel Guztiak	Pixelak Murriztuta
Indibiduoak = 10 Ausaz Sortuta = 10 Generazioak = 3	76,92	76,92
Lortutako Soluzioa	[null, contrast, null, null, gaussian-blur, null, null, null]	[normalize, contrast, null, null, null, null, null, null]
Denbora	40s	34s
Indibiduoak = 10 Ausaz Sortuta = 10 Generazioak = 6	76,92	76,92
Lortutako Soluzioa	[null, contrast, null, null, null, null, null, null]	[normalize, contrast, null, null, null, null, null, null]
Denbora	59s	52s
Indibiduoak = 50 Ausaz Sortuta = 50 Generazioak = 3	**84,61**	76,92
Lortutako Soluzioa	[normalize, gaussian-blur, contrast, null, null, null, null, null]	[normalize, contrast, null, null, null, null, null, null]
Denbora	1m 57s	1m 38s
Indibiduoak = 50 Ausaz Sortuta = 50 Generazioak = 6	**84,61**	76,92
Lortutako Soluzioa	[normalize, gaussian-blur, null, negate, contrast, negate, negate, null]	[normalize, contrast, null, null, null, null, null, null]
Denbora	3m 02s	2m 53s
Indibiduoak = 100 Ausaz Sortuta = 100 Generazioak = 12	**84,61**	76,92
Lortutako Soluzioa	[normalize, contrast, null, null, null, null, null, gaussian-blur]	[normalize, contrast, null, null, null, null, null, null]
Denbora	10m 12s	9m 30s

4.4 Taula: 4.1 Irudian, Algoritmo Genetikoaren Asmatze Tasak

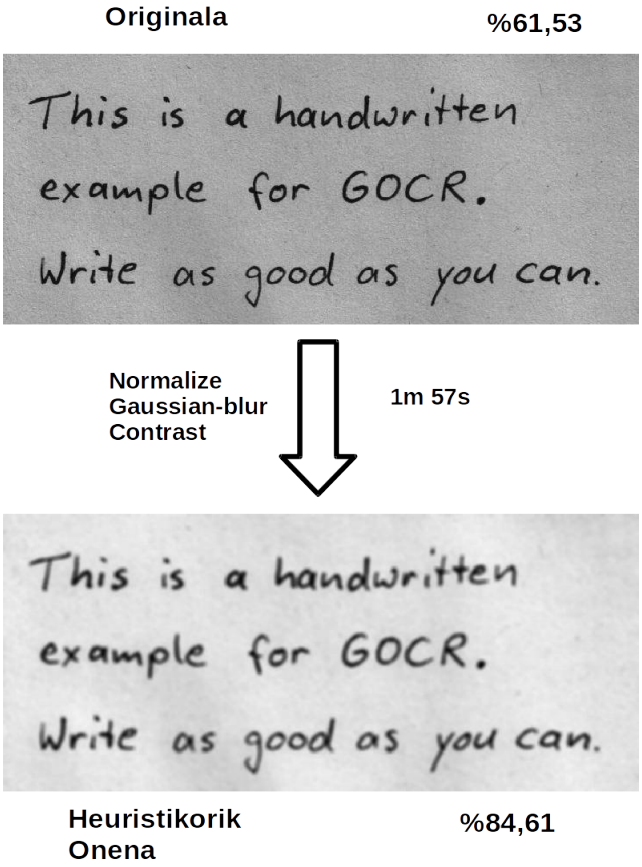
4.4 taulan ikus daiteke, 4.1 irudian heuristikoak lortutako emaitzak. Aipatzekoa da irudi honek pixel gutxiago dituela beste irudiek baino, baita testu kopuru askoz gutxiago ere. Bilaketa heuristikoak hain azkar burutzea horren ondorioa da.

Pixelak murriztean, bilaketa burutzeko denbora gutxiago behar badu ere, pixel originalekin asmatze-tasa altuagoak lortu dira.

Aipatzekoa da baita ere, pixel originaleko hiru errenkadetan lortu dela emaitzarik onena. Soluzioak berdinak ez badira ere, soluzio guztiek sortzen duten irudi berria antzekoa

izango dela suposatu daiteke, soluzio guztiak *Normalize* transformazioarekin hasten dira adibidez eta guztiek dute ondoren *Contrast* eta *Gaussian-Blur* eraldaketa.

Soluzio onena bezala, hirugarren errenkadako soluzioa hartu da. Asmatze-tasetan berdinketa zegoenez, denborari begiratu zaio. 4.5-en ikusi daitekeen bezala, heuristikoak lortutako soluzioak 61,53-ko asmatze-tasatik, 84,61-eko asmatze-tasara pasatzea lortu du.



4.5 Irudia: 4.1 irudiak heuristikoan lortutako aurreprozesaketarik onena

4.5 taulan ikus daitezke, 4.2 irudian heuristikoak lortutako emaitzak. 2.Irudi honen pixel kopurua handiagoa izateak bilaketa heuristikoak soluzioa bilatzen behar duen denboran ederki nabaritu da. Jakina den bezala, heuristikoari pasatako argumentuek ere erantzukizun zuzena dute bilaketaren denboran.

Pixel murriztuekin egindako bilaketek, emaitza txarrak lortu ez badituzte ere, pixel originalekin egindako bilaketen azpitik kokatu dira beti beraien asmatze-tasak.

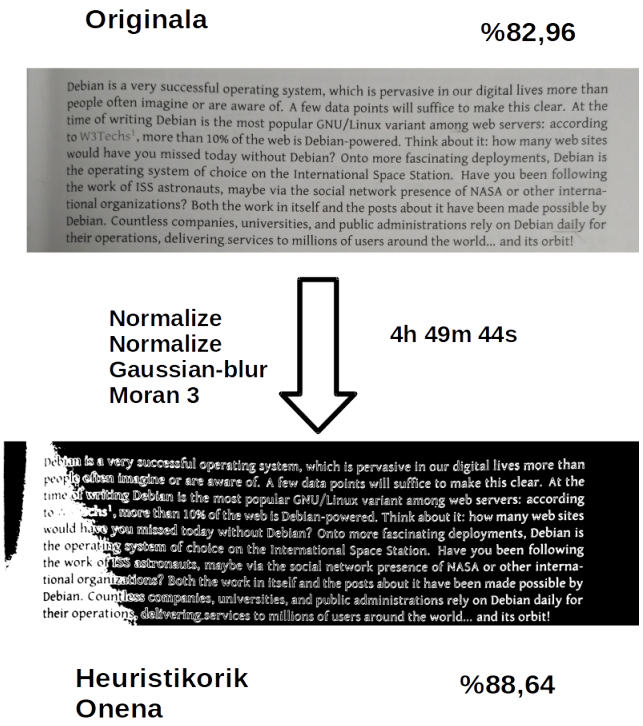
2.Irudia 4.2	Pixel Guztiak	Pixelak Murriztuta
Indibiduoak = 10 Ausaz Sortuta = 10 Generazioak = 3	85,15	83,62
Lortutako Soluzioa	[gaussian-blur, _3M, null, null, null, null, null, null]	[normalize, contrast, null, null, null, null, null, null]
Denbora	17m 43s	8m 47s
Indibiduoak = 10 Ausaz Sortuta = 10 Generazioak = 6	85,37	83,62
Lortutako Soluzioa	[normalize, gaussian-blur, null, _3M, null, null, null, null]	[normalize, contrast, null, null, null, null, null, null]
Denbora	20m 11s	10m 21s
Indibiduoak = 50 Ausaz Sortuta = 50 Generazioak = 3	86,46	84,71
Lortutako Soluzioa	[normalize, negate, null, gaussian-blur, normalize, _3M, null, null]	[negate, null, normalize, normalize, null, contrast, null, null]
Denbora	44m 17s	25m 26s
Indibiduoak = 50 Ausaz Sortuta = 50 Generazioak = 6	86,89	84,71
Lortutako Soluzioa	[negate, normalize, normalize, gaussian-blur, normalize, contrast, _3M, null]	[null, negate, normalize, normalize, null, null, _3M, null]
Denbora	1h 20m 11s	39m 40s
Indibiduoak = 100 Ausaz Sortuta = 100 Generazioak = 12	**88,64**	84,71
Lortutako Soluzioa	[normalize, normalize, null, null, null, null, gaussian-blur, _3M]	[normalize, negate, contrast, contrast, negate, null, normalize, null]
Denbora	4h 49m 44s	1h 53m 21s

4.5 Taula: 4.2 Irudian, Algoritmo Genetikoaren Asmatze Tasak

4.6 irudian, soluzio hoberen moduan kokatutako bilaketa ez da batere azkarra izan. Lau ordu baino gehiago behar izan baititu bilaketa burutzeko. Beraz, galdera bat bururatu daiteke momentu honetan, merezi al du hainbeste denbora itxarotea aurreprozesatze egoki bat lortzeko? Edo hobe da hainbeste ez itxaron eta asmatze-tasak txikiagoak badira ere, ondoren eskuz zuzentzea?

Kasu desberdinak egon daitezke. Adibidez, hamar orrialde digitalizatzea nahi bada ez zara lau ordu aurreprozesatze egoki bat bilatzen egongo. Mila orrialdeko 5 liburu digitalizatu nahi bada berriz, komenigarria izango litzateke bilaketa sakon bat egitea.

Esan bezala, asmatze-tasa altuena lortzen duena kalifikatu dugu soluzio hoberen moduan. Kasu honetan, 82,96-ko amatze-tasatik 88,64-ra igotzea lortu da.



4.6 Irudia: 4.2 irudiak heuristikoan lortutako aurreprozesaketarik onena

4.6 taulan ikus daitezke, 4.3 irudian heuristikoak lortutako emaitzak. Kasu honetan ere, heuristikoaren argumentuak zenbat eta handiagoak izan, bilaketa burutzeko denbora asko igotzen dela ikusi daiteke.

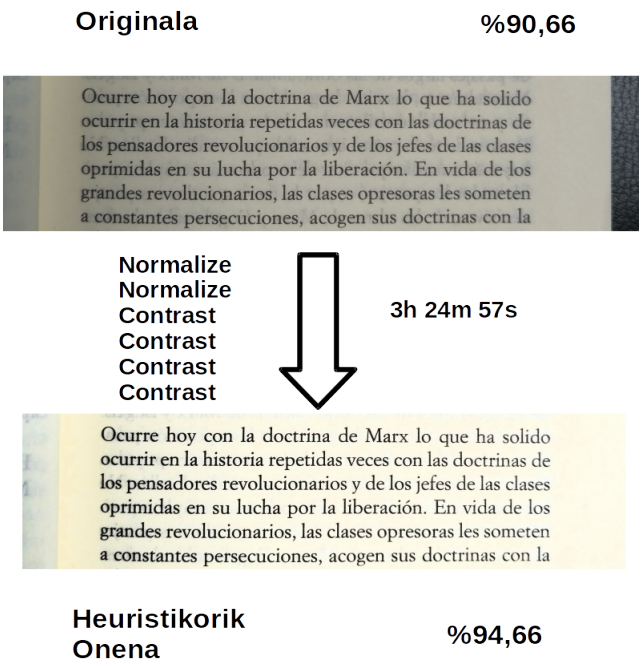
3.Irudia 4.3	Pixel Guztiak	Pixelak Murriztuta
Indibiduoak = 10 Ausaz Sortuta = 10 Generazioak = 3	90,66	92,66
Lortutako Soluzioa	[null, null, null, null, null, null, null, null]	[normalize, contrast, null, null, null, null, null, null]
Denbora	18m 43s	5m 52s
Indibiduoak = 10 Ausaz Sortuta = 10 Generazioak = 6	90,66	92,66
Lortutako Soluzioa	[null, null, null, null, null, null, null, null]	[normalize, contrast, null, null, null, null, null, null]
Denbora	21m 33s	7m 35s
Indibiduoak = 50 Ausaz Sortuta = 50 Generazioak = 3	92,66	92,66
Lortutako Soluzioa	[normalize, contrast, null, contrast, negate, contrast, negate, normalize]	[_3M, contrast, null, null, null, null, null, null]
Denbora	55m 08s	19m 48s
Indibiduoak = 50 Ausaz Sortuta = 50 Generazioak = 6	90,65	93,33
Lortutako Soluzioa	[null, null, null, null, null, null, null, null]	[normalize, normalize, normalize, null, null, contrast, contrast, contrast]
Denbora	1h 11m 28s	28m 29s
Indibiduoak = 100 Ausaz Sortuta = 100 Generazioak = 12	**94,66**	93,33
Lortutako Soluzioa	[normalize, null, null, normalize, contrast, contrast, contrast, contrast]	[normalize, null, contrast, normalize, _3M, negate, normalize, contrast]
Denbora	3h 24m 57s	1h 31m 31s

4.6 Taula: 4.3 Irudian, Algoritmo Genetikoaren Asmatze Tasak

Lehen bezala ordea, soluziorik hoberena aurkitu duen bilaketa, bilaketarik luzeena izan da. 4.7-n ikusi dezakegu soluzio horrek irudian duen eragina. Irudi originalak ere oso emaitza ona eman du 90,66-ko asmatze tasa alegia. Bilaketa heuristikoarekin 94,66-ra igotzea lortu da.

Bestetik, pixelak erdira murriztuta egin diren probetan, oso emaitza positiboak lortu dira. Zenbait kasutan, pixel kopuru originalarekin lortutako asmatze-tasak baino hobeak eta denbora gutxiagoan.

Kasu honetan ere aurreko irudiko galdera berbera egin dezakegu, ea merezi duen 3 ordu eta erdiko bilaketa bat egitea %4 hobetzeko. Esan bezala kasuak eta kasuak egon daitezke.



4.7 Irudia: 4.3 irudiak heuristikoan lortutako aurreprozesaketarik onena

4.7 taulan ikus daitezke, 4.4 irudian heuristikoak lortutako emaitzak. 4.irudi honetan gauzak aldatu dira. Aurreko irudi guztietan soluziorik hoberena pixel kopuru originalekin lortu da, kasu honetan ez.

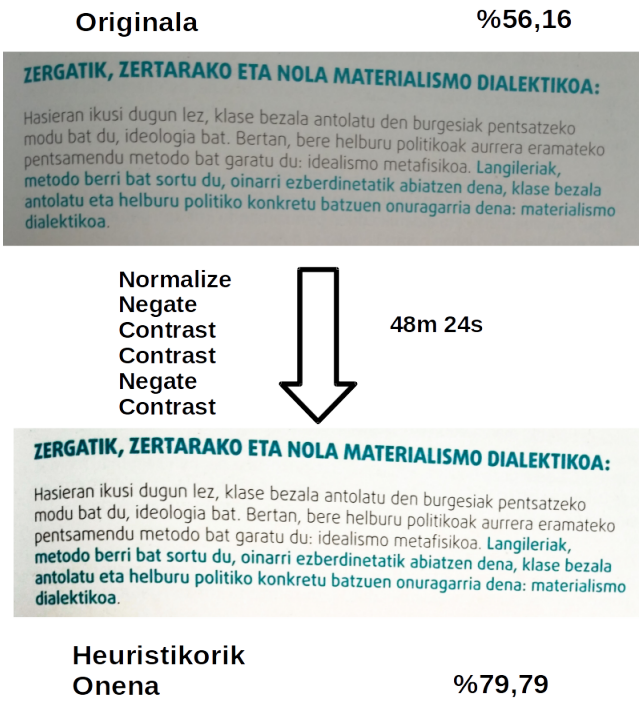
Emaitzarik hoberena, pixel kopuru murriztuarekin eta bilaketa 'azkar' batean lortu da,

4.Irudia 4.3	Pixel Guztiak	Pixelak Murriztuta
Indibiduoak = 10 Ausaz Sortuta = 10 Generazioak = 3	78,08	72,60
Lortutako Soluzioa	[_3M, contrast, null, null, null, null, null]	[negate, normalize, null, null, null, null, null]
Denbora	15m 59s	11m 27s
Indibiduoak = 10 Ausaz Sortuta = 10 Generazioak = 6	78,08	72,60
Lortutako Soluzioa	[_3M, contrast, null, gaussian-blur, null, null, null, null]	[negate, normalize, null, null, null, null, null, null]
Denbora	21m 33s	10m 54s
Indibiduoak = 50 Ausaz Sortuta = 50 Generazioak = 3	78,76	75,34
Lortutako Soluzioa	[null, _3M, null, null, negate, normalize, contrast, contrast]	[contrast, contrast, gaussian-blur, contrast, normalize, negate, _3M, contrast]
Denbora	55m 08s	33m 24s
Indibiduoak = 50 Ausaz Sortuta = 50 Generazioak = 6	78,76	**79,79**
Lortutako Soluzioa	[_3M, _3M, null, null, normalize, null, negate, null]	[normalize, negate, null, contrast, contrast, contrast, negate, contrast]
Denbora	1h 23m 05s	48m 24s
Indibiduoak = 100 Ausaz Sortuta = 100 Generazioak = 12	78,76	79,10
Lortutako Soluzioa	[_3M, negate, negate, _3M, normalize, null, contrast, negate]	[normalize, contrast, null, gaussian-blur, contrast, contrast, normalize, negate]
Denbora	5h 29m 29s	2h 14m 48s

4.7 Taula: 4.4 Irudian, Algoritmo Genetikoaren Asmatze Tasak

48m 24s behar izan ditu. Denbora hori zenbaitentzat asko izan badaiteke ere, aurreko adibideak ikusi eta gero, bilaketa azkar bezala kalifika dezakegu.

4.8 irudian ikusi dezakegun bezala, %56,16-ko asmatze-tasatik %79,79-ko asmatze-tasa lortu da.



4.8 Irudia: 4.4 irudiak heuristikoan lortutako aurreprozesaketarik onena

Irudi Guztientzako Aurreprozesatze Komuna

Esperimentazioan ikusten hari den moduan, emaitzarik onenak lortzeko, irudi bakoitzak bere aurreprozesatze propioa behar du. Hala ere, zenbaitetan ez du merezi horrenbeste denbora zain egotea, irudi eraldaketa egoki bat lortzen. Horrenbestez irudi guztientzat 'ona' den aurreprozesatze komun bat aurkitzen saiatuko da, esperimentazioan zehar lorturako emaitzak erreparatuz.

Horretarako 4.8 taula osatu da, bertan irudi bakoitzarekin lortu den soluziorik onena beste irudiei aplikatuko zaie. Lehenengo errenkadan irudi originalekin lortutako emaitzak ezarriko dira, ondorengo hiru[4] errenkadetan transformazioen biko konbinazio onenen emaitzak eta azken lau errenkadetan heuristikoen bidez lortutako emaitzarik onenak. Errenkada bakoitzaren azken zutabean, asmatze-tasa guztien batezbestekoa kalkulatuko da, aurreprozesatzerik egokiena zein den aukeratzeko. Hau, pixel kopuru originalekin soilik egingo da.

Irudiaren Transformazioa	1.Ir 4.1	2.Ir 4.2	3.Ir 4.3	4.Ir 4.4	Batezbeste
Pixel Kopuru Originala					
Originala	61,53	82,96	90,66	56,16	72,83
Contrast + Normalize	76,92	82,09	86,00	45,54	72,64
Contrast + Negate	76,92	83,62	84,66	52,39	74,40
Moran 3 + Contrast	00,00	74,23	30,66	78,08	45,74
Heuristikoa1	84,61	81,44	88,66	58,56	**78,32**
Heuristikoa2	30,66	88,64	42,00	70,20	57,86
Heuristikoa3	69,23	79,25	94,66	64,04	76,80
Heuristikoa4	00,00	00,00	01,33	78,76	20,02

4.8 Taula: Irudien aurreprozesatze komuna aurkitzeko taula

- **Heuristikoa1 ⇒ Normalize + Gaussian-Blur + Contrast**
- Heuristikoa2 ⇒ Normalize + Normalize + Gaussian-Blur + Moran3
- Heuristikoa3 ⇒ Normalize + Normalize + Contrast + Contrast + Contrast + Contrast
- Heuristikoa4 ⇒ Moran3 + Negate + Negate + Moran3 + Normalize + Contrast + Negate

Batezbestekoei begiratuz, *Heuristikoa1* soluzioak lortu ditu emaitzarik onenak. Hau 4.1 irudiaren aurreprozesatzerik hoberena da, *Normalize*, *Gaussian-Blur* eta *Contrast* transformazioen konbinazioa alegia.

Horrenbestez, aurreprozesatzea burutzen denbora galdu nahi ez duten erabiltzaileek, transformazioen konbinaketa honen bidez aurreprozesatuko dute irudia.

Hiztegiak

4.9 taulan ikusi daitekeen bezala, lehenengo errenkadan *hunspell*-ekin zuzendu gabeko asmatze tasak, bigarrengoan, zuzenketa automatikoarekin lortutako asmatze tasak eta hi-

[4] 4.3 irudiak, biko konbinazioekin baino irudi originalarekin emaitza hobea lortu baitu

rugarrengoak zuzenketa semiautomatikoarekin lortutakoak daude. Zutabeetan berriz, esperimenturako aukeratu diren lau irudiak.

Hiztegiekin Zuzentzen	1.Ir 4.1	2.Ir 4.2	3.Ir 4.3	4.Ir 4.4
Zuzendu Gabe	**84,61**	88,64	**94,66**	**78,76**
Zuzenketa Automatikoa	76,92	89,08	93,33	76,71
Zuzenketa SemiAutomatikoa	84,61	**89,73**	**94,66**	67,46

4.9 Taula: Hiztegiaren bidezko zuzenketaren asmatze-tasak

4.9 taulako emaitzetan ikus daitekeen bezala, ez dira oso emaitza onak lortu atal honetan. Bigarrengo irudian bakarrik lortu da, zuzenketa automatikoarekin zein semiautomatikoarekin asmatze-tasak zertxobait igotzea. Bestelako kasu guztietan, asmatze-tasak jaitsi edo berdin geratu dira.

5. KAPITULUA

Garapena

Diseinu orokorra azaltzerako orduan, proiektua hiru zati nagusitan banatu bada ere, inplementazioa azaltzeko garaian, zertxobait gehiago barneratuko gara.

Garapenaren azalpenak ematerako garaian, kodearen zatirik garrantsitsuenak soilik agertuko dira, https://github.com/pellux/digitalizaizioAutomatikoa_GAP estekan ikuskatu edo jaitsi daiteke gainerako kode guztia.

Hasieraketa

Diseinuan aipatu lez, programari pasatutako argumentu kopuruaren arabera, bide bat edo bestea hartuko du exekuzioak. Argumentu kopurua bost bada, bilaketa heuristikoa egingo du, hiru bada defektuzko aurreprozesatzearekin konformatuko da eta argumentu kopurua bi kopuru horietako bat ez bada berriz, programa errore batez amaituko da.

Bilaketa Heuristikoa

Aurrekarietan aipatu moduan, bilaketa heuristikoaren helburua, irudiei aurreprozesatze egoki bat emateko irudi eraldaketen sekuentzia bat lortzea da. Bilaketa honen bitartez, soluzio optimora iristea bermatzen ez badugu ere, soluzio on bat aurkitzea espero da beti.

Python programa honek hiru argumentu behar ditu bere funtzionalitatea betetzeko. Lehe-

nengoa, eraldaketak aplikatuko dizkiogun irudi originalaren *path*-a izango da. Bigarrena, lehenengo argumentuko irudia testu digital moduan zuzen idatzita. Hau bilaketa heuristikoko ebaluazioa burutzeko erabiliko da, OCR egitean lortzen den testua eta hau konparatuz. Azkenik, hirugarren argumentua hizkuntza izango da. Tesseract-ek OCR egitean, hizkuntza erabakitzeko aukera eskaintzen baitu. Euskara, gaztelania eta ingelesa izango dira programak onartzen dituen hizkuntzak.

```
#eraldatuko dugun argazki originalaren path-a
argazkia = sys.argv[1]
#OCR-egitean fallo kopurua begiratzeko eskuz idatzitako testua
textOCR = sys.argv[2]
#tesseract-i hizkuntza pasatzeko
hizkuntza = sys.argv[3]
```

5.1 Irudia: Argumentuak

Lehenik eta behin, lista bat sortu da (5.2) [1] [2], bertan soluzioak izan ditzakeen transformazio guztiak sartuz. Programa parametrizatuta dagoenez, lista honetatik erraz sartu edo kendu daitezke transformazioak, horrela proba desberdinak egitea ahalbidetuz. Transformazioen artean *null* aurkitzen da. Hau, trasnformazio nulua da, hots, ez dio irudiari inongo aldaketarik eragingo. Bestetik, *index values* programaren bidez egiten diren transformazioei karaktere berezi bat erantsi zaie hasieran, hauek ondoren erraz identifikatzeko.

```
#eraldaketa mota guztiak
eraldaketak=["null","_1M","_2M","_3M","_1E1","_2E1","_3E1","negate",
             "2edge","4edge","6edge","8edge","contrast","normalize"]
```

5.2 Irudia: Eraldaketa Guztien Lista

Behin argumentuak lortuta, erabiltzaileri eskatuko zaizkio programak behar dituen datu gehigarriak. Hauek, argumentu bezala pasatzea posible izango litzateke, baina erabiltzaileari momentuan erabakitzen ustearen hautua egin da. Honen arrazoia, zera da, balio hauen arabera programaren exekuzio denbora erabat aldatzen dela. Beraz, erabiltzaileak bilaketa azkar bat egitea nahi badu, balio txikiak ezarriko ditu bertan. Bilaketa sakonago eta luzeagoa bat egitea nahi badu berriz balio altuagoak txertatuko ditu. Lehenengo *input* [3]-a indibiduoen [4] kopurua izango da, hau da, belaunaldi batetik bestera pasatuko diren

[1] pythonen exekutatzeko kode guztia errenkada batean egon beharko litzateke, hemen hobeto ikusteko jarri da horrela. Gauza bera egingo da ondorengo kode zatietan ere.

[2] Ez dira zertan eraldaketa hauek erabiliko bilaketa heuristikoan. Hori esperimentazioan erabakiko da

[3] Erabiltzaileak teklatu bidez sartuko ditu balioak

[4] Indibiduoei soluzioak ere deitzen zaie

soluzioen kopurua. Bigarrena, indibiduoak hasieratzean, hausaz sortuko direnen kopurua izango da, ondoren azalduko da zertan datzan horrek. Hirugarrenik, algoritmo genetikoaren jenerazio kopurua izango da. Honek exekuzioaren denboraren luzeran erabat eragingo du. Egia esan, ez da erraza parametro hau ondo aukeratzea. Txikiegia jartzen bada, algoritmoak ez baitu konbergituko eta handiegia jartzen bada berriz, jadanik konbergituta dagoen algoritmoa alferrik exekutatzen egotea izango litzateke.

5.3 irudian ikusi daitekeen bezala, hiru parametro horiek osokoak izan behar dute. Ez bada horrela, erabiltzaileri berriz sartzeko eskatuko zaio.

```
while True:
    try:
        n_indibiduaoak = int(input("Zenbat indibiduo ?? ->"))
        n_ausaz = int(input("Zenbat ausaz ?? ->"))
        n_generazioak = int(input("Zenbat generazio ?? ->"))
        break
    except ValueError:
        print("Denek ZENBAKI OSOKOAK izan behar dute!\n")
```

5.3 Irudia: Erabiltzaileari Eskatzen Zaizkion Argumentuak

Parametro guzti horiek lortu ondoren, *onenaIndibi* izeneko aldagaia sortuko da. Hau 5.4 irudian ikus daitekeen klasearen objektu bat da. Objektu honek, transformazioen lista eta honen ebaluazioa izango ditu bere baitan. 5.5 irudian ikusi daitekeen bezala, hasiera batean lista hutsa izango da eta bere ebaluazioa balio negatibo bat.

```
class indibiduo:
    #eraikitzailea
    def __init__(self, eral, ebal):
        self.eraldaketak = eral
        self.ebaluazioa = ebal
```

5.4 Irudia: Indibiduo Klasea

```
#soluziorik onena gordeko duen indibiduoa
onenaIndibi = indibiduo([],-1)
```

5.5 Irudia: Indibiduoreik Onena Gordetzeko Objetua

Hasierako populazioa sortzeko berriz, bi teknika erabili dira. Hala ere, sortutako indibiduo guztiek, eraldaketen luzera berdina izango dute, 8-koa hain zuzen ere. Lehen teknika, 5.6 irudiko lehen blokean ikusi daitekeenez, transformazio guztien biko konbinazio posible

guztiak sortzen ditu. Horrela, 5.2 irudian bezala, 14 transformazio desberdin izanik eta biko konbinazioak osatu nahi direnez, $14^2 = 196$ indibiduo sortuko lirateke. Hauek bi transformazio soilik izango dituztenez, gainerako guztiak *null* izango dira.

Bigarren teknika berriz erabat ausazkoa izango da. 5.6 irudiko bigarren blokean ikusi daitekeen bezala, sortuko diren indibiduo kopurua, 5.3 irudiko *n_ausaz* aldagaiak definituko du.

```
def hasierakoPopulazioa(eraldaketak):

    indibiduoak = []
    #eraldaketa denak denekin 14^2 indibiduo sortuko ditugu alde batetik
    for e1 in eraldaketak:
        for e2 in eraldaketak:
            eral = [e1,e2,"null","null","null","null","null","null"]
            inb = indibiduo(eral,-1)#horaindik ebaluatu gabe baitago
            indibiduoak.append(inb)

    # beste n_indibiduo guztiz ausaz
    for i in range(n_ausaz):
        eral = []
        for i in range(8):
            r = random.randrange(0, len(eraldaketak))
            eral.append(eraldaketak[r])
        inb = indibiduo(eral,-1)#horaindik ebaluatu gabe baitago
        indibiduoak.append(inb)

    return indibiduoak
```

5.6 Irudia: Hasierako Populazioa Sortu

Beraz hainbat indibiduo daude baina hauei ebaluazio balioa falta zaie. Ebaluazioa lortzeko, indibiduoari dagokion eraldatutako irudia lortu behar da lehenik. Ondoren irudi horri OCR egin eta zuzen idatzita dagoen testuarekin konparatuz lortuko baita indibiduo bakoitzaren ebaluazioa. Irudiei eraldaketak aplikatzeko, *irudiakEraldatu* izeneko funtzio bat sortu da. Funtzio honen eginbeharra, indibiduo bakoitzaren eraldaketak irudi originalari aplikatu eta indibiduoaren izenarekin gordetzea da. 5.7 irudian ikusten dena egingo luke, 13. indibiduaren eraldaketak [4edge, negate, null, null, null, null, _2E1, null] eta irudi originalaren izena 'adibidea' izango balira.

Behin $indibi1 \ldots indibi[n\_indibiduo - 1]$ irudiak edukita, hauek ebaluatu behar dira. Horretarako, *irudiakEbaluatu* funtzioa dago, 5.8 irudian ikus daitekeen bezala. Indibiduo bakoitzeko, OCR egin eta zuzen idatzitako testuarekin konparatuko da.

Testuak konparatzeko *testuakKonparatu* funtzioa erabiltzen da. Funtzio honek bi argumentu jasotzen ditu, indibiduoak lortu duen testua eta testu zuzena. Testu bakoitzeko hitzak lista banatan sartzen dira lehenik. Ondoren lista bateko hitzak beste listan aurkitzen

5.7 Irudia: Irudiei eraldaketak Aplikatzen

```
def irudiakEbaluatu(indibiduoak, hiz):

    #indibiduo guztiak ebaluatu
    for i in range(len(indibiduoak)):
        komandoa = "tesseract -l "+hiz+" temp/indibi"+str(i)+" temp/indibi"+str(i)
        print(komandoa)
        args = shlex.split(komandoa)
        subprocess.call(args)

        #konparatu
        ehunekoa = testuakKonparatu("temp/indibi"+str(i)+".txt", textOCR)
        #ehunekoa = testuakKonparatu(textOCR, "indibi"+str(i)+".txt")
        indibiduoak[i].ebaluazioa = ehunekoa

    return indibiduoak
```

5.8 Irudia: Irudiak Ebaluatu

saiatzen da. Bilatzen badu kontagailua igo eta listatik ezabatzen da. Horrela, testu zuzenarekin duen antza ehunekotan adieraztea lortzen da.

Dagoeneko indibiduo bakoitzak bere ebaluazioa jarrita izango luke. Baina, algoritmo genetikoa aurrera eramateko indibiduo kopuru handiegia dago. Lehen aipatu bezala, lehen teknikaren bidez 194 indibiduo sortu dira eta *n_ausaz* gehiago bigarren teknikaren bidez. Beraz, *elekzioaLehena* funtzioarekin, *n_indibiduo* kopuruko aukeraketa bat egingo da, 5.9 irudian ikus daitekeen bezala. Funtzio honek, indibiduoak bi zatitan banatzen ditu, lehen teknikarekin sortutakoen eta bigarrenarekin sortutakoen arabera. Bi zatietako indibiduoak beraien ebaluazioaren arabera ordenatzen ditu ondoren. Bukatzeko, $n\_indibiduo/2$ indibiduo hartzen ditu lehen zakutik eta beste horrenbeste bestetik. Hemen gertatu daitekeen arazoa, $n\_ausaz < (n\_indibiduo/2)$ betetzea da. Kasu honetan programak exekuzio errore bat jasango luke.

```
#Erdiak batetik eta beste erdiak bestetik
def elekzioaLehena(indibiduoak):

    kop = len(eraldaketak)*len(eraldaketak)
    lag1 = []
    lag2 = []

    print(len(indibiduoak))

    #sortutako moduaren arabera sailkatu
    lag1 = indibiduoak[0:kop]
    lag2 = indibiduoak[kop:len(indibiduoak)]

    #ordenatu
    lag1.sort(key=operator.attrgetter('ebaluazioa'),reverse = True)
    lag2.sort(key=operator.attrgetter('ebaluazioa'),reverse = True)

    #sailkapen bakoitzetik indibiduen erdiak hartu
    lag1 = lag1[0:int(n_indibiduaoak/2)]
    lag2 = lag2[0:int(n_indibiduaoak/2)]

    return lag1 + lag2
```

5.9 Irudia: Elekzioa Lehena

Behin *n_indibiduo* aukeratuta, *n_generazio*-ko begizta nagusian sartuko da. 5.10 irudian ikus daitekeen bezala, begiztaren barruan, populazio berriko irudiak eraldatu, hauek ebaluatu, ordu-arteko emaitzarik hoberena eguneratu eta populazio berria sortzen da, hau *n_generazio* aldiz egingo delarik.

Begiztako lehenengo pausoak, irudiak eraldatzea eta irudiak ebaluatzea, aurrez azaldu direnez, hirugarren pausora jauzi egingo da. Ordu-arteko emaitzarik onena eguneratzeko, aurrez aipaturiko *onenaIndibi* aldagaia erabiltzen da. Momentuko indibiduo guztiak

errekorritzen dira, *onenaIndibi*-ren ebaluazioa baino handiagoa duen indibiduo bat bilatu nahian. Aurkitzen bada, *onenaIndibi* eguneratuko da.

```
for generazio in range(n_generazioak):
    print(str(generazio) + ". Generazioa")

    irudiakEraldatu(indibiduoak)
    indibiduoak = irudiakEbaluatu(indibiduoak, hizkuntza)

    #orain arteko onena hobetu badu
    for i in range(len(indibiduoak)):
        if(indibiduoak[i].ebaluazioa>onenaIndibi.ebaluazioa):
            onenaIndibi = indibiduo(indibiduoak[i].eraldaketak, indibiduoak[i].ebaluazioa)


    print("****************************************")
    print("****************************************")
    print(onenaIndibi.eraldaketak)
    print(onenaIndibi.ebaluazioa)
    print("****************************************")
    for ind in indibiduoak:
        print(str(ind.ebaluazioa)+"  "+str(ind.eraldaketak))
    print("****************************************")
    print("****************************************")
    sleep(2)

    gurasoak = elekzioa(indibiduoak)
    berriak = gurutzaketak(gurasoak)
    #berriak = mutazioa(berriak)
    #indibiduoetatik 9/10 berriak sortu dira
    #1/10 berriz, lehengo populazioko onenak izango dira
    indibiduoak = populazioBerria(indibiduoak, berriak)
```

5.10 Irudia: Algortimo Genetikoaren Begizta Nagusia

Begiztaren amaieran, populazio berria sortzeko berriz, bi teknika erabiltzen dira. Populazio berriaren %90, gurasoen gurutzaketaren bidez lortuko da. Beste %10-a berriz momentuko populazioko indibiduo hoberenen eskutik, hauek mutazioa jasateko aukera izango dute.

Gurasoen gurutzaketa burutzeko, lehenik eta behin gurasoak aukeratu behar dira. Horretarako, erruleta-hautespena deritzon teknika erabili da (ikusi 2.1.2 atala). 5.11 irudian ikus daitekeenez, ausaz lortzen dira gurasoak, baina ebaluazio altuagoa duten indibiduoek, gurasoa izateko aukera gehiago dute.

Behin gurasoak aukeratuta, hauek gurutzatu behar dira. 2.4 atalean irakurri daitekeen *puntu bakarreko gurutzaketa* teknika erabili da horretarako. 5.12 irudian ikusi daitekeen bezala, gurutzaketa %90-eko kasuetan egingo da. Kasu horietan *puntua* ausaz lortu eta gurutzaketa buruko delarik.

Gurasoen gurutzaketatik lortutako indibiduo berriak izanik, bigarren teknikako indibiduoak lortzea falta da. Horretarako, indibiduoak beren ebaluazioaren arabera ordenatu eta

```python
def elekzioa(indibiduoak):

    denera = 0
    gehikuntzak = []
    gurasoak = []
    aurrekoa = 0
    for ind in indibiduoak:
        denera += ind.ebaluazioa
        gehikuntzak.append(denera)

    #gurasoak hautatu(roulete wheel selection)
    for i in range(int(9/10*n_indibiduaoak)): #3/4 guraso
        #ausazko zenbakia lortu
        r = random.randrange(0, int(denera))
        #ausazko zenbakia zein indizeri dagokion aurkitu
        for j in range(len(indibiduoak)):
            if r < gehikuntzak[j]:
                gurasoak.append(indibiduoak[j])
                break
    return gurasoak
```

5.11 Irudia: Erruleta-hautespenaren kodea

```python
#guraso kopuruak bikoitia izan behar du
def gurutzaketak(gurasoak):

    berriak = []
    for i in range(int(len(gurasoak)/2)):
        #print("indizea "+str(i)+" => "+str(indizeak[i]))
        gur1 = gurasoak[i]
        gur2 = gurasoak[i+1]
        #gurutzatu egingo ditugu(one point crossover)
        r = random.randrange(1, 101)
        #%90-eko probabilitatearekin egingo dugu gurutzaketa
        if(r > 10):
            #gurutzaketa zein puntutan egingo den erabaki
            r_ind = random.randrange(8)
            berria1 = gur1.eraldaketak[0:(r_ind)] + gur2.eraldaketak[r_ind:8]
            berria2 = gur2.eraldaketak[0:(r_ind)] + gur1.eraldaketak[r_ind:8]
            indBer1 = indibiduo(berria1,-1)
            indBer2 = indibiduo(berria2,-1)
            berriak.append(indBer1)
            berriak.append(indBer2)
        else:
            berriak.append(gur1)
            berriak.append(gur2)

    return berriak
```

5.12 Irudia: Puntu Bakarreko Gurutzaketaren Kodea

ebaluaziorik handiena duten n_indibiduo/10 indibiduo aukeratuko dira. Ondoren, 10-eko probabilitateaz mutazioa eragingo zaie. 5.13 irudian erreparatuz, mutazioa jasango duen posizioa ausaz lortuko da eta baita mutazioa zein izango den ere.

Horrenbestez, bi tekniken bidez lortutako indibiduo berriak batuz, populazio berria eskuragarri dago.

```
def mutazioa(berriak):

    mutatuakEdoEz = []
    for berri in berriak:
        r = random.randrange(1, 101)
        #%10-eko probabilitatearekin egingo dugu mutazioa
        if(r > 90):
            non_ind = random.randrange(8)
            zer_ind = random.randrange(len(eraldaketak))
            berri.eraldaketak[non_ind] = eraldaketak[zer_ind]
            mutatuakEdoEz.append(berri)
        else:
            mutatuakEdoEz.append(berri)

    return mutatuakEdoEz
```

5.13 Irudia: Mutazioa

Hau *n_generazio* aldiz eginez, *onenaIndibi*-n bilaketa heuristiko osoan lortutako transformazioen sekuentziarik onena gordeta izango da. Bihurketen sekuentzia eta honen ebaluazioa fitxategi batean gordetzean, bilaketa heuristikoa amaitutzat izango litzateke.

OCR

Bilaketa Heuristikoaren bidez edo aplikazioan defektuz dagoenarekin, behin irudiaren aurreprozesatzea hautatuta denean, irudia eraldatuko da. Horretarako, *irudiaEraldatu* funtzioaz baliatuko da. Funtzio hau, *Bilaketa Heuristikoak* atalean azaldutako *irudiakEraldatu* funtzioaren oso antzekoa da. Berezitasun nagusia, batak indibiduo guztien irudi-testuari egiten diola eraldaketa eta besteak berriz OCR egitea nahi den testu-irudiari soilik.

Behin testu-irudia aurreprozesatuta dela, *Optical Character Recognition* egingo da. Horretarako *tesseract* aplikazioa erabiliko da. Aplikazioa martxan jartzeko, *Linux*-eko komando lerroaz baliatuko da eta hiru argumentu jarriko dira komandoaren atzetik: Testua zein hizkuntzatan dagoen idatzita, OCR egitea nahi den testu-irudia eta emaitza gordetzea nahi den fitxategiaren izena. Ikus 5.14

```
#tesseract aplikatu
komandoa = "tesseract -l "+str(hizkuntza)+" eral"+str(textu_osoa)+" emaitzaMomentu"+str(textu_osoa)
args = shlex.split(komandoa)
subprocess.call(args)
```

5.14 Irudia: Tesseract Aplikazioari dei egin

Hiztegiak

Tesseract aplikazioaren laguntzaz, iruditik testu digitala lortuta izango litzateke jadanik. Hala ere testu hau ez da erabat zuzena izaten. *Tessreact* saiatzen da testua zuzentzen (horretarako pasatzen zaio hizkuntza) baina ez du lortzen erabat zuzentzea. Beraz, lortutako testu digitalari beste hiztegi bat aplikatzea pentsatu da, *Hunspell* hain zuzen ere.

Hala ere, erabiltzailearen esku utziko da zuzenketa hori egin nahi den edo ez erabakitzea. Beraz, hiru aukeren artean erabaki beharko du erabiltzaileak: testua ez zuzentzea, testua automatikoki zuzentzea edo testua semiautomatikoki zuzentzea. Datozen ataletan sakonduko da hauetako bakoitza. Testua ez zuzentzea, ez da azalduko, honek, jadanik lortuta dagoen testu digitala itzuli eta aplikazioa amaituko baitu.

Zuzenketa Automatikoa

Zuzenketa Automatikoaren kasuan, erabiltzaileak ezer egin gabe, programa testua zuzentzen saiatuko da. Horretarako *python*-eko *Hunspell* liburutegia erabiliko da.

ZuzenketaAutomatikoa funtzioan, liburutegi honen bitartez, testu digitaleko hitz bakoitza banan-banan prozesatuko da. Lehendabizi, hitza, dagokion hizkuntzako hiztegian aurkitzen den edo ez begiratuko da. Aurkitzen bada, hitza zegoen bezalaxe utziko da. Baina ez bada hiztegian aurkitzen, hitz horren antza duten proposamenak lortzen ditu *Hunspell*-ek. Antz gehien duten hitzetatik, gutxiago dutenera ordenatzen dituenez, zuzenketa automatikoaren kasuan, listako lehenengo proposamenagatik aldatuko da hiztegian agertzen ez den hitza.

Zuzenketa SemiAutomatikoa

Zuzenketa semiautomatikoaren kasuan, programa eta erabiltzailearen artean zuzenduko da testua. Kasu honetan ere, *python*-eko *Hunspell* liburutegia erabiliz.

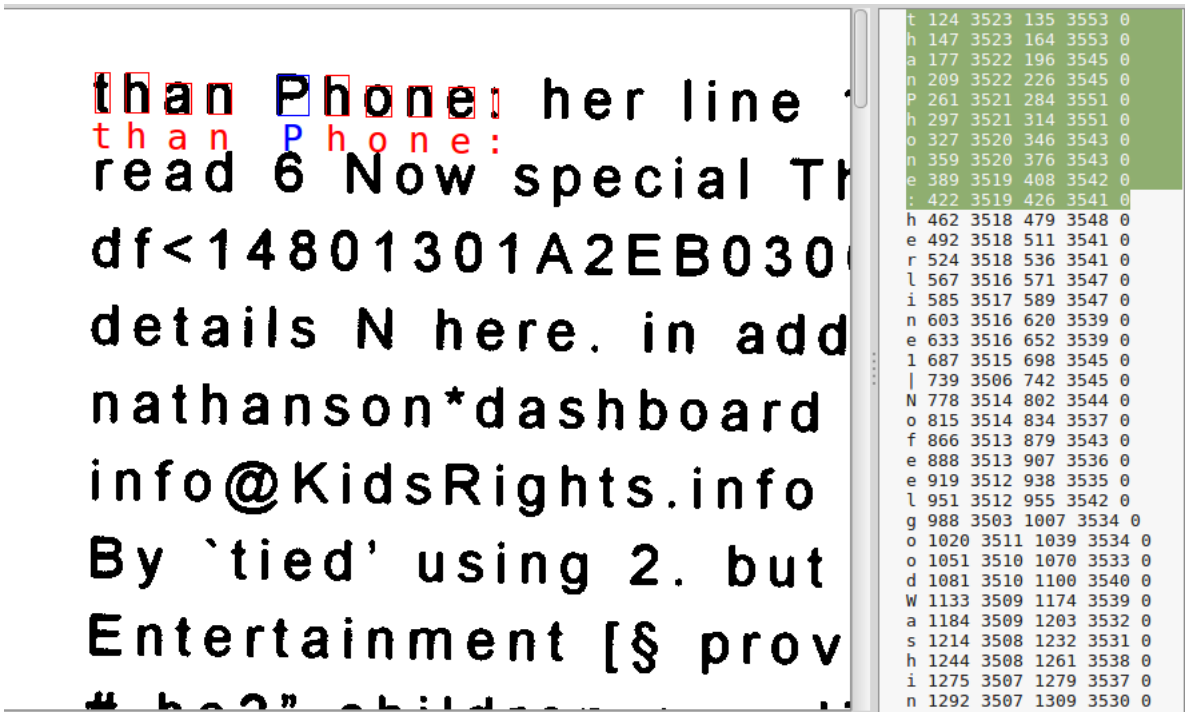
ZuzenketaSemiAutomatikoa funtzioari deitu baino lehen, fitxategi berezi bat eskuratu beharko da, karaktere bakoitzaren *box*-a definitzen duena hain zuzen ere. Fitxategi hau lortzeko *tesseract*-ez baliatuko da. Honakoa da fitxategi hauen itxura:

```
t 124 3523 135 3553 0
h 147 3523 164 3553 0
```

```
a 177 3522 196 3545 0
n 209 3522 226 3545 0
P 261 3521 284 3551 0
h 297 3521 314 3551 0
o 327 3520 346 3543 0
n 359 3520 376 3543 0
e 389 3519 408 3542 0
: 422 3519 426 3541 0
h 462 3518 479 3548 0
e 492 3518 511 3541 0
r 524 3518 536 3541 0
l 567 3516 571 3547 0
i 585 3517 589 3547 0
n 603 3516 620 3539 0
e 633 3516 652 3539 0
1 687 3515 698 3545 0
| 739 3506 742 3545 0
N 778 3514 802 3544 0
o 815 3514 834 3537 0
f 866 3513 879 3543 0
e 888 3513 907 3536 0
```

- 1.ZUTABEA: OCR egitean lortutako karakterea.
- 2.ZUTABEA: 1.zutabeko karakterearen *box*-aren ezkerreko muga
- 3.ZUTABEA: 1.zutabeko karakterearen *box*-aren beheko muga
- 4.ZUTABEA: 1.zutabeko karakterearen *box*-aren eskuineko muga
- 5.ZUTABEA: 1.zutabeko karakterearen *box*-aren goiko muga
- 6.ZUTABEA: Zein horrialdetako karaktera den.

Lehen esan bezala, errenkada bakoitzak hizki baten *box*-a definitzen du. 5.15 adibidean ederto ikusi daitekeen bezala.

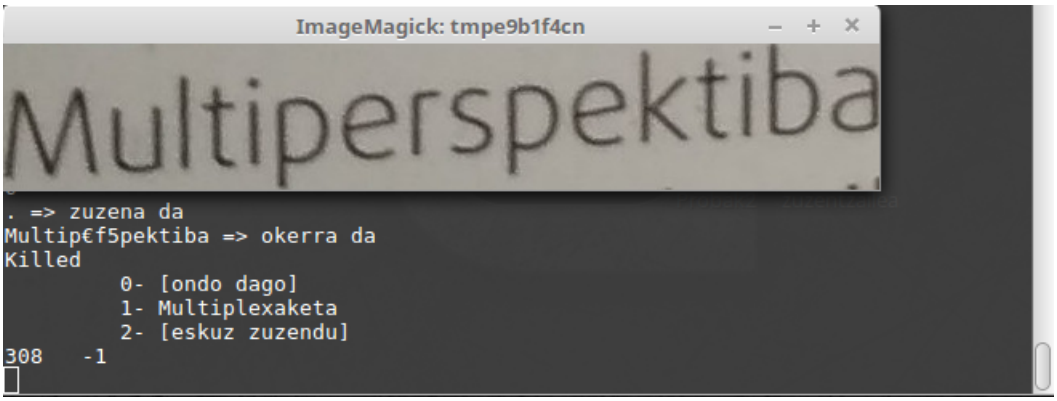


5.15 Irudia: Box-en fitxategiaren adibidea

Behin fitxategi berezi hori lortuta, *zuzenketaSemiAutomatikoa* funtzioari dei egingo zaio. Zuzenketa semiatukomatikoaren ideia, hiztegian agertzen ez diren hitzen irudi zatia pantailaratzea da, eta erabiltzaileak erabakitzea benetan zer jartzen duen. 5.16 adibidean ikusi daitekeen bezala, *Multipf5pektiba* hitza hiztegian ez dagoela esaten du aplikazioak. Beraz, hau zuzentzeko hitz horri dagokion irudiaren zatia pantailaratzen da. Kasu honetan, proposamenen artean, hitz zuzena aurkitzen ez bada ere, erabiltzaileak *[eskuz zuzendu]* aukeraren bitartez, hitza zuzendu ahalko luke.

Ideia hori erabat zuzen garatzea ez da batere erraza ordea. Lehenik eta behin, *box*-ak dituen fitxategiak karaktereen *box*-ak dira. Baina programa hitzekin lan egiten ari denez, karaktereen *box*-etan oinarrituz, hitzen *box*-ak lortu beharko dira. Bestetik, *tesseract*-ek hiztegien bidez hitzak zuzentzen saiatzen da, baina *box*-ak dituen fitxategia hiztegien zuzenketa egin baino lehen lortzen da. Beraz, alde batetik, hiztegiarekin zuzendutako hitzen fitxategia dago eta bestetik, hiztegiekin zuzendu gabekoena. Bi fitxategi horien artean hitzak parekatzeko teknikaren bat ere beharko da beraz.

Hizkien *box*-etatik hitzenetara pasatzeko, testu-irudi bakoitza desberdina denez, lehenik eta behin hitzen arteko tartea zenbatekoa den entrenatu behar da. Horretarako, *tarteaEntrenatu* funtzioa garatu da. Funtzio honek, hizki batetik bestera dauden tarte guztien bataz-



5.16 Irudia: Zuzentzaile Semiautomatikoaren Adibidea

bestekoa kalkulatzen du. Batezbesteko hau kalkulatzeko garaian, lerro jauzietan dauden tarteak alboratzen dira. Funtzio honek, batezbestekoa baino bi unitate handiagoa den tartea itzuliko du. Honela, hitz batetik bestera tarte hori baino gutxiago badago, hitz bera dela suposatuko dugu eta tarte hori baino handiagoa bada berriz, beste hitz baten hasiera dela.

Beraz hitz horien *box*-ak egoki gordetzeko, *hitza* klasea sortu da (ikus 5.17 irudia). Klase honetako objektuek, besteak beste, hitzaren karaktere katea eta *box*-a definitzeko lau balioak dituzte. Balio hauek pixka bat eraldatuta gorde dira, ondoren *ImageMagick*-eko *convert* komandoa erabiliko baita *box*-ak ebakitzeko:

```
class hitza:
    #objetua sortu
    def __init__(self, num):
        self.num = num
        self.minLeft = 10000000
        self.maxRight = -1
        self.minTop = 10000000
        self.maxBotton = -1
        self.xDim = -1
        self.yDim = -1
        self.xPos = -1
        self.yPos = -1
        self.hitz = ""
```

5.17 Irudia: Hitza Klasea

- xDim $\Rightarrow$ *Box*-ak zabaleran dituen pixel kopurua. Hau kalkulatzeko nahikoa da, hitzaren *box*-ak ezker mugan duen baliorik txikiena, eskuin mugan duen baliorik handienarekin kentzea. Emaitza balio absolutura pasaz.

- yDim ⇒ *Box*-ak altueran dituen pixel kopurua. Hau kalkulatzeko nahikoa da, hitzaren *box*-ak beheko mugan duen baliorik txikiena, goiko mugan duen baliorik handienarekin kentzea. Emaitza balio absolutura pasaz.

- xPos ⇒ *Box*-aren zabalerako hasierako pixela. Hau hitzaren *box*-ak ezkerreko mugan duen baliorik txikienaren berdina da.

- yPos ⇒ *Box*-aren altuerako hasierako pixela. Hau kalkulatzeko, irudiaren dimentsioak jakin beharra ditugu [5]. Horretarako, *identify* [6] komandoa erabiltzen dugu. Behin irudiaren dimensioak jakinda, altueraren dimentsioari hitzaren *box*-ak goiko mugan duen baliorik handienarekin kentzea.

Beraz, *box*-ak dituen fitxategi osoa prozesatuz, *Hitza* objektuez osatutako lista bat lortuko da. Lehen aipaturiko lehenengo arazoa ebatzita izango litzateke, karaktereen *box*-etatik hitzen *box*-ak lortzea alegia.

Bigarren arazoa zailagoa da ordea, *tesseract*-ek hiztegia pasatuta daukan testuko hitza, *Hitza* objektuez osatutako listako zeini dagokion erabaki behar da. Behin hori eginda, *Hunspell*-ek hiztegietan bilatzen ez dituen hitzak pantailaratu eta erabiltzaileak zuzendu ahal izateko.

Hiztegia pasatuta daukan hitza, listako zeini dagokion jakitea ez da hain erraza ordea. Oso antz handia duten hitzak soilik onartzen badira, askotan ez du parekatzeko moduko hitzik aurkituko. Hain antz handia ez duten hitzak onartzen badira berriz, hitz horri ez dagokion beste bat aukeratzeko arrisku handia dago.

lortuIndBoxHitza funtzioaren bidez lortzen dugu bi hitzak parekatzea. Funtzio honek hitz bera dela esango du beraien artean %75-eko antza badute.

Horrela, hitzez hitz zuzentzen joango da, lehen esan bezala, hiztegian agertzen ez diren hitzak, 5.16 irudian bezala pantailaratuz. Horrela, erabiltzaileak hitza dagoen bezala laga, *hunspell*-ek egindako proposamen bategatik aldatu edo berak eskuz idatzitako beste hitz batengatik aldatu ahal izango du.

[5] Altuerako pixelak, goitik behera kontatu beharrean, behetik gora kontatzen ditu. Guk erabiliko dugun *convert* komandoak ez ordea. Horregatik lortu behar izan dugu, irudiaren dimentsioa.

[6] Linux-eko komandoa da eta irudien propietateak ikusteko balio du

6. KAPITULUA

Ondorioak eta Etorkizuneko Lanak

Ondorioak

Gogora dezagun hasiera batean bete behar genuen helburu nagusia: testuen irudiak izanik, modu automatiko batean, irudi horiek testu bihurtzeko *Tesseract* aplikazio libreari hobekuntzak egitea zen helburu nagusia. Helburu nagusi hori betetzeko, hasiera batean, hobekuntzak hiru modu desberdinetan egitea pentsatu zen: Irudiak aurreprozesatuz, OCR algoritmoak hobetuz eta hiztegien bidez testuak zuzenduz. Proiektuan zehar hasierako plangintza aldatuz joan da eta Irudien Aurreprozesatzea izan da proiektu honetan gehien sakondu den atala. Testua hiztegiarekin zuzentzearen proba batzuk egin dira eta OCR algoritmoak ez dira ukitu ere egin azkenean.

Irudien aurreprozesatzean zentratuz, esan beharra dago testu desberdin bakoitzak aurreprozesatze desberdin bat behar duela emaitzak hobetzeko. Ondorioz, irudi guztientzako aurreprozesatze egoki bat aurkitzen saiatu bagara ere, aplikazio nagusian, testua digitalizatzen hasi aurretik, testu horrentzako bilaketa heuristiko bat egiteko aukera jarri da, testu propioarentzako aurreprozesatzerik hoberena aurkitzeko.

Bilaketa heuristiko honen bidez, emaitzak hobetzea lortzen bada ere, zenbait kasutan denbora asko behar du soluzio egoki batera iristeko. Horrenbestez ez du beti merezi bilaketa heuristiko luze bat egitea. Adibidez, 10 orrialdeko testu bat digitalizatu nahi bada, ez dugu lau orduko bilaketa bat burutuko. Baina, itxura berdina duten 600 orrialdeko 10 liburu digitalizatu nahi badira, litekeena da lau orduko bilaketa heuristikoa egiteak merezi izatea, OCR egiten hasi aurretik aurreprozesatze egoki bat aurkitzeko.

Beraz, bilaketa heuristikoaren erabilera asko aldatu daiteke testu mota, kopuru eta erabiltzailearen asmoen arabera. Hortaz, programa nagusian zehar erabiltzaileari heuristikoa egin edo ez eta egitea erabakitzen badu, zenbaterako sakontasunarekin egiteko aukera eskaintzen zaio.

Hiztegiei dagokionez, hasiera batean *Tesseract* aplikazioaren barneko hiztegiak hobetzea pentsatzen bazen ere, hiztegi horiek zeuden bezala utzi dira. Horrenbestez, *Hunspell* hiztegia aplikatu zaie *Tesseract*-ek lortutako testu digitalei. Aurrez ikusi dugun bezala, hiztegien bidezko zuzenketa bi modu desberdinetara inplementatu da: Automatikoa eta semiautomatikoa.

Zuzenketa automatikoaren kasuan, hitz batzuk zuzentzea lortzen badu ere, ondo dauden beste batzuk okertzen ditu. Adibidez, “debian” hitza ingelesezko hiztegian ez dagoenez hau zuzentzen saiatzen da. Hiztegiak antz gehiena duen hitz bategatik aldatzen duenez, okerreko hitz bat ezartzen du testuan eta ondorioz asmatze-tasek behera egiten dute.

Zuzenketa semiautomatikoaren kasuan berriz, erabiltzailearen laguntzaz testua erabat zuzentzeko aukera badago ere, ez da lortzen gaizki dauden hitz guztien ebaketa egokia lortzea.

Helburuen dokumentuan, aplikazio nagusia mugikorretarako garatuko zela esan bazen ere, azkenean, ordenagailurako burutu da, hain zuzen ere, Linux sistema eragileetarako. Programari zenbait aldaketa eginez gero bestelako sistema eragileetan jartzeko modua ere egongo litzateke. Ordenagailurako soilik garatzearen arrazoi nagusia bilaketa heuristikoak egiteko behar den prozesatze gaitasuna izan da. Lehen esan bezala, zenbait bilketa sakon egiteko (ordenagailu nahiko indartsu batean[1]) lau ordu baino gehiago behar dituenez, pentsa zenbat beharko luken mugikor batek.

Etorkizuneko Lanak

Etorkizuneko lan bezala ondo egongo litzateke, hasierako plangintzan esan bezala, mugikorreko aplikazioa garatzea. Gaur egun mugikor eta tabletek sekulako indarra baitute eta hau handitzen joango baita etorkizunean. Ondorioetan aipatu bezala mugikorren prozesatzeko gaitasuna ordenagailuena baino txikiagoa izatea da lehenengo arazoa. Bestetik, kodean ere aldaketak egin beharko lirateke, garatutako programak mugikorretan funtzionatzeko.

[1] Prozesadorea: Intel Core i5-4690 3.5Ghz

Bi arazo horiek modu bakar batean konpontzea izango litzateke nire ideia. Bezero zerbitzari eredu bat inplementatuz. Horrela, proiektuan garatu den programa, zerbitzaria izango den ordenagailu indartsu batean exekutatuko litzateke. Bezeroak berriz, mugikor edo tabletak izango lirateke, hauetan liburuei argazkiak atera eta zerbitzariari bidaliko lizkioke, exekuzio astun guztia bertan exekutatuz.

Beste aukera bat, zuzenketa semiautomatikoa hobetzea izango litzateke. Gaizki dauden hitzen ebaketa zuzenak pantailaratzea lortuz gero, erabiltzailearen ahalegin gutxi baten bidez, testua ia guztiz zuzentzea lortu baitaiteke, denbora asko pasatu gabe.

Eranskinak

A. ERANSKINA

Aplikazio Nagusia Probatzen

Aplikazio nagusiaren prozesua ikusteko, testu baten irudi sinple batekin proba bat burutuko da, aplikazioak eskaintzen dituen aukera bakoitzarekin lortzen diren emaitzak bistaratuz.

3.1 irudiko eskema orokorra gogoratuz, aplikazioak eskaintzen duen lehenengo aukera, irudiarentzat aurreprozesatze propioa aurkitzea edo defektuzkoa aurreprozesatzea burutzea da. Ondoren, OCR egin eta huztegien bidez ez zuzendu, automatikoki zuzendu edo semiautomatiko zuzentzeko aukera eskaintzen du. Beraz, aurreprozesatzeari dagozkion bi aukera eta hiztegien bidezko zuzenketaren beste hiru aukera daudenez, 6 bide desberdin egongo lirateke aplikazioan.

Adibide honetan, bide horietako bi probatuko dira:

- 1.Adibidea $\Rightarrow$ Bilaketa Heuristikoa + Zuzenketa Automatikoa
- 2.Adibidea $\Rightarrow$ Defektuzko Aurreprozesaketa + Zuzenketa Semiautomatikoa

Bi adibideetan testu-irudi berbera erabiliko da, 4.3 irudia alegia.

Irudi originalari *tesseract* aplikazioaren bidez, OCR eginez gero, honako emaitza lortu da, $\%90,66$-ko asmatze-tasarekin:

```
1, La soc1edad de clases y el Estado

1. El Estado, producto del carácter irreconciliable
de las contradicciones de clase

Ocurre hoy con la doctrina de Marx lo que ha solido
ocurrir en la historia repetidas veces con las doctrinas de
los pensadores revolucionarios y de los jefes de las clases
oprimidas en su lucha por la liberación. En vida de los
grandes revolucionarios, las clases opresoras les someten
a constantes persecuciones, acogen sus doctrinas con la
rabia más salvaje, con el odio más furioso, con la campa-
ña más desenfrenada de mentiras y calumnias. Después
de su muerte, se intenta convertirlos en iconos inofensi-
VOS canonizarlos, por decirlo así, rodear sus nombres‘‘ de
una cierta aureola de gloria para <<consolar » y engañar a
188 Clases ºprimidas, castrando el contemd0 de su doctri-

umolue1onar1a, mellando su filo revolucionario envi-
_’ ww . En semejante <<arreglo» del marxismo se dan
```

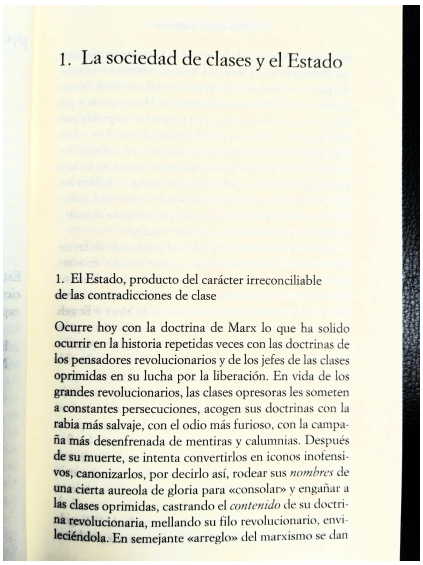
1.Adibidea

Adibide honetan, irudiaren aurreprozesatze propioa lortzeko bilaketa heuristikoa burutuko da eta ondoren modu automatikoan hiztegien bidez testua zuzenduko.

BilaketaHeuristikoa

A.2 irudia lortu da bilaketa heuristikoaren bidez.

Irudi honi *Tesseract* aplikazioa pasatuz, ondorengo testu digitala eta %94,66-ko asmtze tasa lortu dira.



1. La sociedad de clases y el Estado

1. El Estado, producto del carácter irreconciliable de las contradicciones de clase

Ocurre hoy con la doctrina de Marx lo que ha solido ocurrir en la historia repetidas veces con las doctrinas de los pensadores revolucionarios y de los jefes de las clases oprimidas en su lucha por la liberación. En vida de los grandes revolucionarios, las clases opresoras les someten a constantes persecuciones, acogen sus doctrinas con la rabia más salvaje, con el odio más furioso, con la campaña más desenfrenada de mentiras y calumnias. Después de su muerte, se intenta convertirlos en iconos inofensivos, canonizarlos, por decirlo así, rodear sus *nombres* de una cierta aureola de gloria para «consolar» y engañar a las clases oprimidas, castrando el *contenido* de su doctrina revolucionaria, mellando su filo revolucionario, envileciéndola. En semejante «arreglo» del marxismo se dan

A.1 Irudia: Bilaketa Heuristikoaren bidez lortutako aurreprozesatzea: Normalize + Normalize + Contrast + Contrast + Contrast + Contrast

```
1, La sociedad de clases y el Estado

1. El Estado, producto del carácter irreconciliable
de las contradicciones de clase

Ocurre hoy con la doctrina de Marx lo que ha solido
ocurrir en la historia repetidas veces con las doctrinas de
los pensadores revolucionarios y de los jefes de las clases
oprimidas en su lucha por la liberación. En vida de los
grandes revolucionarios, las clases opresoras les someten
a constantes persecuciones, acogen sus doctrinas con la
rabia más salvaje, con el odio más furioso, con la campa-
ña más desenfrenada de mentiras y calumnias. Después
de su muerte, se intenta convertirlos en iconos inofensi-
VOS, canºnizarlos, por decirlo así, rodear sus nombres de
una cierta aureola de gloria para <<consolar» y engañar a
las clases oprimidas, castrando el contenido de su doctri-
na revolucionaria, mellando su filo revolucionario, envi-
lºdéndola. En semejante <<arreglo» del marxismo se dan
```

Zuzenketa Automatikoa

Zuzenketa automatikoaren bidez berriz, asmatze-tasa jaitsi egin da, %93,33-ra alegia. Bi testuak konparatzen badira, arazo bat ikus daiteke. Adibidez 'Marx' izen propioa, 'Marc' izen propioagatik aldatzen da. Horrelako akatsengatik jaisten da beraz asmatze-tasa.

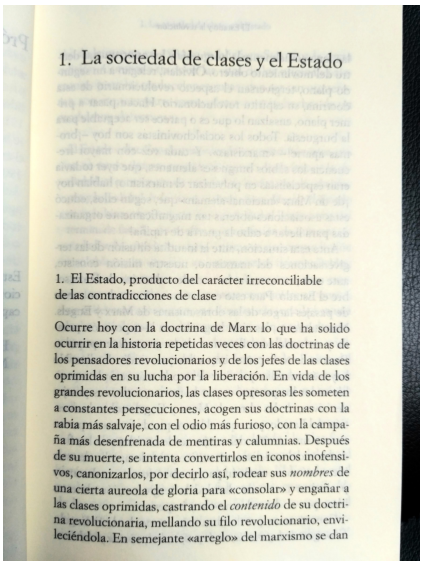
```
1, La sociedad de clases y el Estado

1. El Estado, producto del carácter irreconciliable
de las contradicciones de clase

Ocurre hoy con la doctrina de Marc lo que ha solido
ocurrir en la historia repetidas veces con las doctrinas de
los pensadores revolucionarios y de los jefes de las clases
oprimidas en su lucha por la liberación. En vida de los
grandes revolucionarios, las clases opresoras les someten
a constantes persecuciones, acogen sus doctrinas con la
rabia más salvaje, con el odio más furioso, con la campa-
ña más desenfrenada de mentiras y calumnias. Después
de su muerte, se intenta convertirlos en iconos inofensivo-
VOS, canonizarlos por decirlo así, rodear sus nombres de
una cierta aureola de gloria para desconsolar y engañar a
las clases oprimidas, castrando el contenido de su doctrina-
na revolucionaria, mellando su filo revolucionario, envió
oliéndola En semejante arreglo del marxismo se dan
```

2.Adibidea

Adibide honetan, defektuzko aurreprozesatzea burutuko da eta ondoren modu semiautomatikoan hiztegien bidez testua zuzendu.

Defektuzko Aurreprozesaketa

A.2 irudia lortu da bilaketa heuristikoaren bidez.



1. La sociedad de clases y el Estado

1. El Estado, producto del carácter irreconciliable de las contradicciones de clase

Ocurre hoy con la doctrina de Marx lo que ha solido ocurrir en la historia repetidas veces con las doctrinas de los pensadores revolucionarios y de los jefes de las clases oprimidas en su lucha por la liberación. En vida de los grandes revolucionarios, las clases opresoras les someten a constantes persecuciones, acogen sus doctrinas con la rabia más salvaje, con el odio más furioso, con la campaña más desenfrenada de mentiras y calumnias. Después de su muerte, se intenta convertirlos en iconos inofensivos, canonizarlos, por decirlo así, rodear sus *nombres* de una cierta aureola de gloria para «consolar» y engañar a las clases oprimidas, castrando el *contenido* de su doctrina revolucionaria, mellando su filo revolucionario, envileciéndola. En semejante «arreglo» del marxismo se dan

A.2 Irudia: Defektuzko Aurreprozesatzearen bidez lortutako irudia: Normalize + Gaussian-Blur + Contrast

Irudi honi *Tesseract* aplikazioa pasatuz, ondorengo testu digitala eta %88,66-ko asmtze tasa lortu dira. Hau da, irudi originalarekin baino emaitza txarragoa lortu da.

```
1, La soc1edad de clases y el Estado

1. El Estado, producto del carácter irreconciliable
de las contradicciones de clase

Ocurre hoy con la doctrina de Marx lo que ha solido
ocurrir en la historia repetidas veces con las doctrinas de
los pensadores revolucionarios y de los jefes de las clases
ºprimidas en su lucha por la liberación. En vida de los
grandes revolucionarios, las clases opresoras les someten
a constantes persecuciones, acogen sus doctrinas con la
rabia más salvaje, con el odio más furioso, con la campa-
ña más desenfrenada de mentiras y calumnias. Después
de su muerte, se intenta convertirlos en iconos inofensi-
Vº|, canonizarlos, por decirlo asi, rodear sus nom/m x de
... cierta aureola de gloria para <<consolar» y engañar a
’‘‘ Clases oprimidas, castrando el ¿()/¡tenido de su doctri-
```

```
‘‘...lucionaría, mellando su filo revolucionario, emi-

.... En semejante <<arreglo» del marxismo se dan
```

Zuzenketa Semiautomatikoa

Zuzenketa Semiautomatikoarekin, %88,66-tik %90,66-ko asmatze-tasa izatera pasatzea lortu da.

```
1, La sociedad de clases y el Estado

1. El Estado, producto del carácter irreconciliable
de las contradicciones de clase

Ocurre hoy con la doctrina de Marx lo que ha solido
ocurrir en la historia repetidas veces con las doctrinas de
los pensadores revolucionarios y de los jefes de las clases
oprimidas en su lucha por la liberación. En vida de los
grandes revolucionarios, las clases opresoras les someten
a constantes persecuciones, acogen sus doctrinas con la
rabia más salvaje, con el odio más furioso, con la campa-
ña más desenfrenada de mentiras y calumnias. Después
de su muerte, se intenta convertirlos en iconos inofensi-
Vº|, canonizarlos, por decirlo así rodear sus nom/m x de
... cierta aureola de gloria para <<consolar» y engañar a
’‘‘ Clases oprimidas, castrando el ¿()/¡tenido de su doctri-

‘‘...lucionaría, mellando su filo revolucionario, emi-

.... En semejante <<arreglo» del marxismo se dan
```

B. ERANSKINA

Proiektuaren Helburuen Dokumentua

Proiektuaren Deskribapena eta Helburuak

Proiektu honen helburua, testu historikoen irudiak izanik, modu automatiko batean, irudi horiek testu bihurtzea da. Hau burutzeko OCR [1] teknika inplementatuta daukan *tesseract* aplikazio libreaz baliatuko gara. Horrela software librearen ideologia jarraituz, jadanik inplementatuta dagoena aprobetxatuko da, eta honen gainean hobekuntzak egiten saiatu.

Helburuak gauzatzeko, proiektua azpi-helburuetan banatzea komeni da. Horrela, azpi-helburuak garrantziaren arabera sailkatu eta horren arabera joateko atazak burutzen.

Nire proiektua konputazioa espezialitatekoa denez, helburu nagusienetakoa, adimen artifizialeko teknika desberdinak probatzea da. Horrela, *tesseract* aplikazioan aldaketak gauzatuz, testu historikoei OCR egitean, asmatze-tasak igotzea lortuz edo egindako probak dokumentatuz behintzat.

Bestetik, OCR teknika hiztegiez baliatzen denez, garrantzitsua izango da hauek aztertzea eta aldatzea. Adibidez, gure testu historikoak, ez dute zertan gaur egungo hiztegiak onartzen duten moduan idatzita egon. Horrenbestez, “tesseract” aplikazioak arazoak izan ditzake testu hauek prozesatzeko garaian. Beraz, gure testu historikoentzat egokiak diren hiztegiak inportatzen baditugu, emaitzak hobetu ditzakegu.

Gainera, testu historikoen irudiak zuzenean *tesseract* aplikazioari pasatu beharrean, au-

[1] Optical character recognition

rrez irudi hauek prozesatuz, emaitzak hobetzen al diren begiratzea komeni da. Azken batean, irudian dauden letra edo hizkiekin soilik geratzea, hots, irudia garbitzea.

Helburu horiez gain ondo egongo litzateke, testu historikoei OCR egitean, bihurketa testu planoan esportatu beharrean, jatorrizko pdf-aren gainean egitea. Hau da, jatorrizko pdf-a, bilaketak egiteko aukera moduan lagatzea.

Aurreko helburuak barnebilduz, egindako guztia aplikazio batean bilduko da, proiektuari osotasun bat emanez.

Atazak

Behin helburuak definitu ditugula, proiektuan zehar burutu beharreko lana deskonposatuko dugu, lan pakete desberdinetan deskonposatuz (LDE).

LDE[2]-tik abiatuz (B.2 irudia), hona hemen proiektua arrakastaz burutzeko garatu beharreko azpi-ataza edo lan pakete bakoitzaren deskribapena edo azalpenak:

1.1.1 Plangintza: Proiekturen helburuak betetzeko jarraiutko den lan metodologia zehaztuko da.

1.1.2 Jarraipena eta Kontrola: Plangintzan definitutakoa betetzen hari garen edo ez kontrolatzeko erabiliko da. Proiektuan zehar emango da jarraipen hori eta behin amaitutakoan konparaketak egiteko plangintzan definitutakoarekin konparatzeko balioko digu.

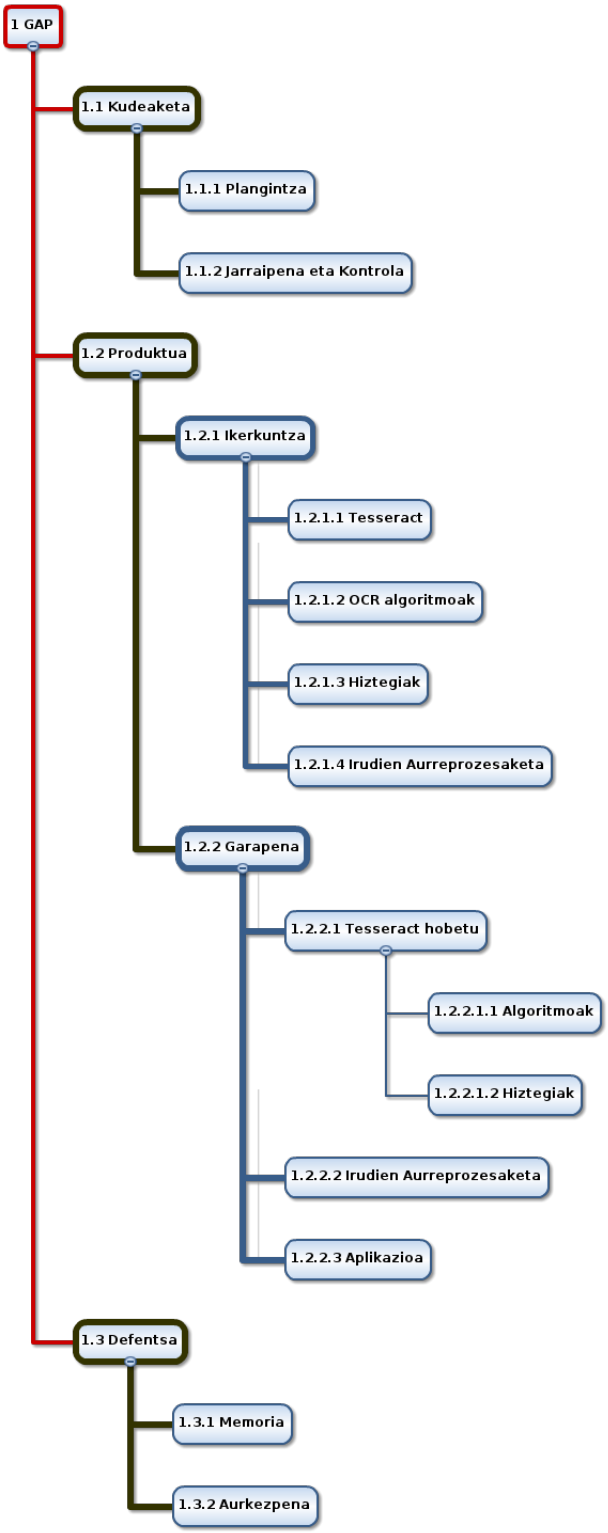
1.2.1.1 Tesseract: *tesseract* programa osoak nola funtzionatzen duen jakitea oso zaila bada ere, guri interesatzen zaizkigun atalen inplementazioa aztertu eta nola dabilen ulertu beharko da.

1.2.1.2 OCR algoritmoak: *tesseract* aplikazioak zein OCR algoritmo erabiltzen dituen aztertu eta hobekuntzak nola egin ditzazkegun ikertu.

1.2.1.3 Hiztegiak: *tesseract* aplikazioak hiztegiak nola kudeatzen dituen aztertu eta hobekuntzak nola egin ditzazkegun ikertu.

1.2.1.4 Irudien Aurreprozesaketa: Irudiei OCR egin aurretik, hauek aurreprozesatzeko zein aukera dauden ikertu.

[2]Lanaren Deskonposaketa Egitura



B.1 Irudia: LDE

1.2.2.1.1 Algoritmoak: Aurrez ikertutako algotimoak “tesseract”-en inportatu eta probak egin

1.2.2.2.2 Hiztegiak: *tesseract* aplikazioan hiztegi desberdinak inportatu eta probak egin.

1.2.2.2. Irudien Aurreprozesaketa: Aurrez ikertutakoa inplementatu (ez badago inplementatuta) eta probak egin

1.2.2.3 Aplikazioa: Egindako guztia mugikorreko aplikazio batean bildu.

1.3.1 Memoria: Proiektuan garatuko den produktuaren dokumentazioa idatzi.

1.3.2 Aurkezpena: Proiektuan garatuko dena laburbiltzen duen azalpena prestatu.

Estimazioa eta Egutegia

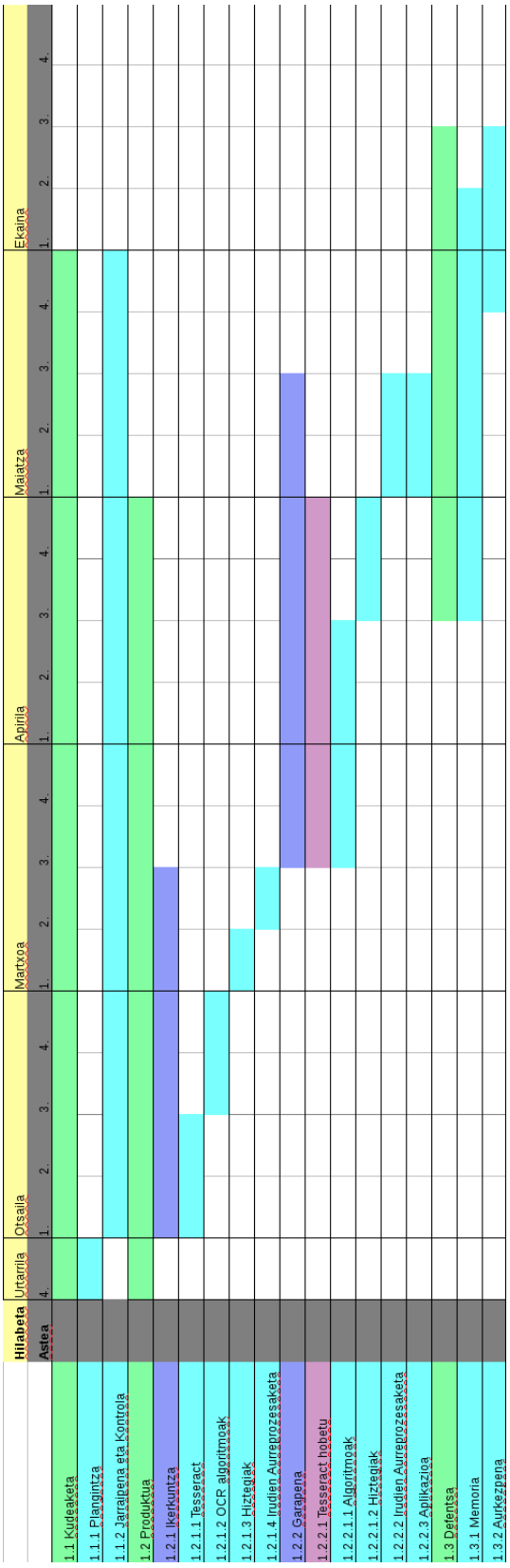
Atal honetan, proiektua burutzeko estimatutako orduak (300 ordu) zein atazetan inbertituko diren azalduko da. Bestetik, ataza horiek egutegian zehar noiz garatuko diren, Gantt diagramaren laguntzaz.

Estimazioa

Lan Paketea	Lanaren Estimazioa (Ordutan)
1.1.1 Plangintza	15
1.1.2 Jarraipena eta Kontrola	20
1.2.1.1 Tesseract	20
1.2.1.2 OCR algoritmoak	30
1.2.1.3 Hiztegiak	15
1.2.1.4 Irudien Aurreprozesaketa	15
1.2.2.1.1 Algoritmoak	40
1.2.2.2.2 Hiztegiak	20
1.2.2.2 Irudien Aurreprozesaketa	20
1.2.2.3 Aplikazioa	30
1.3.1 Memoria	60
1.3.2 Aurkezpena	15
Guztira	300

B.1 Taula: Ataza Bakoitzaren Estimazioa

Gantt Diagrama



B.2 Irudia: Gantt Diagrama

Irismena

Proiektuan ezartzen diren helburuak arrakastaz bete nahi badira, garrantzitsua da helmuga batzuk definitzea.

Proiektuaren helburu nagusia, aurreko atalean azaltzen den bezala, testu historikoei OCR egitean asmatze-tasak hobetzea da. Hori lortzeko *tesseract* aplikazio librean aldaketak egitea da asmoa. Baina horretan hasi baino lehen, "tesseract"programak nola funtzionatzen duen eta nola hobetu daitekeen ikertu beharra dago.

Behin ikerketa hori burututa, programaren kodea aldatzen eta probak egiten hasiko ginateke. Garrantzitsua izango da, egindako aldaketa bakoitza zergatik egin den justifikatzea eta aldaketa bakoitzaren ostean probak burutu eta hauen emaitzak gordetzea.

Aldaketen eta proben zati horrek irismenean eragin handia izango duelakoan nago. Hasieratik proposatutako aldaketek emaitza egokiak ematen badituzte helmuga edo helburu nagusira azkar iristea esan nahiko luke eta horrenbestez beste helburuak gauzatzeko denbora edukitzea. Hori horrela ez bada ordea, aldaketa eta proba gehiago egin beharko dira programan, eta posible da estimatutako orduetatik pasatzea. Kasu horretan, bigarren mailako helburuak proiektutik kanpo uztea ekar dezake, adibidez, mugikorrerako aplikazioa gauzatzea.

Lan Metodologia

Proiektua gauzatzeko jarraian azaltzen den metodologia erabiliko da. Honek fase desberdinak izango dituelarik.

Hasieran, ezer garatu aurretik, 'hasierako hurratsak' izeneko dokumentu ez ofizial bat idatzi nuen. Bertan, *tesseract* aplikazioa gainetik aztertu nuen eta Wikisouerce weborriak *tesseract*-ekin zuen interakzioa begiratu. Bestetik, aurreko urteerako GAP bat irakurri nuen, OCR-rekin lotura zuen bat zehazki.

Ondoren proiekturen helburuak pentsatu eta plangintza bat burutu da. Plangintzan proiektuarekin lortu nahi diren helburuak eta hauek lortzea nola espero den zehaztu da. Bertan, proiektua atazetan banatu eta ataza bakoitza noiz burutuko den eta zenbat denbora hartuko duen estimatu da besteak beste.

Behin plangintza bukatuta dagoela, ikerkuntzan zentratuko gara. Fase honetan *tesseract*

aplikazioak nola funtzionatzen duen eta zein algoritmo erabiltzen dituen ulertu beharko da. Baita hiztegiak nola kudeatzen dituen ere. Bestetik zein beste algoritmo inportatu daitekeen ere ikertuko da. *tesseract*-i irudia pasa aurretik hau aurreprozesatzeko zein aukera dauden ere begiratuko da. Mugikorreko aplikazioa gauzatzeko modurik egokiena zein den begiratzea ere ongi egongo zen, baina hori bukaerarako utziko da, ez baitago ziur helburu hori garatuko den ala ez.

Ikerkuntza amaituta, bertan ikasitakoa garatu eta probatu baharko da. Fase honetan lortutako emaitzak kontrastatu eta ondorioak atera beharko dira. Gogoratu beharra dago *tesseract* aplikazioa Googlek garatu duela azken urteotan eta oso tresna indartsua izatera iritsi dela. Horrenbestez bertan hobekuntzak egitea ez da lan erraza izango. Hobekuntzak egitea lortzen ez bada ere, egindako probak eta aldaketak ondo dokumentatzea izango da helburua.

Hau guztia egiten ari naizen bitartean, kalkulu horri batetan garatutakoa zehaztuko da. Hiru atal izango ditu: Zer garatu den, noiz garatu den eta zenbat denbora pasatu dudan garatzen.

Aipatzekoa da baita ere, astean behin gutxienez, proiektuaren segurtasun kopiak egingo direla. Batetik, hodeian, gitlab plataforman. Bertan egindako aldaketa guztiak ere gordetzen dira eta erraza da aurreko bertsioetara bueltatzea. Bestetik, disko gogor batetan ere gordeko dira segurtasun kopiak, hodeia arazoak ematen hasten bada ere.

Behin garapena bukatuta, memoria idazteari ekingo zaio. Memoria idatzita dagoelarik, hobekuntzak egiteko denbora baldin badago 300 ordu horiek asebetetzeko, bigarren mailako helburuei ekingo zaie, besteak beste. Garapenean arazorik balego eta memoria idazteko denbora justu dagoela estimatzen bada, garapena bertan behera uztearen zergatia azaldu beharko da.

Azkenik, memoria ondo errepasatu eta aurkezpena prestatzeari ekingo zaio.

Arriskuen Analisia

Proiektu honetan hauek dira aurreikusteen diren arriskua nagusiak:

- **Datuen Galera**: Arrisku hau ekiditeko, aldaketak egindako bakoitzean sarean gordeko da, gitlab plataforman. Bestetik, disko gogorrean ere gordeko dira kopiak, aldaketa garrantsitsuak egindako bakoiktzean adibidez.

- **Asmatze-tasak ez igotzea**: Egindako plangintzaren barruan, asmatze-tasak igotzea lortzen ez bada, plangintza zertxobait aldatu eta denbora gehiago eskeiniko zaio atal honi. Hala ere, memorian ondo jaso beharko da egindako guztia.

Interesatuak

Hasteko, proiektua arrakastaz amaitzeko lehen interesatua ikaslea bera izango da. Proiektuaren izenak dioen bezala, Gradu Amaierako Proiektua, arrakastaz amaitu beharra dago gradua amaitzeko.

Beste interesatuak, proiektuko zuzendariak izango lirateke: Basilio Sierra eta Iñaki Alegria. Hauen zeregina, ikaslea proiektuan zehar gidatzea eta zalantzak argitzea izango da. Basilio, "konputazio Zientziak eta Adimen Artifiziala"sailekoa izanik, AA-ko teknikietan gidatuko nau. Iñaki berriz, IXA taldeko kide izenaik, hizkuntza eta hiztegiekin lagunduko dit.

Bukatzeko, proiektua arrakastaz burutu eta emaitzak onak badira, zenbait historialarik proiektu hau beren lanean erabilgarria suertatzea ere gertatu daiteke.

C. ERANSKINA

Jarraipena eta Konrola

Atal honetan, proiektuaren plangintzan definitutakoa bete den edo ez aztertuko da. Besteak beste, garatutako proiektua eta hasieran planifikatutakoaren arteko aldea begiratuko dugu, baita proiektua egiteko denbora erreala aurretik planifikatutako estimazioarekin alderatu ere.

Aldaketak

Proiektuan hasieran planifikatu zena eta garatu denaren artean alde handia dago. Hasierako planifikazioan, ez zen uste irudien aurreprozesatzean horrenbeste zentratuko zenik eta OCR algoritmoekin lan gehiago egingo zela uste zen adibidez.

Aldaketa horiek, lan paketeetan sartu beharreko denbora erabat aztoratu dute, hurrengo atalean ikusi daitekeen bezala.

Desbideraketak

C.1 taulan ikusi daiteke, lan pakete bakoitzak izan duen desbiderapena. Plangintza aldatzearen ondorioz, *Tesseract* aplikazioko algoritmoak hobetzean, ez da batere ordurik sartu. Bestetik, irudien aurreprozesatzean gehiago zentratu denez, desbiderapena ikusi daiteke.

Bestetik, memoria idazteari ordu gutxiegi aurreikusi zitzaizkiola ikusi daiteke. Ia ordu bikoitza pasatu baita ataza horretan.

Lan Paketea	Estim.	Erreala	Desb.
1.1.1 Plangintza	15 ordu	13 ordu	-2 ordu
1.1.2 Jarraipena eta Kontrola	20 ordu	10 ordu	-10 ordu
1.2.1.1 Tesseract	20 ordu	17 ordu	-3 ordu
1.2.1.2 OCR algoritmoak	30 ordu	10 ordu	-20 ordu
1.2.1.3 Hiztegiak	15 ordu	17 ordu	+2 ordu
1.2.1.4 Irudien Aurreprozesaketa	15 ordu	20 ordu	+5 ordu
1.2.2.1.1 Algoritmoak	40 ordu	0 ordu	-40 ordu
1.2.2.2.2 Hiztegiak	20 ordu	25 ordu	+5 ordu
1.2.2.2 Irudien Aurreprozesaketa	20 ordu	52 ordu	+32 ordu
1.2.2.3 Aplikazioa	30 ordu	28 ordu	-2 ordu
1.3.1 Memoria	60 ordu	118 ordu	+58 ordu
1.3.2 Aurkezpena	15 ordu	?	
Guztira	300	310	+10 ordu

C.1 Taula: Hasierako estimazioen eta denbora errealaren desbiderapenak.

Desbiderapen orokorra begiratzen bazaio, planifikatutakoa baino 10 ordu gehiago sartu direla ikusi daiteke. Gainera, hasierako planifikazioan *Aurkezpena* lan paketea ere kontatu zen. Oraindik ataza hori burutu ez denez, ezin da jakin zenbat ordu pasatu beharko diren. Horrenbestez, desbiderapena zertxobait altuagoa izango da proiektuaren bukaeran.

Bibliografia

[Apache, 2017] Apache (2017). Apache lizentzia — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Apache_License. Atzipen Eguna: 2017-04-30.

[Borja Calvo, 2017] Borja Calvo, Josu Ceberio, U. M. (2017). *Bilaketa Heuristikoak: Teoria eta adibideak R lengoaian*. UPV/EHU.

[BSD, 2017] BSD (2017). Bsd lizentzia — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/BSD_licenses. Atzipen Eguna: 2017-04-20.

[Gaussian-Blur, 2017] Gaussian-Blur (2017). Gaussian-blur — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Gaussian_blur). Atzipen Eguna: 2017-06-12.

[GPL, 2016] GPL (2016). Gpl lizentzia — Wikipedia, the free encyclopedia. https://eu.wikipedia.org/wiki/GNU_General_Public_License. Atzipen Eguna: 2017-05-24.

[Hunspell, 2015] Hunspell (2015). Hunspell — Wikipedia, the free encyclopedia. https://eu.wikipedia.org/wiki/Hunspell. Atzipen Eguna: 2017-05-24.

[Image Magick, 2017] Image Magick (2017). Image magick — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/ImageMagick. Atzipen Eguna: 2017-04-30.

[Kontrastea, 2017] Kontrastea (2017). Kontrastea. https://www.tutorialspoint.com/dip/brightness_and_contrast.htm. Atzipen Eguna: 2017-06-10.

[Lat, 2003] Lat (2003). Local adaptive thresholding. http://homepages.inf.ed.ac.uk/rbf/HIPR2/adpthrsh.htm. Atzipen Eguna: 2017-06-12.

[Mugarza, 2016] Mugarza, A. J. (2016). Autokorrelazio espazialeko indizeetan oinarritutako ertz detekziorako metodoak.

[MySpell, 2016] MySpell (2016). Myspell — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/MySpell. Atzipen Eguna: 2017-04-25.

[Normalizazioa, 2017] Normalizazioa (2017). Normalizazioa — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Normalization_(image_processing). Atzipen Eguna: 2017-06-10.

[OCR, 2017] OCR (2017). Optical character recognition — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Optical_character_recognition. Atzipen Eguna: 2017-06-09.

[OpenCV, 2017] OpenCV (2017). Opencv — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/OpenCV. Atzipen Eguna: 2017-04-27.

[Populazioan Oinarritutako Algoritmoak, 2017] Populazioan Oinarritutako Algoritmoak (2017). Populazioan oinarritutako algoritmoak. https://egela1617.ehu.eus/pluginfile.php/833835/mod_resource/content/0/04_populaioan_oinarritutako_algoritmoak.svg#1_0. Atzipen Eguna: 2017-05-10.

[Tesseract, 2017] Tesseract (2017). Tesseract — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Tesseract_(software). Atzipen Eguna: 2017-04-30.