Attēlu līniju atpazīšana eksperimentālā sistēmā ar paralēlas apstrādes elementiem

Paulis Ķikusts
Rīga, 2015, gada rudens

Anotācija. Aprakstīts algoritmisks eksperiments, kas veikts ar nolūku pārliecināties, ka ļoti vienkārša, pat primitīva, bet būtiski paralelizēta, redzes datu primārā apstrāde ir pietiekoši spēcīga, lai tiktu detektētas svarīgas attēla ģeometriskās īpašības, proti, līnijas digitālā attēlā. Eksperiments aprobežojas tikai ar pašu līniju konstatējumu, bet pieļaujot principā patvaļīgu un pat mainīgu katras līnijas platumu, liekumu, spožuma intensitāti, attēla spožuma nevienmērību un trokšņus.

Līnijas tiek detektētas, paralēlā apstrādē atrodot lokālus līniju elementus. Apstrāde organizēta trijos viens otram sekojošos apstrādes līmeņos, katrā līmenī risinot specifisku matemātisku uzdevumu:

- līniju elementu detektēšana – pie dažādiem izmēriem analizē katra pikseļa apkārtni nolūkā noteikt, vai šajā apkārtnē ir saskatāms līnijas elements,
- līniju elementu grupēšana – noskaidro, kuri līniju elementu pāri der par speciāla grafa šķautni un atrod uzbūvētā grafa sakarīgās komponentes,
- līniju elementu ķēdēšana – atrod speciālā grafa sakarīgo komponenšu raksturīgas ģeometriski sakārtotas apakškopas.

Apstrādes līmeņu dati sakārtoti atbilstošos slāņos, un informācija pirmā līmeņa apstrādē tiek ņemta no dotā attēla dažādu izmēru pikseļu apgabaliem, tā nodrošinot dažāda izmēra detaļu detektēšanu. Katra līmeņa apstrādājošos elementus var uzskatīt par pēc vajadzības spēcīgiem procesoriem, kas katra līmeņa ietvaros darbojas maksimāli paralelizēti.

Nobeigumā doti eksperimentu rezultāti, imitējot piedāvāto paralēlo apstrādi uz tradicionāla procesora.

Atslēgas vārdi: attēlu analīze, līniju detektēšana, paralēlas apstrādes algoritmi.

Autoreferāts (galveno atziņu pārstāsts)

1 Ievads. Ar šo dokumentu autors mēģina sakārtot savas domas par izklāstāmo priekšmetu. Tas darīts, veicot attiecīgu datoreksperimentu, kuru stimulējusi novērotā nopietnā ažiotāža neironu tīklu sakarā, kas pasvītro mākslīgā intelekta algoritmu saskari ar bioloģiskas dabas struktūrām. Spilgts algoritmiskas apstrādes bioloģiskas dabas piemērs ir redze, un viena no iespaidīgākajām šis apstrādes iezīmēm ir darbības *paralēlisms*. Atšķirībā no augstāku līmeņu informācijas apstrādes smadzenēs, redzes signālu dabiskās apstrādes mehānismi ir lielā mērā iepazīti un izvirzīti ticami to principi. Galvenokārt tas attiecas uz redzes datu *primāro* apstrādi, to datu, kas vairāk vai mazāk tieši saistīti ar acu tīklenēm.

Mūsu mērķis ir pārliecināties, ka samērā primitīva, bet būtiski paralelizēta, redzes datu primārā apstrāde ir pietiekoši spēcīga, lai tiktu detektētas svarīgas attēla ģeometriskās īpašības. Mēs izklāstām savu eksperimentālu skatījumu uz konkrētu redzes datu veidu un to primāro apstrādi, proti, līniju atpazīšanu digitālā attēlā. Aprobežojamies faktiski tikai ar pašu līniju konstatējumu, bet pieļaujot principā patvaļīgu un pat mainīgu katras līnijas platumu, liekumu, spožuma intensitāti, attēla spožuma nevienmērību un trokšņus.

2 Sistēmas uzbūves un darbības principi. Galvenais princips ir ne ar ko neierobežots daudzums maksimāli vienkāršu paralēlas darbības apstrādes elementu – lozunga līmenī "vismaz pa procesoram uz attēla pikseli". Attēla struktūra tradicionāla, bet ierobežota tikai ar pelēkajiem toņiem ar spožuma vērtībām segmentā [0, 1]. Arī eksperimentu uzdevumu izvēlamies tradicionālu – par katru pikseli noskaidrot, vai tas pieder kādai viduslīnijai. Viduslīnijai piederošu pikseli raksturojam ar speciālu objektu – *līnijas elementu*, ko saīsināti saucam par *lineli*, un tam tiek piekārtota triāde (x, y, α), kas apraksta viduslīnijas pieskari šī pikseļa tiešā tuvumā: x un y ir pieskaršanās punkta koordinātas attēla plaknē, bet α ir pieskares leņķis ar x-asi. Lineli var iztēloties arī kā īsu pieskares nogriezni attiecīgā pikseļa rajonā.

Apstrāde tiek veikta trijos viens otram sekojošos apstrādes līmeņos:

- līniju elementu detektēšana,
- līniju elementu grupēšana,
- līniju elementu ķēdēšana.

Starp apstrādes līmeņiem izvietoti *datu slāņi*, kas satur atbilstošo apstrādes līmeņu ieejas un izejas informāciju:

- pirmais datu slānis – dotā attēla pikseļi,
- otrais datu slānis – detektētie līniju elementi,
- trešais datu slānis – līniju elementu grupas,
- ceturtais datu slānis – līniju elementu ķēdes.

Informācija katra līmeņa apstrādē tiek ņemta no iepriekšējā slāņa elementu kompleksiem, kā tas ir tradicionāli attiecībā uz neironu tīkliem, turklāt pirmā līmeņa apstrādē no dažādu izmēru pikseļu apgabaliem, tā nodrošinot dažāda izmēra detaļu detektēšanu, kas idejiski atbilst tam, ko varam saukt par daudzmērogu (*multiscale*) attēlu analīzi. Katra līmeņa apstrādājošos elementus uzskatīsim par pēc vajadzības spēcīgiem *procesoriem*, kas katra līmeņa ietvaros darbojas maksimāli paralelizēti.

Lineļu detektēšanas līmenis. Katrs procesors atbilst vienam attēla pikselim un ir saistīts ar t.s. receptīvo lauku, proti, dotajā pikselī centrētu kvadrātisku apgabalu, ko sauksim arī par *r*-apkārtni, kad atklāti jāuzrāda šī apgabala rādiuss *r*. Visi šie procesori vienlaicīgi un neatkarīgi cits no cita apstrādā datus no sava receptīvā lauka, pie tam, pakāpeniski mainot šī lauka izmērus. Katra procesora darbības rezultāts ir slēdziens par tā pikselim atbilstoša lineļa eksistenci, kas tiek kodēta lineļu datu slānī pie pikselim atbilstošā elementa.

Lineļu grupēšanas līmenis. Šī līmeņa procesori jāorganizē komplicētāk un mums te nav konkrēta priekšlikuma. Vispārējā ideja ir, ka grupēšana būtu jāveic divos apakšlīmeņos. Pirmais, lineļu grafa būvēšana, pārbaudot vai atsevišķs lineļu pāris der par grafa šķautni. Otrais, uzbūvētā grafa sakarīgo komponenšu atrašana. Katras komponentes procesoru kompleksu darbības rezultāts ir kopa no komponentei piederošajiem lineļiem, šo piederību kodējot grupu datu slānī.

Lineļu ķēdēšanas līmenis. Arī te procesori jāorganizē pietiekoši komplicēti, lai realizētu apstrādi, kuras būtība ir no katras grupu datu slānī kodētas lineļu grupas izdalīt raksturīgāko apakškopu, kura turklāt būtu ģeometriski sakārtota.

3 Sistēmas matemātiski-algoritmiskā bāze. Katrā apstrādes līmenī tiek risināts specifisks matemātisks uzdevums:

- lineļu detektēšanas līmenī – pie dažādiem *r* analizē katra pikseļa *r*-apkārtni nolūkā noteikt, vai šajā apkārtnē ir saskatāms linelis,
- lineļu grupēšanas līmenī – noskaidro, kuri lineļu pāri der par lineļu grafa šķautni un atrod uzbūvētā grafa sakarīgās komponentes,
- lineļu ķēdēšanas līmenī – atrod sakarīgo komponenšu raksturīgās ģeometriski sakārtotas apakškopas.

3.1) Lineļa detektors. Kā jau teikts, lineļa esamība tiek noteikta uz atbilstoša receptīvā lauka – pikseļa *r*-apkārtnes, datu pamata. Šie dati ir pelēko toņu attēla pikseļu gaismas starojuma intensitātes vērtību sadalījums apkārtnē. Lineļa detektēšanas procedūra ir heiristiska un balstās sekojošos apsvērumos:

- viena pikseļa apkārtne ietver īsu apstrādājamās attēla līnijas fragmentu,
- ja šim fragmentam atbilst linelis, tad šis fragments ir iegarenas formas ar garenasi līnijas lokālajā virzienā,
- fragmenta centrālais punkts atrodas tiešā viduslīnijas tuvumā,
- ja, apstrādājot dažādu izmēru *r*-apkārtnes, vairākās no tām ir saskatāms linelis, tad uz visu to pamata konstruējam kopēju rezultējošo lineli.

Izstiepuma kontrole. Lineli ģeometriski raksturo triāde (x, y, α), kas raksturo īso iztēlojamo taisnes nogriezni, kuru sauksim par *izstiepuma nogriezni*, bet tā centru par *izstiepuma centru*. Izstiepuma nogriežņa parametrus varam atrast, aproksimējot līnijas fragmentu izvēlētajā r-apkārtnē ar taisni, kas ir klasisks ģeometrisku datu apstrādes uzdevums un ir viegli atrisināms mazāko kvadrātu nostādnē. Tā tiek noskaidrotas divas lineļa esamībai svarīgas lietas:

- izstiepuma eksistence,
- izstiepuma centra tuvums r-apkārtnes centram.

Profilu kontrole. Ja abas nosauktās izstiepuma lietas ir spēkā, tad vēl jāpārliecinās, vai faktiskais intensitātes sadalījums r-apkārtnē nav pretrunā ar līnijas fragmenta klātbūtni:

- izstiepuma virzienā abpus intensitātes maksimuma zonai ir pazeminātas intensitātes zonas,
- intensitātes maksimuma zona ir relatīvi kompakta.

Šo divu kritēriju izpildi pārbaudām heiristiski. Aplūkojam r-apkārtnes intensitātes sadalījuma telpisko grafiku un tā projekciju uz vertikālas plaknes, kas perpendikulāra izstiepuma virzienam, un uz vertikālas plaknes, kas paralēla izstiepuma virzienam – pirmā ir intensitātes sadalījuma *šķērsprofils*, otrā *garenprofils*. Tā kā atsevišķa projicējošā stara virzienā iespējamas dažādas telpiskā grafika vērtības, tad par katra profila konkrēto vērtību faktiski ņemsim vidējo no šīm dažādajām grafika vērtībām.

Eksperimenta ietvaros pārliecināmies par sekojošo:

- šķērsprofila centrālā vērtība pietiekoši pārsniedz šķērsprofila vidējo vērtību,
- garenprofila centrālā vērtība ir relatīvi tuva garenprofila vidējai vērtībai.

Intensitātes inversija. Vispārīgā gadījumā ir iespējams, ka gan līnija var būt tumšāka par fonu, gan fons tumšāks par līniju. Mēs šo situāciju normalizējam, lineļa detektoram norādot tikai abstraktas intensitātes vērtības. Tā kā lineļa eksistence konkrētā r-apkārtnē nozīmē, ka šīs apkārtnes centrālā daļa ir intensīvāka par apkārtnes perifēriju, tad nepieciešams, lai intensitātes skaitliskās vērtības tieši centrālajā daļā būtu lielākās. Vajadzīgajā gadījumā pielietojam intensitātes inversiju $I \leftarrow 1 - I$.

Tehniski nepieciešamību pēc inversijas noskaidrojam, salīdzinot vidējo spožumu dotā pikseļa r-apkārtnē ar vidējo spožumu šī pikseļa $r/2$-apkārtnē.

3.2) Lineļu grafs. Lineļu grafa virsotnes ir paši lineļi, bet lineļu grafa šķautne ir galvenais integrējošais elements, kas iedibina līnijveida struktūru lineļu konfigurācijā un tā izsaka lineļu pāra lokālu ģeometrisko īpašību atrasties uz vienas līnijas.

Lineļu grafa šķautne. Šķautnes nosacījums ir būtisks moments šāda tipa līniju izsekošanas uzdevumos. Mēs savā eksperimentā ierobežojamies ar diviem pašiem vienkāršākajiem kritērijiem: attālumu starp lineļiem un to leņķu saskaņotību attiecībā pret lineļus savienojošo taisnes nogriezni. Mēģinājums iztikt bez liekuma novērtējuma prasa šķautnei pieļaut tikai pietiekoši tuvus lineļus. Tāpēc pirmais šķautnes nosacījums ir no augšas ierobežots maksimālais attālums starp lineļiem. Leņķu saskaņotības prasība ir formulējama analoģiski – katra lineļa leņķis ar lineļus savienojošo nogriezni arī nepārsniedz kādu maksimālo vērtību.

Lineļu grafa būvēšana. Katra lineļu pāra atbilstība lineļu grafa šķautnes kritērijam ir noskaidrojama pilnīgi neatkarīgi no citiem pāriem, tāpēc šis process ir vienkārši paralelizējams. Turklāt var apstrādāt ne pilnīgi visus pārus, bet gan tikai tos, kuru lineļi ir viens otram tuvāk par dotu attālumu.

3.3) Lineļu grupēšana. Lineļu grupēšanas būtība ir lineļu grafa sakarīgo komponenšu atrašana. Dažādām komponentēm ir dažādi izmēri, pie tam, izmēri nosakāmi divējādi – gan komponentes lineļu skaits, gan komponentes ģeometriskais lielums, ko varam izteikt ar lineļu centrālo punktu kopas diametru. Komponentes izmērs ir svarīgs parametrs, kas ļauj atfiltrēt trokšņu radītas parazītiskas lineļu grupas.

3.4) Raksturīgāko lineļu ķēdēšana. Šīs apstrādes būtība ir no katras lineļu grafa sakarīgās komponentes lineļiem izdalīt raksturīgāko apakškopu, kura turklāt būtu ģeometriski sakārtota. Dabisks pamats tādam sakārtojumam ir diametrālais ceļš komponentes lineļu centrālo punktu t.s. ģeometriskās tuvības grafā, kurā šķautnes savieno tuvu punktu pārus, papildus nodrošinot grafa sakarību.

4 Sistēmas kopējā algoritmiskā struktūra. Visu augstāk iztirzāto algoritmisko elementu kopsavilkums:

```
ievada attēlu
parallel for each ievadītā attēla pikselis
        for each r = 2, 4, 8, ...
                if vajadzīga inversija
                        invertē r-apkārtnes intensitātes vērtības
                if r-apkārtnei ir pietiekošs izstiepums
                        aprēķina potenciālā lineļa datus (x, y, α)
                        if punkts (x, y) ir pietiekoši tuvu r-apkārtnes centram
                          un r-apkārtnes intensitātes profili atbilst līnijas esamībai
                                konstruē un lokāli uzkrāj jaunu lineli
        end for
        if lokāli uzkrāts vismaz viens linelis
                no lokāli uzkrātajiem lineļiem konstruē un reģistrē rezultējošo lineli
parallel for each lineļu pāris p, kuru lineļi nav tālāk viens no otra, kā pieļaujams
        if p apmierina šķautnes kritēriju
                pievieno p būvējamā grafa šķautņu sarakstam
grupē lineļus
parallel for each lineļu grupa
        būvē lineļu grupas ģeometriskā tuvuma grafu
        heiristiski atrod ģeometriskā tuvuma grafa diametrālā ceļa tuvinājumu
        pievieno diametrālā ceļa tuvinājumu ķēdējumu sarakstam
```

5 Eksperimentu rezultātu izvērtējums. Imitējot piedāvāto paralēlo apstrādi uz tradicionāla procesora, centāmies veikt pietiekoši daudzveidīgus līniju atpazīšanas eksperimentus ar attēliem, kuros parādās samērā komplicētas līniju konfigurācijas ar atšķirīgu, pat mainīgu, katras līnijas platumu, liekumu, spožuma intensitāti. Arī vispārējais attēla spožums pieļauts nevienmērīgs un trokšņains.

Tā kā mūsu rīcībā nebija nekādu paralēlai darbībai organizējamu procesoru, ne arī instrumentu tādu organizēšanai, saprotams, ka nācās strādāt ar ierobežota izmēra attēliem un ierobežota izmēra *r*-apkārtnēm un tāpēc nevarējām sniegt ne tuvu dabiskām redzes sistēmām adekvātus attēla izmērus, pat ne tradicionālai attēlu apstrādei atbilstošus. Tomēr pielikumos sniegtie eksperimentu rezultāti pārliecina, ka samērā primitīva, bet masīva un būtiski paralelizēta, redzes datu primārā apstrāde ir pietiekoši spēcīga, lai tiktu detektētas svarīgas attēla ģeometriskās īpašības.

Bez šaubām, mūsu veiktais eksperiments skar ļoti šauru loku galveno momentu. Mēs ierobežojāmies tikai ar līniju brīvajām daļām, neskatot krustojumus, un tikai ar viduslīnijām, neskatot kontūrlīnijas. Tomēr jau demonstrētie rezultāti rosina tālākus pētījumus, pirmkārt, aptverot arī šos neskatītos līniju ģeometriskās struktūras elementus. Turklāt aktuāls kļūst tieši saistīts svarīgs tālāks jautājums – atrast adekvātu pāreju no šīs specifiskās redzes līmeņu apstrādes uz augstāku redzes vizuālo tēlu līmeņa apstrādi universālā neironu tīkla stilā, kad pievienojas arī citu sajūtu doti tēli.

1 Ievads

Ar šo dokumentu autors mēģina sakārtot savas domas par izklāstāmo priekšmetu un nepretendē nedz uz oriģinalitāti, nedz novitāti, neizslēdzot pat naivas folkloras atkārtošanu. Tomēr cerot, ka izklāstītie atzinumi neļaus kādam ieinteresētam lasītājam krist zemāk par piedāvāto, vienīgi to pārspēs, pat, ja dosies atšķirīgā virzienā.

Stimuls veiktajam pētījumam bija divējāds. No vienas puses, Latvijas Valsts pētījumu programmas ietvaros tiek veikts darbs ar nosacītu nosaukumu "Bioloģisko struktūru grafu-teorētiskais aspekts" [1]. No otras puses, vērojama nopietna ažiotāža neironu tīklu sakarā, kad tiek attīstīts t.s. *deep learning* virziens [2], pie tam, tik strauji attīstīts, ka ieinteresētam lasītājam te pašam jāseko jaunākajiem atzinumiem. Šo stimulu ietekmē tad arī radās vēlēšanās ievest zināmu kārtību autora senā interešu lokā.

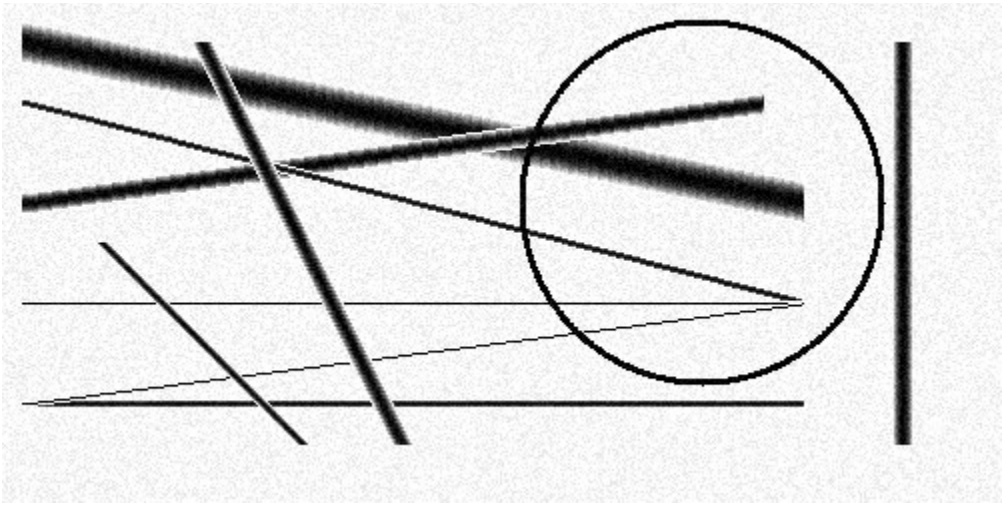
Bioloģisku struktūru algoritmiskas analīzes kontekstā iederas arī pašu algoritmu bioloģiskas dabas struktūras. Pēdējā tēze gan mums vairāk ir organizatoriska metafora, jo par bioloģiskas dabas algoritmiskiem procesiem vēl ir pārāk daudz neskaidrību, lai gan šāda rakstura pētījumi jau tiek veikti dažus desmitus gadu, sevišķi saistībā ar mākslīgajiem neironu tīkliem [3].

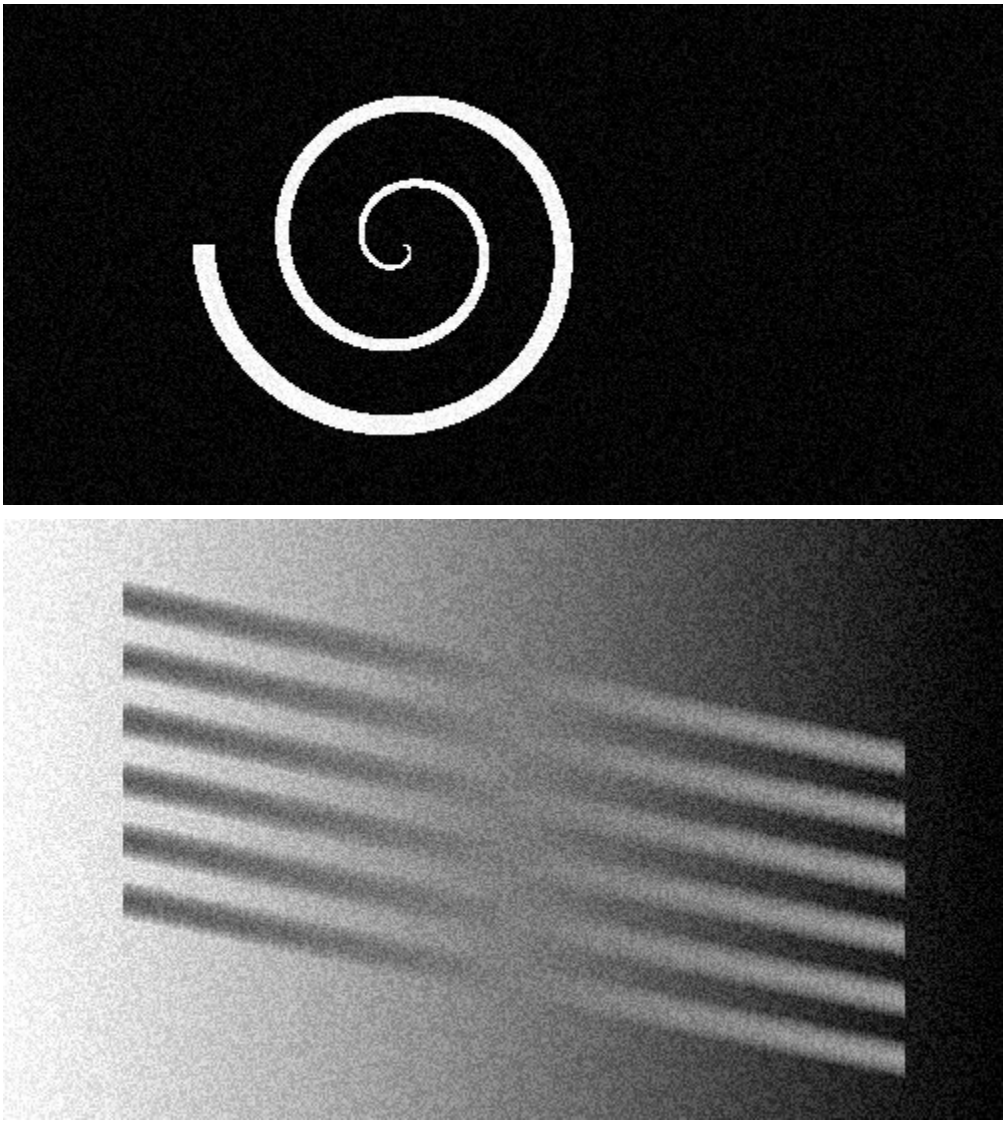
Spilgts algoritmiskas apstrādes bioloģiskas dabas piemērs ir redze, un viena no iespaidīgākajām šis apstrādes iezīmēm ir darbības *paralēlisms*. Atšķirībā no augstāku līmeņu informācijas apstrādes smadzenēs, redzes signālu dabiskās apstrādes mehānismi ir lielā mērā iepazīti un izvirzīti ticami to principi. Galvenokārt tas attiecas uz redzes datu *primāro* apstrādi, to datu, kas vairāk vai mazāk tieši saistīti ar acu tīklenēm [4], par ko tiek veikti pētījumi arī mūsu pašu Universitātē [5].

Mūsu mērķis ir pārliecināties, ka samērā primitīva, bet būtiski paralelizēta, redzes datu primārā apstrāde ir pietiekoši spēcīga, lai tiktu detektētas svarīgas attēla ģeometriskās īpašības.

Šajā dokumentā mēs izklāstām savu eksperimentālu skatījumu uz konkrētu redzes datu veidu un to primāro apstrādi, proti, līniju atpazīšanu digitālā attēlā. Visticamāk, galīgais apstrādes rezultāts nevar būt īpaši universāls, jo tam jābūt atkarīgam no specifiskā redzes analizatoram dotā uzdevuma – varbūt konkrētā gadījumā svarīgi atpazīt tikai, teiksim, vertikālas līnijas, vai varbūt riņķus, vai vienādības zīmes, vai citas svarīgas līniju kombinācijas.

Tāpēc mēs aprobežosimies faktiski tikai ar pašu līniju konstatējumu, attēlā uzrādot līniju centrālās asis, jeb *viduslīnijas*, pie tam tikai līniju *brīvajās* daļās, t.i. starp dažādu līniju krustošanās vietām. Tomēr, pieļaujot principā patvaļīgu un pat mainīgu katras līnijas platumu, liekumu, spožuma intensitāti, attēla spožuma nevienmērību un trokšņus. Zemāk redzami daži tādu attēlu ģenerēti piemēri:





2 Sistēmas uzbūves un darbības principi

Galvenais princips ir ne ar ko neierobežots daudzums maksimāli vienkāršu paralēlas darbības apstrādes elementu – lozunga līmenī "vismaz pa procesoram uz attēla pikseli".

Attēla struktūru izvēlamies tradicionālu. Uzskatot, ka pikseļi modelē tīklenes gaismas receptorus, ierobežosimies tikai ar pelēkajiem toņiem ar spožuma vērtībām segmentā [0, 1].

Arī eksperimentu uzdevumu izvēlamies tradicionālu – par katru pikseli noskaidrot, vai tas pieder kādai viduslīnijai. Viduslīnijai piederošs pikselis tiek raksturots ar vairākām līnijas īpašībām, kuras kodē speciāli objekti – *līnijas elementi*. Līnijas elementu saīsināti sauksim par *lineli*, un tam piekārtotie dati ir triāde (x, y, α), kas faktiski apraksta viduslīnijas pieskari šī pikseļa tiešā tuvumā: x un y ir pieskaršanās punkta koordinātas attēla plaknē, bet α ir pieskares leņķis ar x-asi. Lineli var iztēloties arī kā īsu pieskares nogriezni attiecīgā pikseļa rajonā.

Mūsu apstrāde tiek veikta trijos viens otram sekojošos apstrādes līmeņos:

- līniju elementu detektēšana,
- līniju elementu grupēšana,

un, lai apstrādei piešķirtu zināmu loģisku noslēgumu, veicam vēl vienu, trešā līmeņa apstrādes etapu:

- līniju elementu ķēdēšana.

Starp apstrādes līmeņiem izvietojam *datu slāņus*, kas pēc nozīmes ir *maģistrāli* un satur atbilstošo apstrādes līmeņu ieejas un izejas informāciju:

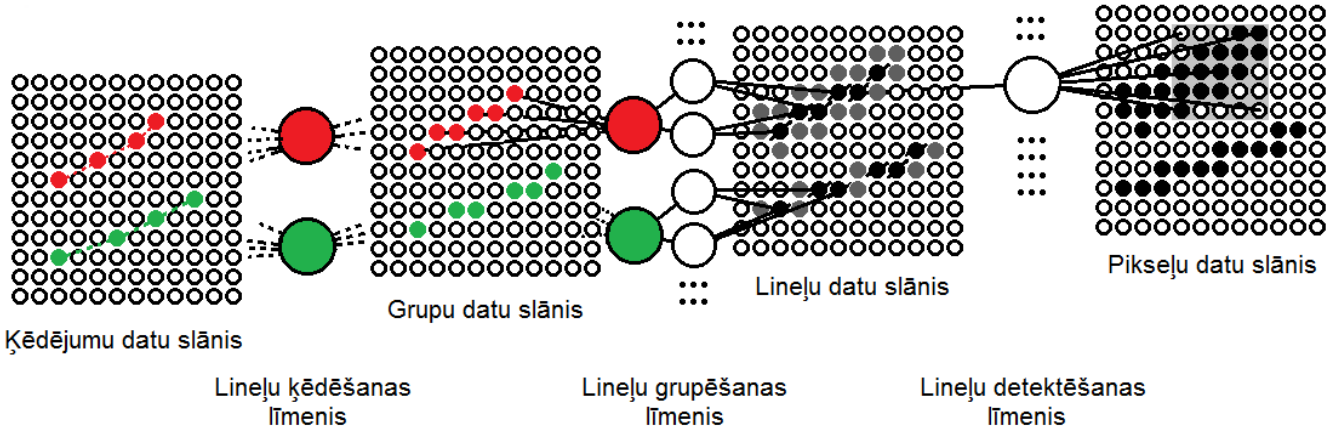
- pirmais datu slānis – dotā attēla pikseļi,
- otrais datu slānis – detektētie līniju elementi,
- trešais datu slānis – līniju elementu grupas,
- ceturtais datu slānis – līniju elementu ķēdes.

Mūsu eksperimenta ietvaros otro, trešo un ceturto datu slāni reprezentē atsevišķi plaknes punkti, kas ir vienviennozīmīgā atbilstībā ar dotā attēla pikseļiem. Protams, šiem punktiem ir piekārtoti katram slānim specifiski dati. Principā, vismaz aplūkojamā eksperimenta ietvaros, varam pat uzskatīt, ka ir tikai viens datu slānis – pikseļi, kuriem apstrādes līmeņi pakāpeniski aizpilda atbilstošos piekārtotu datu laukus. Tomēr, sadalījums slāņos šķiet bioloģiski adekvātāks. Turklāt, slāņu elementu vienviennozīmīga atbilstība attēla pikseļiem patiesībā nebūtu obligāta, jo tā nozīmē pa apstrādes līmeņiem nemainīgu vizuālo izšķiršanas spēju, kas kā prasība noteikti būtu pārspīlējums.

Informācija katra līmeņa apstrādē tiek ņemta nevis no atsevišķiem izolētiem iepriekšējā slāņa elementiem, bet gan no to kompleksiem, kā tas ir vispāratzīts princips attiecībā uz neironu tīkliem, turklāt pirmā līmeņa apstrādē no dažādu izmēru pikseļu apgabaliem, tā nodrošinot dažāda izmēra detaļu detektēšanu, kas idejiski atbilst tam, ko varam saukt par *daudzmērogu* (*multiscale*) attēlu analīzi [6]. Mēs vienīgi apstrādājošo elementu uzbūvi neierobežojam ar prasību būt tikai atsevišķiem neironiem. Katra līmeņa apstrādājošos elementus uzskatīsim par pēc vajadzības spēcīgiem *procesoriem*. Katram attiecīgā līmeņa ieejas datu slāņa elementam, vai pat to specifiskiem grupējumiem, pieļaujot, ka, jo augstāks līmenis, jo specifiskāks grupējums, piekārtosim pa vienam tādam procesoram. Iedomāsimies šos procesorus kā uzbūvētus no specifiski konstruētiem neironiem, kuri kopumā sastāda attiecīgi specializētu neironu tīklu. Nenoraidīsim arī iespēju apstrādes iekšējos datus organizēt papildus datu līmeņos, bet šī iespēja gan visticamāk prasa atdalītus maģistrālos datu slāņus, nevis tos apvienotus kopējā pikseļu slānī.

Atšķirībā no maģistrālajiem datu slāņiem, apstrādes līmeņu struktūra ir dažāda un atkarīga no apstrādē veicamā uzdevuma. Katra līmeņa ietvaros apstrādei jābūt pēc iespējas vairāk paralelizētai. Sevišķi pirmajā līmenī, kurā katrs pikselis tiek apstrādāts pilnīgi neatkarīgi un vienlaicīgi ar citiem.

Sekojošajā zīmējumā shematiski ieskicēta mūsu apstrādes līmeņu un datu slāņu struktūra un savstarpējā saistība:



Zīmējumā redzami visi četri maģistrālie datu slāņi ar ilustratīvu informāciju tajos – divām

dažāda platuma līnijām, kas pakāpeniski tiek modificētas attiecīgajos apstrādes līmeņos. Apstrādes līmeņi zīmējumā attēloti ar specifiski organizētu procesoru kopumu. Procesori simboliski apzīmēti ar lielākiem apļiem.

Apstrādes arhitektūra mūsu sistēmā ir sekojoša.

Lineļu detektēšanas līmenis. Katrs procesors atbilst vienam attēla pikselim un ir saistīts ar t.s. *receptīvo lauku* [7], ko mēs dēvēsim par pikseļa *r-apkārtni*, proti, dotajā pikselī centrētu kvadrātisku apgabalu ar malas garumu $2r + 1$ vienība – shematiskajā zīmējumā piemēra 2-apkārtne pikseļu datu slānī ietonēta pelēka. Visi šie procesori vienlaicīgi un neatkarīgi cits no cita apstrādā datus no sava receptīvā lauka, pie tam, pakāpeniski mainot šī lauka izmērus.

Katra procesora darbības rezultāts ir slēdziens par tā pikselim atbilstoša lineļa eksistenci, kas tiek kodēta lineļu datu slānī pie pikselim atbilstošā elementa. Zīmējumā šie lineļi apzīmēti ar pārsvītrotiem pikseļu simboliskajiem aplīšiem.

Lineļu grupēšanas līmenis. Šī līmeņa procesori jāorganizē komplicētāk un mums te nav konkrēta priekšlikuma. Vispārējā ideja ir, ka grupēšana būtu jāveic divos apakšlīmeņos. Pirmais, lineļu grafa būvēšana, pārbaudot vai atsevišķs lineļu pāris der par grafa šķautni – to veic speciāli pāriem piekārtoti procesori (mazākie simboliskie apļi zīmējumā). Otrais, uzbūvētā grafa sakarīgo komponenšu atrašana, kas būtu jāveic specifiskiem dinamiskas struktūras procesoru kompleksiem, kas zīmējumā apzīmēti ar lielākiem krāsainiem apļiem, simbolizējot atrastās komponentes.

Katras komponentes procesoru kompleksu darbības rezultāts ir slēdziens par komponentei piederošajiem lineļiem, šo piederību kodējot grupu datu slānī, īpaši atzīmējot pikselim/linelim atbilstošo elementu. Zīmējumā šie elementi simboliski iekrāsoti katrai komponentei savā krāsā.

Lineļu ķēdēšanas līmenis. Kā jau teikts, šis līmenis vairāk ir sistēmas darba rezultātu demonstrējošā vainagojošā fāze, nevis obligāti nepieciešama apstrādes komponente. Arī te procesori jāorganizē komplicēti, lai realizētu apstrādi, kuras būtība ir no katras grupu datu slānī kodētas lineļu grupas izdalīt raksturīgāko apakškopu, kura turklāt būtu ģeometriski sakārtota, kā tas ar pārtrauktām līnijām simboliski attēlots rezultējošajā ķēdējumu datu slānī.

Par procesoru arhitektūru varam piebilst, ka lineļu detektēšanas procesori visiem pikseļiem var darboties pēc vienas un tās pašas programmas, tādējādi realizējot SIMD (*single instruction, multiple data*) principu [8]. Tāpat tas attiecas arī uz lineļu grafa šķautņu būvēšanas procesoriem. Savukārt sakarīgo komponenšu un raksturīgāko ķēdēto apakškopu atrašanu var nākties realizēt PRAM (*parallel random-access machine*) stilā [9]. Jāapsver arī jaunākā tehniskā attīstība CUDA platformā [10].

3 Sistēmas matemātiski-algoritmiskā bāze

Katrā apstrādes līmenī tiek risināts specifisks matemātisks uzdevums:

- lineļu detektēšanas līmenī – pie dažādiem r analizē katra pikseļa r-apkārtni nolūkā noteikt, vai šajā apkārtnē ir saskatāms linelis,
- lineļu grupēšanas līmenī – noskaidro, kuri lineļu pāri der par lineļu grafa šķautni un atrod uzbūvētā grafa sakarīgās komponentes,
- lineļu ķēdēšanas līmenī – atrod sakarīgo komponenšu raksturīgās ģeometriski sakārtotas apakškopas.

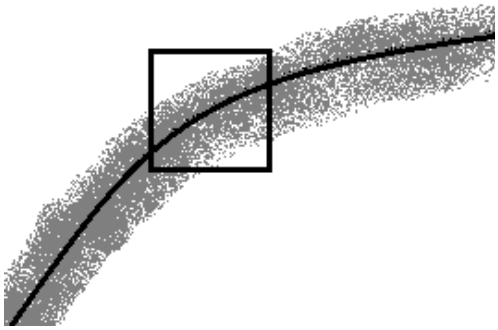
Visspecifiskākie mūsu sistēmas uzdevumi ir lineļa detektors un lineļu grafa šķautnes noteicējs, daļēji arī lineļu ķēdētājs – par tiem seko izvērsts skaidrojums. Savukārt, sakarīgās komponentes ir standarta grafu procedūra un tuvāku izklāstu neprasa.

3.1 Lineļa detektors

Kā jau teikts, lineļa esamība tiek noteikta uz atbilstoša receptīvā lauka – pikseļa *r*-apkārtnes, datu pamata. Šie dati ir pelēko toņu attēla pikseļu gaismas starojuma intensitātes vērtību sadalījums apkārtnē. Lineļa detektēšanas procedūra ir heiristiska un balstās sekojošos apsvērumos:

- viena pikseļa apkārtne ietver īsu apstrādājamās attēla līnijas fragmentu,
- ja šim fragmentam atbilst linelis, tad šis fragments ir iegarenas formas ar garenasi līnijas lokālajā virzienā,
- fragmenta centrālais punkts atrodas tiešā viduslīnijas tuvumā.

Uzskaitītās īpašības ilustrē sekojošais zīmējums:



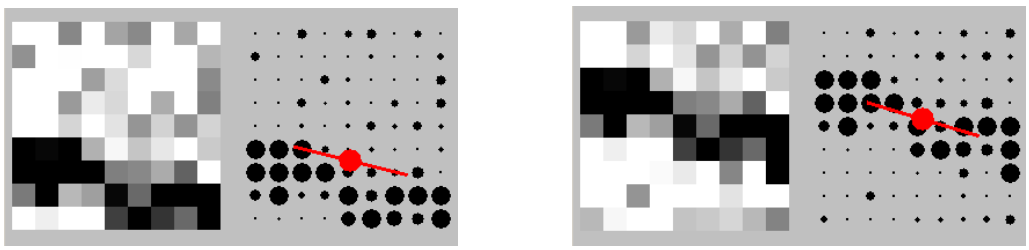
3.1.1 Izstiepuma kontrole. Lineli ģeometriski raksturo triāde (x, y, α), kur (x, y) ir linelim piekārtotā īsā taisnes nogriežņa centrs (faktiski simboliska nogriežņa), bet α – šī nogriežņa slīpums, t.i. leņķis ar x-asi. Sauksim šo nogriezni par *izstiepuma nogriezni*, bet tā centru par *izstiepuma centru*. Nosauktos izstiepuma nogriežņa parametrus varam atrast, aproksimējot līnijas fragmentu izvēlētajā *r*-apkārtnē ar taisni, kas ir klasisks ģeometrisku datu apstrādes uzdevums un ir viegli atrisināms mazāko kvadrātu nostādnē [11]. Proti, dotai plaknes punktu kopai S atrast taisni t, lai minimizētu summu

$$\sum_{P\in S} d^2(P,t),$$

kur $d(P, t)$ nozīmē punkta P attālumu līdz taisnei t. Šī minimizējošā taisne iet caur kopas S smaguma centru, kuru tad arī ņemsim par (x, y). Iespējams gadījums, kad visas taisnes, kas iet caur S smaguma centru, dod tuvas minimizējamās funkcijas vērtības – šādu gadījumu atfiltrējam ar piemērotu slieksni q_{izst} un sakām, ka izstiepuma nav un linelis neeksistē. Sliekšņa q_{izst} vērtības ņemam no segmenta [0, 1] un tas raksturo pieļaujamo minimizējamās funkcijas mazākās un lielākās vērtības attiecību.

Faktiski mēs šo uzdevumu risinām nedaudz modificētā nostādnē, kad kopa S ir svarotu *r*-apkārtnes pikseļu centru kopa, kur svari ir pikseļu intensitātes vērtības. Tas nozīmē, ka $d(P, t)$ vietā ņemam $I(P)\cdot d(P, t)$, kur $I(P)$ ir pikseļa P intensitātes vērtība.

Nākošais zīmējums ilustrē šo aprēķinu rezultātus reālos datos 4-apkārtnē:



Zīmējuma kreisajā pusē parādīts gadījums, kad līnijas fragments nav iecentrēts 4-apkārtnē, bet labajā pusē, kad ir iecentrēts. Savukārt, katrā pusē pa kreisi redzama apkārtne faktiskajās pikseļu krāsās, bet pa labi atbilstošās kopas S simboliska ilustrācija, kurā aplīšu diametri ir proporcionāli pikseļu intensitātēm. Attēloti arī izstiepuma nogriežņi ar centriem svaroto punktu kopas S smaguma centrā.

Redzam, ka lineļa esamībai ir svarīgas divas lietas:

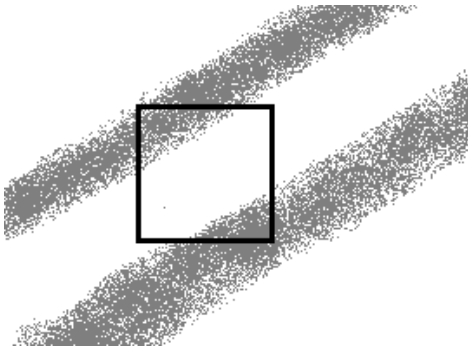
- noskaidrot, izstiepuma eksistenci (slieksnis q_{izst}),
- noskaidrot, vai izstiepuma centrs ir tuvs r-apkārtnes centram.

Izstiepuma centra tuvumu r-apkārtnes centram nosakām, salīdzinot to ar slieksni d_{centr}, kuram, kā rāda eksperimenti, ir lietderīgi būt augošai funkcijai no r.

3.1.2 Profilu kontrole. Ja abas nosauktās izstiepuma lietas ir spēkā, tad tomēr vēl jāpārliecinās, vai faktiskais intensitātes sadalījums r-apkārtnē nav pretrunā ar līnijas fragmenta klātbūtni. Tam mums ir vēl divi kritēriji:

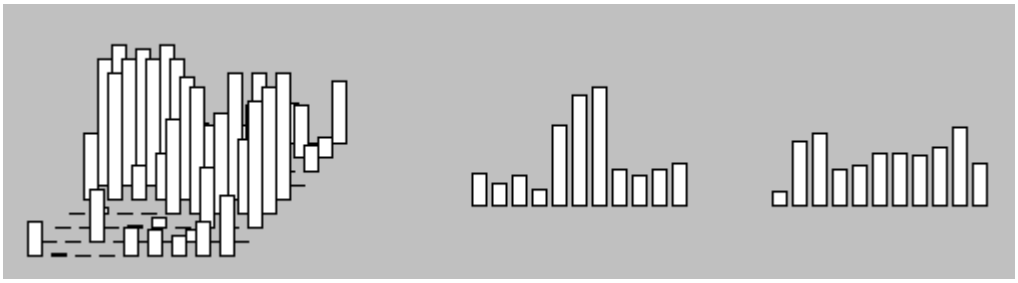
- raugoties izstiepuma virzienā, abpus intensitātes maksimuma zonai ir pazemināta intensitātes zonas,
- intensitātes maksimuma zona ir relatīvi kompakta, bez iekšējiem būtiska pazeminājuma laukumiem.

Nākošais zīmējums rāda acīmredzamu piemēru, kad r-apkārtnē ir izpildīti izstiepuma kritēriji, tomēr intensitātes sadalījums ir pilnīgi neatbilstošs vienai tumšas intensitātes līnijai caur šo apkārtni:



Pēdējo divu kritēriju izpildes pārbaudi balstīsim sekojošā heiristikā. Aplūkosim r-apkārtnes intensitātes sadalījuma telpisko grafiku un tā projekciju uz vertikālas plaknes, kas perpendikulāra izstiepuma virzienam, un uz vertikālas plaknes, kas paralēla izstiepuma virzienam – pirmā ir intensitātes sadalījuma *šķērsprofils*, otrā *garenprofils*. Tā kā atsevišķa projicējošā stara virzienā iespējamas dažādas telpiskā grafika vērtības, tad par katra profila konkrēto vērtību faktiski ņemsim vidējo no šīm dažādajām grafika vērtībām.

Sekojošajā zīmējumā redzams r-apkārtnes intensitātes sadalījuma telpiskais grafiks un abi tam atbilstošie profili, šķērsprofils un garenprofils:



Zīmējumā demonstrētie profili labi parāda gadījumu, kad ir apmierināti abi intensitātes sadalījuma kritēriji:

- šķērsprofils rāda, ka, raugoties izstiepuma virzienā, abpus intensitātes maksimuma zonai ir pazemināta intensitātes zonas,
- garenprofils rāda, ka intensitātes maksimuma zona ir relatīvi kompakta, bez iekšējiem būtiska pazeminājuma laukumiem.

Eksperimenta ietvaros izvēlamies vienkāršotas šo kritēriju analītiskās izteiksmes:

- šķērsprofila centrālā vērtība > šķērsprofila vidējā vērtība + δ_{int},
- garenprofila centrālā vērtība > garenprofila vidējā vērtība × q_{apak} un
- garenprofila centrālā vērtība < garenprofila vidējā vērtība × $q_{augš}$.

3.1.3 Intensitātes inversija. Pēdējais lineļu detektēšanu skarošais jautājums ir līnijas un fona intensitātes savstarpējā atbilstība. Iepriekšējos spriedumos tika pieņemts, ka līnijas pikseļi ir intensīvāki par fona pikseļiem, tomēr netika atklāti pateikta šīs intensitātes saistība ar faktisko spožumu, piemēros atklāti demonstrējot, ka tumšie pikseļi ir intensīvāki par gaišajiem. Vispārīgā gadījumā ir iespējams, ka gan līnija var būt tumšāka par fonu, gan fons tumšāks par līniju.

Mēs šo situāciju gribam normalizēt, lineļa detektoram norādot tikai abstraktas intensitātes vērtības, un, tā kā lineļa eksistence konkrētā *r*-apkārtnē nozīmē, ka šīs apkārtnes centrālā daļa ir intensīvāka par apkārtnes perifēriju, tad nepieciešams, lai intensitātes skaitliskās vērtības tieši centrālajā daļā būtu lielākās. Pie gaišākas līnijas uz tumšāka fona tas jau ir spēkā, bet pretējā gadījumā to nodrošina spožuma intensitātes inversija, proti, transformācija $I \leftarrow 1 - I$. Saprotams, šī operācija nedrīkst modificēt pašu doto attēlu, bet jāveic vienīgi speciālā palīgatmiņā.

Tehniski nepieciešamību pēc inversijas noskaidrosim, salīdzinot vidējo spožumu dotā pikseļa *r*-apkārtnē ar vidējo spožumu šī pikseļa *r*/2-apkārtnē:

- ja *r*-apkārtnes vidējais spožums ir lielāks par *r*/2-apkārtnes vidējo spožumu,
  tad invertē visas r-apkārtnes spožuma vērtības.

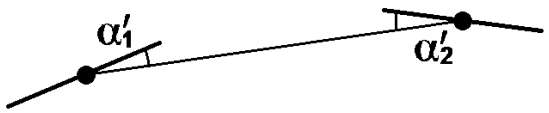
3.1.4 Lineļa detektora pseidokods. Galvenā ideja ir pie viena un tā paša dota centrālā pikseļa ar attēla koordinātām (*i*, *j*) apstrādāt dažādu izmēru *r*-apkārtnes un, ja vairākās no tām ir saskatāms linelis, tad uz visu to pamata konstruēt rezultējošo lineli, vienkāršākajā gadījumā ņemot tik vienu no tiem, teiksim, no mazākās *r*-apkārtnes Turklāt procedūras atsevišķu moduļu dažādie nosacījumi satur dažādas sliekšņa tipa vērtības, kuru lielums ir jāizvēlas pēc iespējas universāls, bet saskanīgs ar apstrādājamo attēlu raksturu gan no līniju platuma, gan attēlu trokšņu līmeņa viedokļa.

```
algorithm buildLinel(i, j)
for each r = 2, 4, 8,... rmax
        if vajadzīga inversija
                invertē r-apkārtnes intensitātes vērtības
        if r-apkārtnei ir pietiekošs izstiepums
                aprēķina potenciālā lineļa datus (x, y, α)
                if punkts (x, y) ir pietiekoši tuvu r-apkārtnes centram
                  un r-apkārtnes intensitātes profili atbilst līnijas esamībai
                        konstruē un lokāli uzkrāj jaunu lineli ar datiem (x, y, α)
end for
if nav uzkrāts neviens linelis
        return null
else
        return pirmais no lokāli uzkrātajiem lineļiem
end if
end algorithm
```

3.2 Lineļu grafs

3.2.1 Lineļu grafa šķautne. Lineļu grafa virsotnes ir paši lineļi, bet tā šķautņu noteikšana ir otrais no visspecifiskākajiem mūsu sistēmas uzdevumiem. Lineļu grafa šķautne ir galvenais integrējošais elements, kas iedibina līnijveida struktūru lineļu konfigurācijā un tā izsaka lineļu pāra lokālu ģeometrisko īpašību atrasties uz vienas līnijas.

Šķautnes nosacījums ir būtisks moments šāda tipa līniju izsekošanas uzdevumos. Tādā vai citādā formā un terminos tas sastopams tik dažādos ģeometriska rakstura datu kontekstos, ka uzlūkojams varbūt par pašu spilgtāko tādas folkloras paraugu. Nosacījumam būtu jāietver dažādi apsvērumi, kas bez paredzamā attēla trokšņu līmeņa un pieļaujamā līniju platuma, iekļauj arī, piemēram, līniju liekumu. Mēs savā eksperimentā ierobežosimies ar diviem pašiem vienkāršākajiem kritērijiem: attālumu starp lineļiem un to leņķu saskaņotību attiecībā pret lineļus savienojošo taisnes nogriezni. Iesaistītie lielumi ilustrēti sekojošajā zīmējumā:



Mēģinājums iztikt bez liekuma novērtējuma prasa šķautnei pieļaut tikai pietiekoši tuvus lineļus. Tāpēc pirmais šķautnes nosacījums ir lai maksimālais attālums starp lineļiem būtu mazāks par d_{max}. Leņķu saskaņotības prasību izteiksim tikpat vienkārši – lineļa leņķis α_i (i = 1, 2) ar lineļus savienojošo nogriezni nepārsniedz kādu maksimālo vērtību. Faktiski prasīsim, lai $|\sin(\alpha_i)| < s_{max}$.

Pie lineļu parametriem (x_1, y_1, α_1) un (x_2, y_2, α_2) abi nosacījumi ir vienkārši izsakāmi analītiski. Papildus ar β apzīmēsim lineļus savienojošā taisnes nogriežņa leņķi ar x-asi.

Attālums starp lineļiem: $d=\sqrt{\left(x_2-x_1\right)^2+\left(y_2-y_1\right)^2}$.

$c = \cos(\beta) = \frac{x_2-x_1}{d}$, $s = \sin(\beta) = \frac{y_2-y_1}{d}$.

$c_1 = \cos(\alpha_1)$, $s_1 = \sin(\alpha_1)$, $c_2 = \cos(\alpha_2)$, $s_2 = \sin(\alpha_2)$.

$|\sin(\alpha_1)| = |c \cdot s_1 - c_1 \cdot s|$, $|\sin(\alpha_2)| = |c \cdot s_2 - c_2 \cdot s|$.

Tādējādi mūsu nosacījums pierakstāms sekojoši

$$d < d_{max} \text{ un } |c \cdot s_1 - c_1 \cdot s| < s_{max} \text{ un } |c \cdot s_2 - c_2 \cdot s| < s_{max}.$$

3.2.2 Lineļu grafa būvēšana. Katra lineļu pāra atbilstība lineļu grafa šķautnes kritērijam ir noskaidrojama pilnīgi neatkarīgi no citiem pāriem, tāpēc šis process ir vienkārši paralelizējams. Turklāt var apstrādāt ne pilnīgi visus pārus, bet gan tikai tos, kuru lineļi ir viens otram tuvāk par d_{max}.

Attiecīgais pseidokods ir sekojošs.

algorithm buildLinelGraph
parallel for each lineļu pāris *p*, kuru lineļi ir viens otram tuvāk par d_{max}
    if *p* apmierina šķautnes kritēriju
        pievieno *p* būvējamā grafa šķautņu sarakstam
end algorithm

3.3 Lineļu grupēšana

Lineļu grupēšanas būtība ir lineļu grafa sakarīgo komponenšu atrašana. Kā jau teikts, par šo apstrādes līmeņa daļu mums nav konkrēta paralelizācijas priekšlikuma, tomēr iespējams balstīties uz nu jau klasiskām PRAM tipa iestrādēm, kas veiktas jau 1980-tajos gados, piemēram, [12] no mūsu personīgās kolekcijas, vai sekojot ļoti interesantajam atzinumam "In the early 1990s, there was considerable interest in parallelizing connected-component algorithms in image analysis applications, due to the bottleneck of sequentially processing each pixel. The interest to the algorithm arises again with an extensive use of CUDA." [13].

Katra atrastā komponente ir atsevišķs lineļu grafa apakšgrafs. Dažādām komponentēm ir dažādi izmēri, pie tam, izmēri nosakāmi divējādi – gan komponentes lineļu skaits n_{comp}, gan komponentes ģeometriskais lielums l_{comp}, ko varam izteikt ar lineļu centrālo punktu kopas diametru. Komponentes izmērs ir svarīgs parametrs, kas ļauj atfiltrēt trokšņu radītas parazītiskas lineļu grupas.

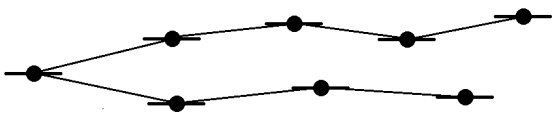
Neiedziļinoties citās detaļās, grupēšanas pseidokods ir sekojošs.

```
algorithm clusterLinels
while atrodama lineļu grafa sakarīgā komponente
        if komponentes lineļu skaits > n_comp un
          komponentes ģeometriskais diametrs > l_comp
                pievieno komponentes lineļu kopu kopējai grupu struktūrai
end algorithm
```

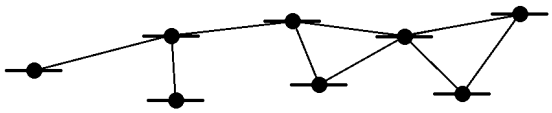
3.4 Raksturīgāko lineļu ķēdēšana

Šī apstrāde, lai arī nav galvenā mūsu eksperimenta daļa, tomēr loģiski noslēdz līniju brīvo, starpkrustojumu daļu meklētāju un ir arī interesanta pati par sevi. Turklāt tieši šie lineļu ķēdējumi ir patiesā lineļu ģeometrijas reprezentācija un var kalpot par tālāku augstāku līmeņu apstrādes ieejas informāciju, piemēram, lai ķēdētu pašus šos ķēdējumus, tā izsekojot attēla līnijas visā to garumā, un, kas ir svarīgākais, tālāk strādājot ar būtiski kompaktākiem ģeometriskiem datiem, nenoslogotiem ar pikseļu līmeņa detaļām.

Ķēdēšanas būtība ir no katras lineļu grafa sakarīgās komponentes lineļiem izdalīt raksturīgāko apakškopu, kura turklāt būtu ģeometriski sakārtota. Galvenā detektēto līniju gabalu iezīme ir šauras joslas forma. Tā kā šo joslu reprezentē lineļu grafa sakarīgā komponente, tad šīs komponentes diametrālais ceļš ir piemērots raksturīgākās apakškopas kandidāts. Tomēr šim nolūkam neder pats lineļu grafs, jo lineļu grafa šķautnes tagad ne tikai nav būtiskas, bet, kā redzams sekojošā uz reālu novērojumu pamata konstruētā piemērā, pat var šajā aspektā būt pavisam neadekvātas:



Piemērā redzams, ka lineļu grafa komponentes diametrālais ceļš dubulti nosedz ģeometrisko formu un abas tā galu virsotnes atrodas līnijas vienā galā. Viegli iedomāties arī citas radniecīgas neadekvātas lineļu grafa kombinatoriskās konfigurācijas. Skaidrs, ka, tā kā mēs gribam aprakstīt ģeometrisko formu, tad arī lineļu attieksmēm jābūt vairāk ģeometriskām – dabisks pamats tādām ir kāds no plaknes punktu t.s. *ģeometriskās tuvības grafiem* (*proximity graph*), teiksim [14], kuros šķautnes savieno tuvu punktu pārus, papildus nodrošinot grafa sakarību. Iepriekš dotajai lineļu konfigurācijai tāds ģeometriskās tuvības grafs varētu būt, piemēram, sekojošs, kurā redzams, ka diametrālais ceļš ir pilnīgi atbilstošs formai:



Diametrālā ceļa ideju šādā aspektā mēs pirmo reizi publiski izvirzījām Latvijas Universitātes mācību kursa ietvaros [15], kad piedāvājām šādi analizēt attēla pikseļu grafu. Lineļu gadījumā pikseļu tuvums var neizrādīties pietiekošs, jo lineļi ar pikseļiem tomēr nav tieši ģeometriski saistīti, tāpēc arī nepieciešams būvēt specifisku grafu.

Tomēr, tā kā lineļu ķēdēšana nav mūsu eksperimenta svarīgākais moments, tad vairāk šajā jautājumā neiedziļināsimies. Sniegsim vienīgi šī apstrādes līmeņa vispārējās struktūras pseidokodu.

parallel for each lineļu grupa *linelCluster*
    buildLinelPath(*linelCluster*)

algorithm buildLinelPath(lineļu grupa *linelCluster*)
būvē lineļu konfigurācijas ģeometriskā tuvuma grafu
heiristiski atrod ģeometriskā tuvuma grafa diametrālā ceļa tuvinājumu
pievieno diametrālā ceļa tuvinājumu ķēdējumu sarakstam
end algorithm

4 Sistēmas kopējā algoritmiskā struktūra

Visu augstāk iztirzāto algoritmisko konstrukciju kopsavilkums:

ievada attēlu
parallel for each ievadītā attēla pikselis
    for each r = 2, 4, 8, ...r_{max}
        if vajadzīga inversija
            invertē r-apkārtnes intensitātes vērtības
        if r-apkārtnei ir pietiekošs izstiepums
            aprēķina potenciālā lineļa datus (x, y, α)
            if punkts (x, y) ir pietiekoši tuvu r-apkārtnes centram
              un r-apkārtnes intensitātes profili atbilst līnijas esamībai
                konstruē un lokāli uzkrāj jaunu lineli ar datiem (x, y, α)
    end for
    if lokāli uzkrāts vismaz viens linelis
        pirmo no lokāli uzkrātajiem lineļiem pievieno lineļu struktūrai
parallel for each lineļu pāris p, kuru lineļi ir viens otram tuvāk par d_{max}
    if p apmierina šķautnes kritēriju
        pievieno p būvējamā grafa šķautņu sarakstam
grupē lineļus
parallel for each lineļu grupa
    būvē lineļu grupas ģeometriskā tuvuma grafu
    heiristiski atrod ģeometriskā tuvuma grafa diametrālā ceļa tuvinājumu
    pievieno diametrālā ceļa tuvinājumu ķēdējumu sarakstam

5 Eksperimentu rezultāti

Imitējot mūsu paralēlo apstrādi uz tradicionāla procesora, centāmies veikt pietiekoši daudzveidīgus līniju atpazīšanas eksperimentus ar attēliem, kuros parādās samērā komplicētas līniju konfigurācijas ar atšķirīgu, pat mainīgu, katras līnijas platumu, liekumu, spožuma intensitāti. Arī

vispārējais attēla spožums pieļauts nevienmērīgs un trokšņains. Sistēmas parametri mēģināti noregulēt tā, lai atpazīšana būtu saprātīgas kvalitātes gan pie daudzveidīgām līniju konfigurācijām, gan dažādiem attēla trokšņu līmeņiem.

Sistēmas parametru konkrētās vērtības visos eksperimentos tika izvēlētas sekojošas:

- izstiepuma funkcijas mazākās un lielākās vērtības attiecība mazāka par $q_{\text{izst}} = 0.75$,
- izstiepuma centra attālums no r-apkārtnes centra mazāks par $d_{\text{centr}} = r / 25$,
- šķērsprofila centrālā vērtība pārsniedz vidējo vērtība vairāk kā par $\delta_{\text{int}} = 0.05$,
- garenprofila centrālās vērtības attiecība pret vidējo vērtību lielāka par $q_{\text{apak}} = 0.5$,
- garenprofila centrālās vērtības attiecība pret vidējo vērtību mazāka par $q_{\text{augš}} = 1.5$,
- lielākās analizējamās r-apkārtnes rādiuss $r_{\max} = 16$,
- lineļu šķautnes garums ir mazāks par $d_{\max} = 10$,
- sinuss lineļa leņķim ar lineļus savienojošo nogriezni ir mazāks par $s_{\max} = 0.15$,
- lineļu komponentes elementu skaits lielāks par $n_{\text{comp}} = 2$,
- lineļu komponentes ģeometriskais diametrs lielāks par $l_{\text{comp}} = 10$.

Pielikumos sniegti eksperimentu rezultāti. Katrs eksperiments ilustrēts ar trim attēliem: oriģinālo attēlu, lineļu grupu attēlu ar trekni iezīmētām grupām gadījuma krāsās un uzbūvēto ķēdējumu attēlu. Lineļu grupu attēlos vietām redzamie lielāki vai mazāki sarkanu punktu grupējumi galvenokārt ir trokšņu izraisīti lineļu centri, kuri nav iekļauti rezultējošās grupās.

Novēroti artefakti:

- iespējamas lineļu kombinācijas, kas sapludina vienu pret otru šaurā leņķī esošas līnijas (pielikums 4, sarkanā līnija ķēdējumu attēlā),
- dēļ intensitātes inversijas tiek atrastas arī fona līnijveida struktūras (tas uzkrītoši redzams pielikumā 6 starp spirāles zariem).

Minētā līniju saplūšana jāuzlūko par apstrādes nepilnību, kura būtu jānovērš. Savukārt, lai gan līnijveida struktūras fonā arī varētu šķist kā atpazīšanas defekti, tomēr šajā attēlu analīzes līmenī tās jāatzīst par patiesām attēla īpašībām. Spilgti to demonstrē pielikums 5, kura attēls ir apzināti konstruēts tā, ka tā lielākajā daļā priekšplāns un fons ir līdzvērtīgi.

Saprotams, nācās strādāt ar ierobežota izmēra attēliem un ierobežota izmēra r-apkārtnēm. Tā, pielikuma 9 piemērs (izmērs 1763 × 690), kas, pēc būtības, ir tikai neliels fragments no pilna attēla, prasīja aptuveni 25 sekundes. Šis pielikuma 9 attēls ir 18 reizes palielināts fragments no pielikumā 8 parādītā attēla. Šāds palielinājums bija nepieciešams, lai mūsu sistēmas parametru uzstādījumi izrādītos vairāk vai mazāk atbilstoši ciparu līniju platumam. Pie tam, redzot, ka šīs līnijas pēc formas vairāk gan ir laukumi, jāatzīst, ka atpazīšanas rezultāts ir pieņemams. Protams, vienlaicīgi mēs redzam mūsu eksperimentālās sistēmas nespēju efektīvi darboties pie tik lielas izšķiršanas spējas – ja oriģinālais pielikuma 8 attēls tiktu dots šādā izšķiršanas spējā, tā izmērs būtu aptuveni 7000 × 7000, kas nepavisam nav pārmērīgs, bet pie mūsu pieejas tikai reālā daudzprocesoru sistēmā varētu cerēt uz ātru reāla laika rezultātu.

6 Noslēgums

Kā tika teikts Ievadā, mūsu tehniskais mērķis bija pārliecināties, ka samērā primitīva, bet masīva un būtiski paralelizēta, redzes datu primārā apstrāde ir pietiekoši spēcīga, lai tiktu detektētas svarīgas attēla ģeometriskās īpašības. Apstrādes vienkāršums izpaužas, pirmkārt, kā lineļa klātbūtnes noteikšana pēc acīmredzamām receptīvo lauku intensitātes sadalījuma īpašībām un, otrkārt, kā lineļu grupēšana uz elementāru ģeometriski-kombinatorisku īpašību pamata. Domājam, ka arī dabiskās redzes

sistēmās apstrāde nav īpaši komplicētāka, bet, protams, arī ne pārāk vienkārša, jo ir liecības par telpiski orientētas Gabora transformācijas klātbūtni tajās [16].

Lai gan mūsu rīcībā nebija nekādu paralēlai darbībai organizējamu procesoru, ne arī instrumentu tādu organizēšanai, un tāpēc nevarējām sniegt ne tuvu dabiskām redzes sistēmām adekvātus attēla izmērus, pat ne tradicionālai attēlu apstrādei atbilstošus, tomēr, mūsuprāt, arī te demonstrētie eksperimentu rezultāti apliecina sagaidāmo, ka masveida apstrādes paralēlisms ir spēcīgs redzes primārā līmeņa risinājums.

Bez šaubām, mūsu veiktais eksperiments skar ļoti šauru loku galveno momentu. Mēs ierobežojāmies tikai ar līniju brīvajām daļām, neskatot krustojumus, un tikai ar viduslīnijām, neskatot kontūrlīnijas. Tomēr jau demonstrētie rezultāti rosina tālākus pētījumus, pirmkārt, aptverot arī šos neskatītos līniju ģeometriskās struktūras elementus.

Var izvērtēt konkrētos tehniskos risinājumus, vai, piemēram, ir ieguvums, ka iztikām bez atklātas gludināšanas, kas netiešā veidā, protams, iekļauta gan izstiepuma aprēķinā, gan intensitātes profilu novērtēšanā. Tāpat tieši eksponenciāla *r*-apkārtņu izmēru pieaugšana visticamāk nav obligāta. Mēs to izvēlējāmies, imitētās apstrādes apjoma ekonomijai tradicionālās piramidālās apstrādes stilā. Tas būtu nopietni pārbaudāms reālā daudzprocesoru sistēmā.

Nemaz neskārām vienu mūsu sākotnējo pētījumu pielietot intensitātes funkcijas kvadrātisku regresiju izstiepuma un intensitātes profilu apvienotā novērtēšanā, kuru tomēr atzinām par pārāk neprecīzu un neatbilstošu primitivitātes koncepcijai.

Nopietns, iespējams, ka galvenais, sistēmas trūkums šobrīd ir fiksētas visu parametru vērtības neatkarīgi no apstrādājamā attēla. Viens no tradicionāliem paņēmieniem adekvātu parametru vērtību piemeklēšanai ir attēla dažādu globālu un lokālu raksturlielumu izmantojums – tādu, kā spožuma histogrammas. Tomēr iespējami arī specifiskāki pilnveidojumi, piemēram, rūpīgāka intensitātes profilu novērtēšana vai lineļu pāru saskaņotības noteikšana, ņemot vērā arī lineļu *r*-apkārtņu izmērus.

Neskatoties uz esošās eksperimentālās sistēmas trūkumiem un ierobežojumiem, noteikti ir apliecināta tēze par redzes datu lokāli primitīvas analīzes spēku masveida paralēlas primārās apstrādes vidē. Tieši saistīts svarīgs tālāks jautājums ir arī atrast adekvātu pāreju no šīs specifiskās redzes līmeņu apstrādes uz augstāku redzes vizuālo tēlu līmeņa apstrādi universālā neironu tīkla stilā, kad pievienojas arī citu sajūtu doti tēli.

Pateicības

Autors izsaka vislielāko pateicību kolēģim Mārtiņam Opmanim, kura ieinteresētība iedvesmoja šo eksperimentālo pētījumu, kā arī kolēģim un draugam Kārlim Podniekam par jautājumiem un ieteikumiem.

Atsauces

1. Valsts pētījumu programma "Nākamās paaudzes informācijas un komunikācijas tehnoloģiju (IKT) pētniecības valsts programma (NexIT)", LR Ministru kabineta rīkojums Nr.559 Rīgā 2014.gada 7.oktobrī. Projekts Nr.2 "Biometrija, biosignāli un neinvazīvās bezkontakta diagnostikas tehnoloģijas".

2. J. Schmidhuber. Deep Learning in Neural Networks: An Overview. *Neural Networks* 61: 85–117 (2015). arXiv:1404.7828 (2014) http://arxiv.org/abs/1404.7828 (skatīts 14.11.2015).

3. https://en.wikipedia.org/wiki/Artificial_neural_network (skatīts 14.11.2015).

4. https://en.wikipedia.org/wiki/Visual_cortex (skatīts 15.11.2015).

5. S. Fomins. KRĀSU UN FORMAS NOZĪME ATTĒLU ATPAZĪŠANĀ, Promocijas darbs doktora zinātniskā grāda iegūšanai fizikā, LU Rīga, 2010, https://dspace.lu.lv/dspace/handle/7/5067 (skatīts 14.11.2015).

6. https://en.wikipedia.org/wiki/Pyramid_%28image_processing%29 (skatīts 14.11.2015).

7. https://en.wikipedia.org/wiki/Receptive_field (skatīts 14.11.2015).

8. https://en.wikipedia.org/wiki/SIMD (skatīts 14.11.2015).

9. https://en.wikipedia.org/wiki/Parallel_random-access_machine (skatīts 14.11.2015).

10. Parallel Programming and Computing Platform CUDA, https://en.wikipedia.org/wiki/CUDA (skatīts 14.11.2015).

11. http://mathworld.wolfram.com/LeastSquaresFitting.html (skatīts 14.11.2015).

12. Peter H. Hochschild, Ernst W. Mayr, Alan R. Siegel. Techniques for Solving Graph Problems in Parallel Environments. *FOCS* 1983: 351-359.

13. https://en.wikipedia.org/wiki/Connected-component_labeling (skatīts 15.11.2015).

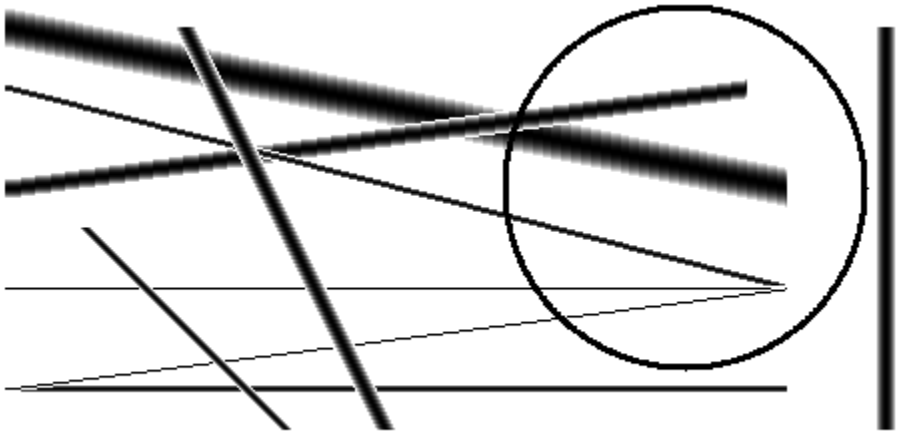
14. https://en.wikipedia.org/wiki/Relative_neighborhood_graph (skatīts 15.11.2015).

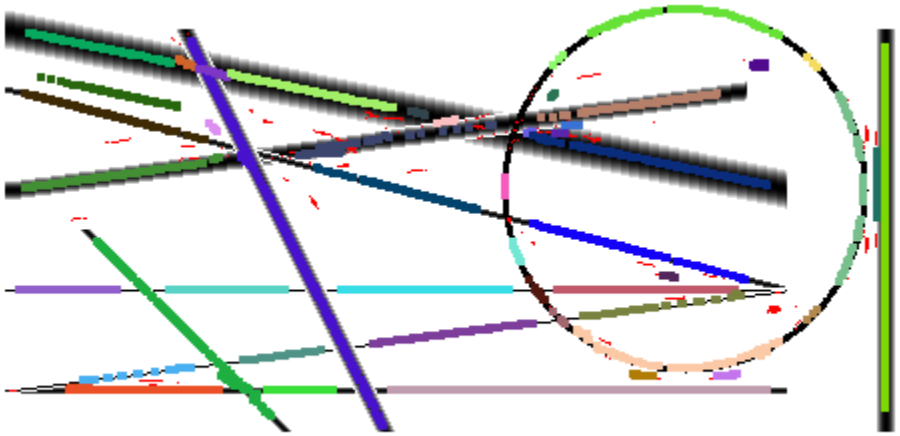
15. LU e-studijas, Datorikas fakultātes kurss DatZ3073: Datoru grafikas un attēlu apstrādes pamati. Temats "Vienkāršāko iezīmju (taišņu, riņķu, stūru) atpazīšana", http://estudijas.lu.lv/course/view.php?id=1345 (skatīts 15.11.2015).

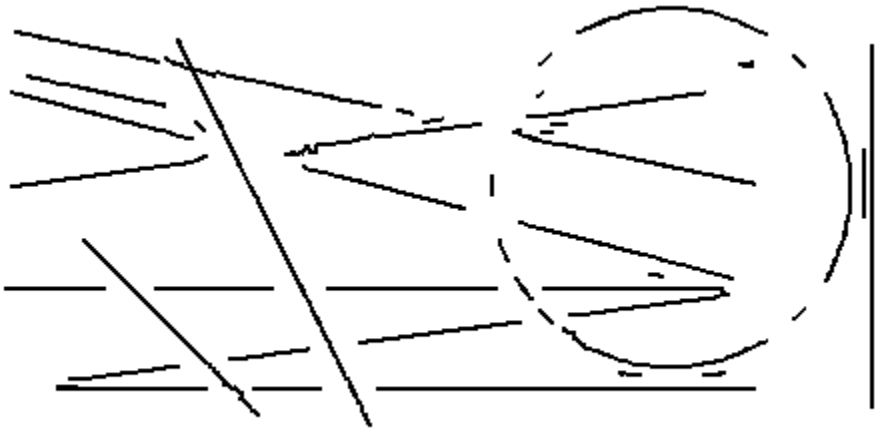
16. https://en.wikipedia.org/wiki/Gabor_filter (skatīts 15.11.2015).

Pielikums 1

Ģenerētu līniju konfigurācija, trokšņu līmenis ±0%

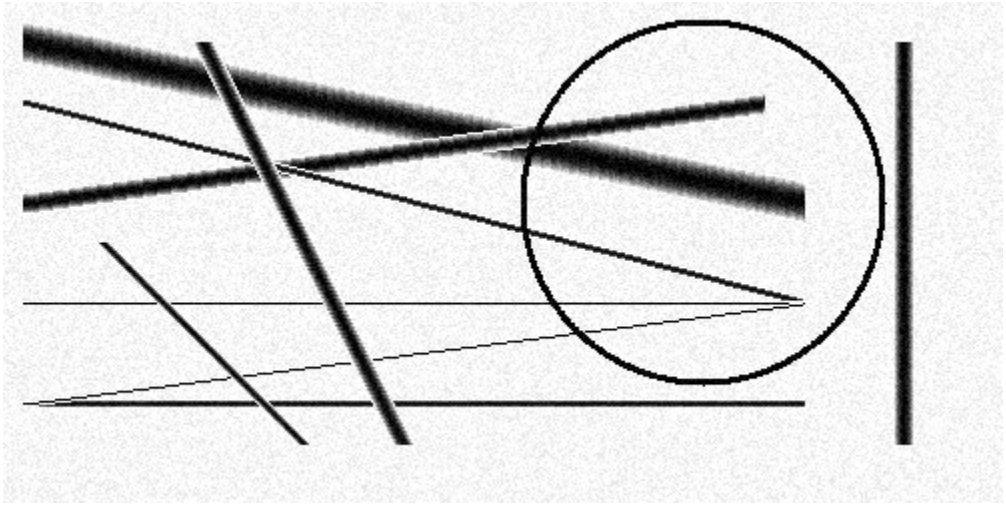


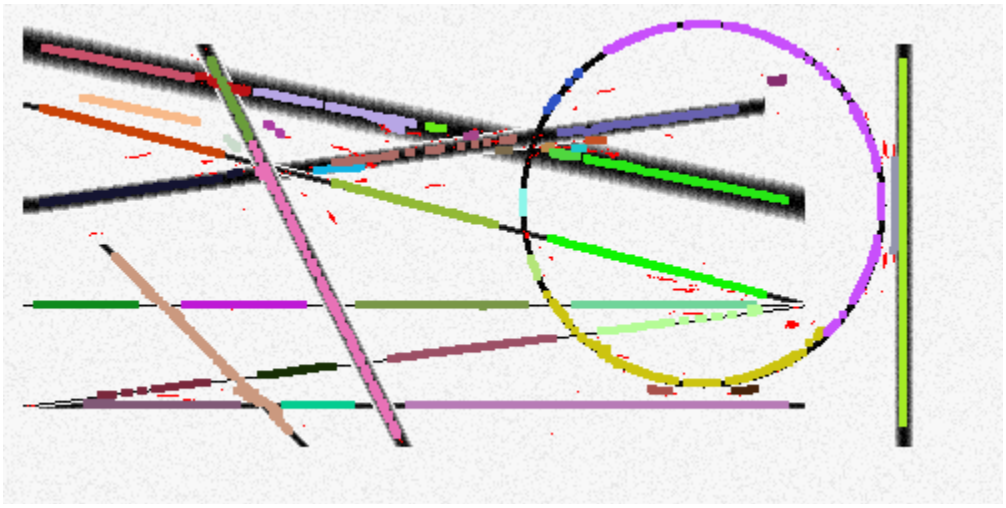


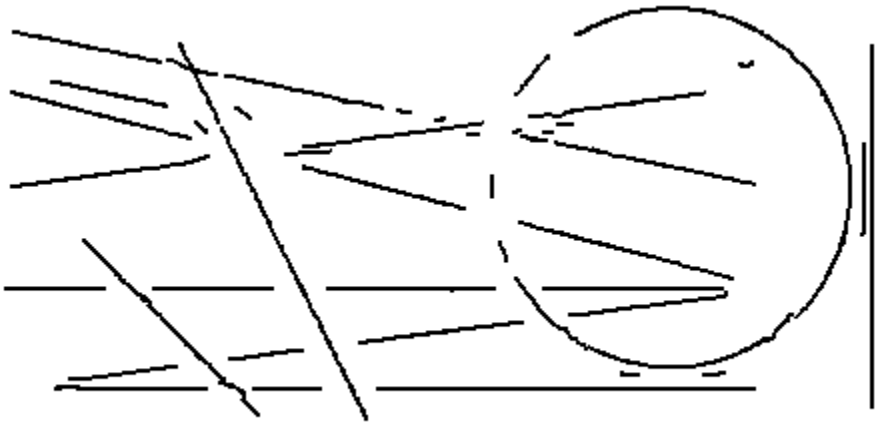


Pielikums 2

Ģenerētu līniju konfigurācija, trokšņu līmenis ±10%





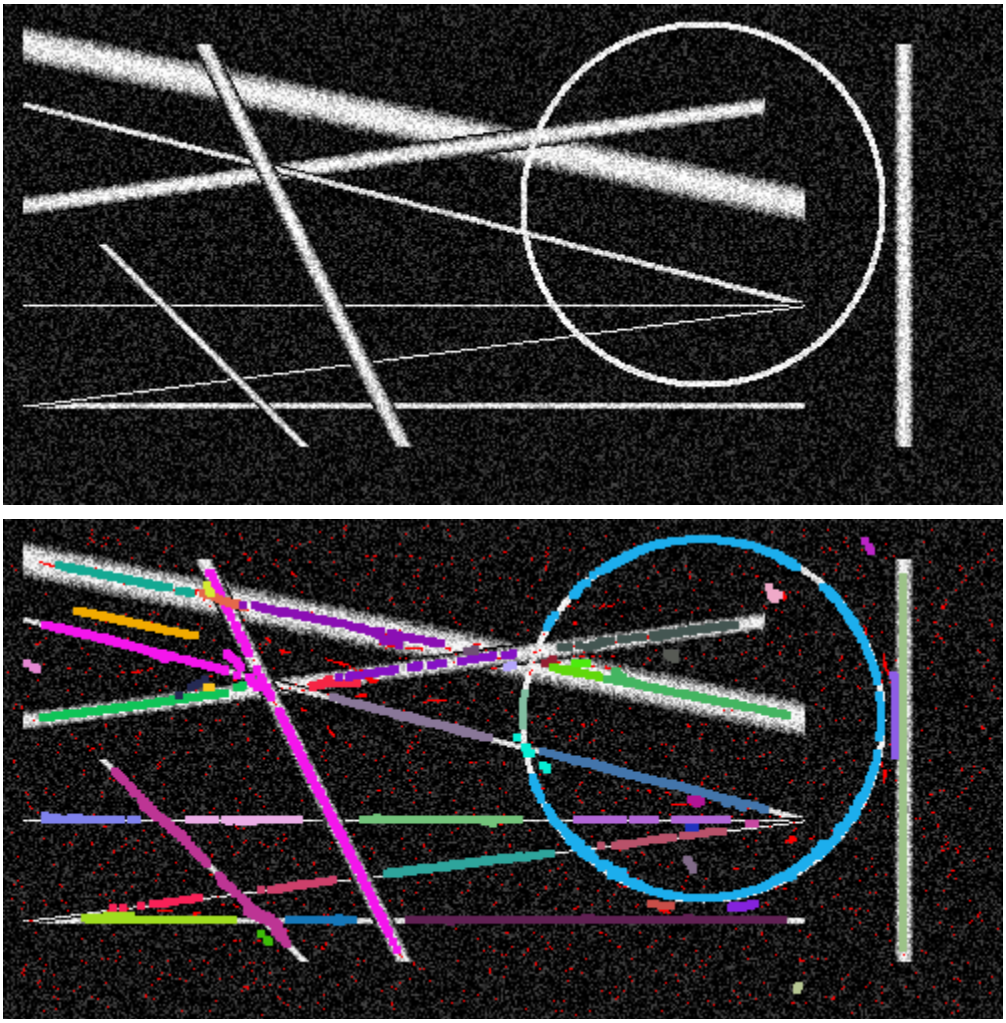


Pielikums 3

Ģenerētu līniju konfigurācija, trokšņu līmenis ±50%

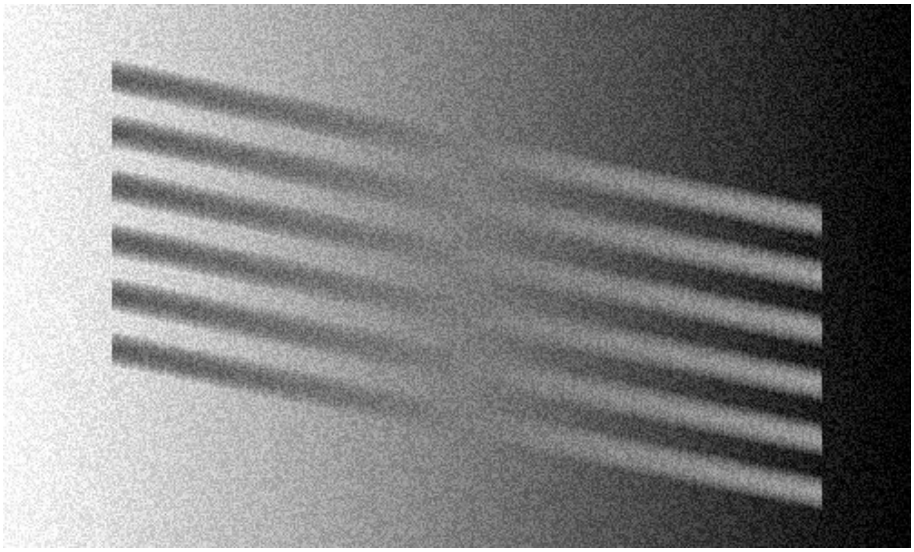
Pielikums 4

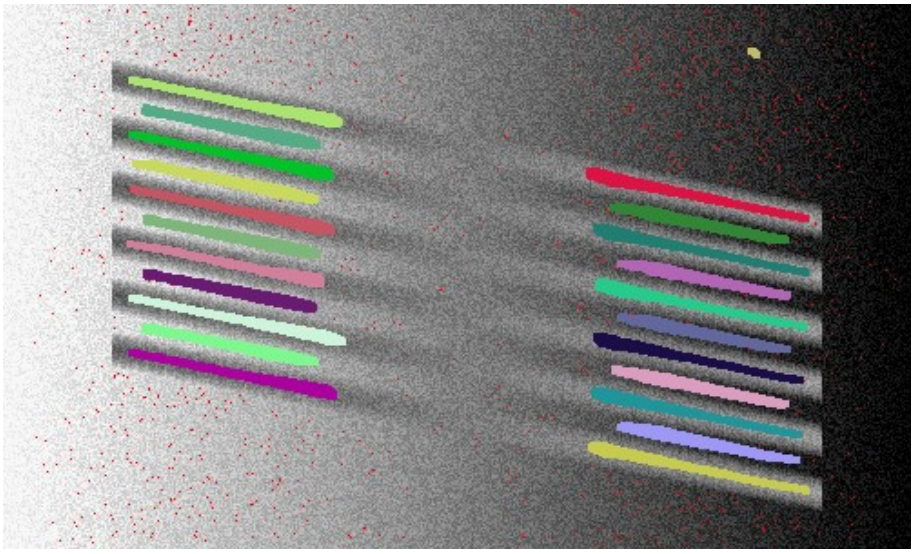
Ģenerētu līniju konfigurācija, trokšņu līmenis ±30%

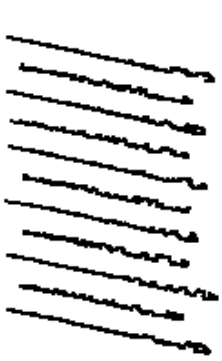


Pielikums 5

Ģenerētu līniju konfigurācija, trokšņu līmenis ±10%

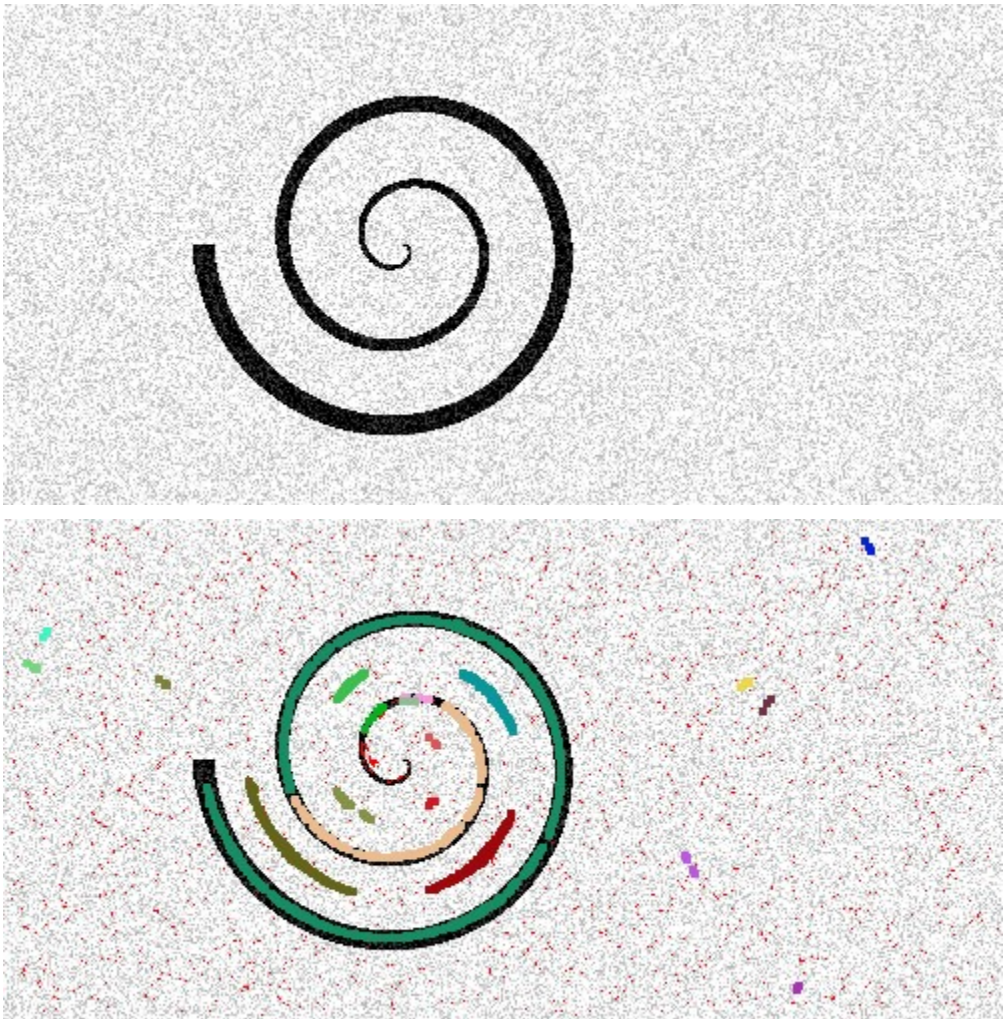






Pielikums 6

Ģenerēta spirāliska līnija, trokšņu līmenis ±30%

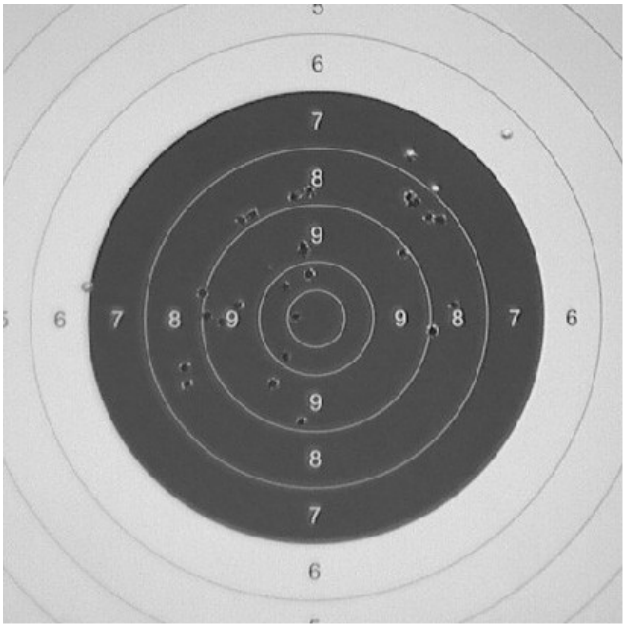


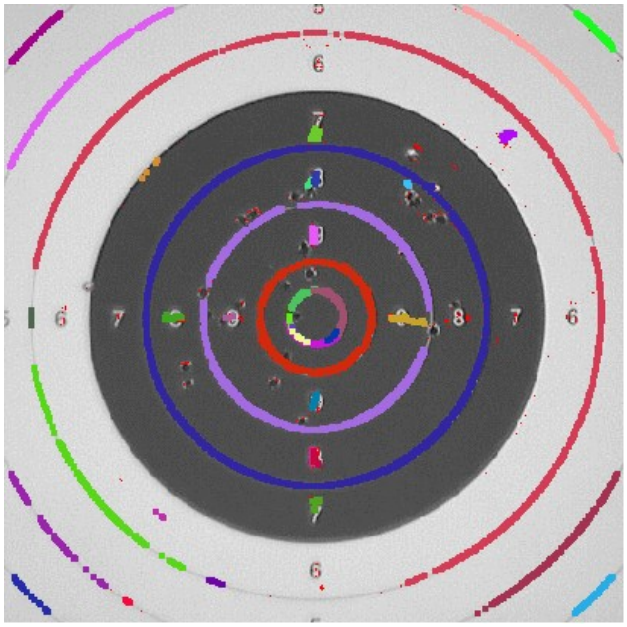
Pielikums 7

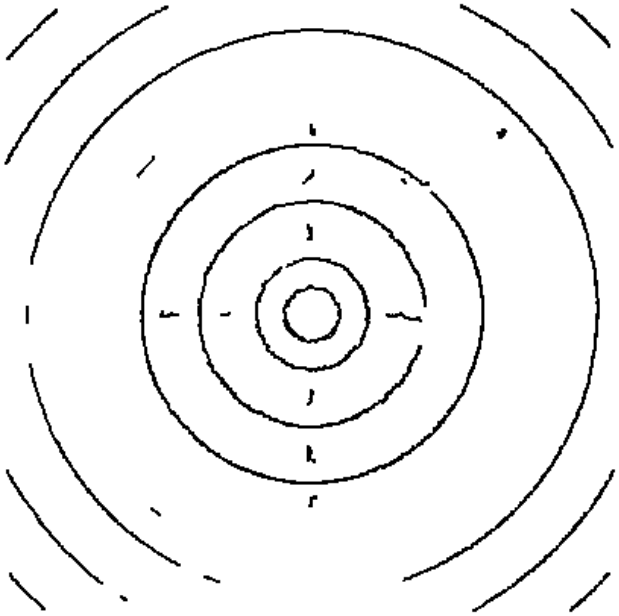
Fotogrāfija ar dabisku trokšņu līmeni

Pielikums 8

Fotogrāfija ar dabisku trokšņu līmeni







Pielikums 9

Palielināts iepriekšējās fotogrāfijas fragments