IEGULDĪJUMS TAVĀ NĀKOTNĒ

EIROPAS SAVIENĪBA

Eiropas Reģionālās attīstības fonds
Prioritāte: 2.1. Zinātne un inovācijas
Pasākums: 2.1.1.1. Zinātne, pētniecība un attīstība
Aktivitāte: 2.1.1.1. Atbalsts zinātnei un pētniecībai
Projekta nosaukums: „Multimodālas biometrijas tehnoloģija drošai un ērtai personu autentifikācijai" (BiTe)
Līguma noslēgšanas datums: 10.12.2010.g.
Projekta sākuma datums: 01.01.2011.g.
Projekta beigu datums: 31.12.2013.g.
Vienošanās Nr.2010/0285/2DP/2.1.1.1.0/10/APIA/VIAA/098
Eiropas Reģionālās attīstības fonda finansējuma saņēmējs: Elektronikas un datorzinātņu institūts (EDI)

ZINĀTNISKĀ PĒTĪJUMA PROGRESU APLIECINOŠA DOKUMENTĀCIJA

Pārskata numurs Nr.7. par periodu no 01.11.2012.g. līdz 28.02.2013.g.

Projekta zinātniskais vadītājs: Modris Greitāns, Dr.sc.comp., vad. pētnieks

Pētījuma projekta izpildītāju saraksts: Mihails Broitmans, Dr.sc.comp., vad. pētnieks
Rihards Fuksis, pētnieks
Mihails Pudžs, pētnieks
Rinalds Ruskuls, asistents
Oļegs Ņikišins, asistents
Zanda Seržāne, asistente
Artūrs Kadiķis, programmēšanas inženieris
Dāvis Barkāns, elektronikas inženieris
Teodors Eglītis, elektronikas inženieris
Dainis Backāns, informātikas/elektronikas tehniķis

ELEKTRONIKAS UN DATORZINĀTŅU INSTITŪTS
Rīga, Latvija
2013. gads

Satura rādītājs

Anotācija

BiTe ir ERAF līdzfinansēts projekts zinātnei un pētniecībai. Projekta mērķis ir droša, ērta un plaši pielietojama personu identifikācijas risinājuma izveide. Projekts ietver sevī sekojošas aktivitātes: Lietišķos pētījumus biometrijas parametru ieguvē, apstrādes metožu implementēšanā ierobežotu resursu sistēmās, atsevišķu sistēmas komponenšu un bloku izstrādē; Eksperimentālos pētījumus tehnoloģiju demonstratora izstrādē; Rūpniecisko tiesību aizsargāšana, patenta pieteikums; Projekta publicitāte. Šajā dokumentā dots pārskats par projekta septītajā periodā (01.11.2012.-28.02.2013) veiktajiem pētniecības darbiem un šobrīd sasniegtiem rezultātiem.

Projektu atbalsta Eiropas Reģionālās attīstības fonds, līguma Nr. 2010/0285/2DP/2.1.1.1.0/10/APIA/VIAA/098

Ievads

Identitātes nozagšana ir viens no modernajiem noziegumiem, kas „zaglim" dod iespēju radīt personām finansiālus zaudējumus. Katru gadu identitātes nozagšana pasaulē rada aptuveni 221 miljardu dolāru zaudējumus. Lielu interesi par biometrijas izmantošanu personu identifikācijā izrāda banku sektors. Bankas atzīst, ka šādas sistēmas ieviešana dotu iespēju aizstāt (vai kombinēt) uz parakstu vai PIN kodu ievadi balstītu norēķinu karšu autorizāciju ar kartes īpašnieka biometrisku atpazīšanu. Radot ērtu, lētu un drošu personu identifikācijas risinājumu, ir plašas iespējas to ieviest ikdienas dzīvē – bankās, tirdzniecības vietās, objektu piekļuves un lietošanas kontrolē (ieskaitot autotransportu) u.c.

BiTe pētniecības grupas darbs ir saistīts ar augstāk minēto problēmu risinājumu. Šajā projekta pārskata posmā ir veikti darbi un sasniegti rezultāti sekojošos pētījumos:

- Sejas un plaukstas attēlu iegūšanas metožu izpēte;
- Biometrisko datu apstrādes algoritmu darbības izvērtēšana;
- Biometrisko datu iegūšanas algoritmu paralelizācija un implementēšana programmējamos loģiskos masīvos;
- Biometrisko datu kriptēšanas, glabāšanas un lietojama apakšsistēmas izveide;
- Algoritmu implementēšana eksperimentālajā maketā.

Rezultātu kopsavilkums

Projekta septītajā periodā (01.11.2012.-28.02.2013.) veikto pētniecības uzdevumu un eksperimentālo rezultātu kopsavilkums ir sekojošs:

- Sejas un plaukstas attēlu iegūšanas metožu izpēte – iepriekšējos pārskata periodos tika apskatītas plaukstas specifisko pazīmju izdalīšanas iespējas; izpētīti plaukstas ģeometrijas izmēri, kas minimāli atkarīgi no plaukstas stāvokļa; turpināts plaukstas ģeometrijas algoritmu apraksts MATLAB vidē; analizēta mērķa principa metodika; izpētīts sejas detektēšanas algoritms, darbam daudzkodolu procesorā un veikta

vienlaicīga oriģinālu un filtrētu bilžu iegūšana. Šajā pārskata periodā tika noteikta lielākā iespējamā riņķa ievilkšana plaukstas kontūrā. Izmantojot Voronoi šūnas.

- Biometrisko datu apstrādes algoritmu darbības izvērtēšana – Iepriekšējos periodos realizēta 1.versija 2D sejas atpazīšanas algoritmam, kas implementēts iegultā sistēmā; veidots automātisks sejas atpazīšanas algoritms TMS320C6416; pievienoti galvenie elementi kameras modulim; izveidotas programmas attēla vizualizācijai un kvalitātes novērtēšanai; analizētas sejas atpazīšanas algoritmu implementācijas iespējas un apskatīti iespējamie risinājumi sejas atpazīšanas izstrādei iegultās sistēmās. Septītajā pārskata periodā uzsākta iegultas sistēmas izstrāde uz Raspberry Pi, veikta sejas atpazīšanas algoritmu optimizācija un izpētītas Hāra kaskādes.
- Biometrisko datu iegūšanas algoritmu paralelizācija un implementēšana programmējamos loģiskos masīvos – iepriekš tika izveidota uz FPGA bāzēta attēlu ieguves sistēma; izstrādāta iespiedplate sejas atpazīšanas algoritmu testēšanai; projektēts 2D filtrs; veikta attēla filtrēšana reālā laikā; izveidota jauna sistēmas arhitektūra; izpildīta jaunā Aptina MT9V024 attēlu sensora montāža; apskatīta datora saskarne ar prototipu; izveidota attēlu filtrācija FPGA platformā; apskatīti varianti kā samazināt dubulto leņķi un uzsākta LCD kontroliera izveide uz FPGA platformas. Šajā periodā tika turpināta plaukstas ģeometrijas algoritmu izstrāde un uzsākta to implementācija FPGA.
- Biometrisko datu kriptēšanas, glabāšanas un lietojuma apakšsistēmas izveide – iepriekšējos pārskata periodos tika izanalizēti algoritmi *biohash* funkcijas skaitļošanai; izveidots algoritms, kas iegūst vidēji tādu pašu EER kā salīdzinot nešifrētus datus; izstrādāta programmatūra, kas salīdzina nešifrētus un šifrētus datus uz Java viedkartes; izstrādāts protokols, kas ļauj izveidot drošu sakaru kanālu starp viedkarti un autentifikācijas iekārtu; sastādīts ROI noteikšanas algoritms; apskatīts Rīda–Solomona kļūdu koriģējošs algoritms; izveidota viedkartes programma, kurā ir iespējams ierakstīt cilvēka datus. Septītajā periodā uzlabots BioHash algoritms, apskatīta un pielietota vektoru rotācija, kas implementēta FPGA.
- Atsevišķu sistēmas komponenšu un bloku izstrāde un izpēte – iepriekšējos periodos tika veidotas funkcionālo bloku elektriskās principiālās shēmas *Altium Designer* vidē; izveidota principiālā viedkaršu komunikācijas maketa shēma; uzsākta kameras moduļa iespiedplates montāža; tika programmēti Aptina MT9V024 attēlu sensora reģistri; projektēts biometrijas sistēmas ietvars; izveidota iespiedplate viedkaršu saskarnes nodrošināšanai ar FPGA izstrādes rīka; programmēts mikrokontrolieris MSP430; realizēti LBP, LDP algoritmi MATLAB vidē un veikta fāzes korelācijas algoritma izveide plaukstas biometrijas datu pārbaudei.
- Multimodālas biometrijas tehnoloģijas koncepta definēšana – iepriekšējos periodos tika aprakstīts prototipa funkcionālo bloku darbības principi un savstarpējā sadarbība.
- Tehnoloģijas demonstratora eksperimentālā maketa montēšana – piektajā pārskata periodā tika veikta biometrijas prototipa 1.versijas montāža un visi sistēmas bloki tika

ievietoti organiskā stikla ietvarā, veidojot vienotu prototipēšanas sistēmu. Sestajā pārskata periodā tika uzņemta datubāze un apstrādāti attēli, izmantojot esošo prototipu.

- Algoritmu implementēšana eksperimentālajā maketā – iepriekšējā pārskata periodā veiktas CMF modifikācijas; nomainīts maskas izmērs no 9x9 uz 15 x15; izmainīta pati maska; gūts priekšstats par VHDL parametrizējamām ķēdēm un uzsākta Fast NH-CMF2 realizācija. Septītajā pārskata periodā tika veikta horizontālā, vertikālā un 2D parametrizējāmā filtra implementācija FPGA.

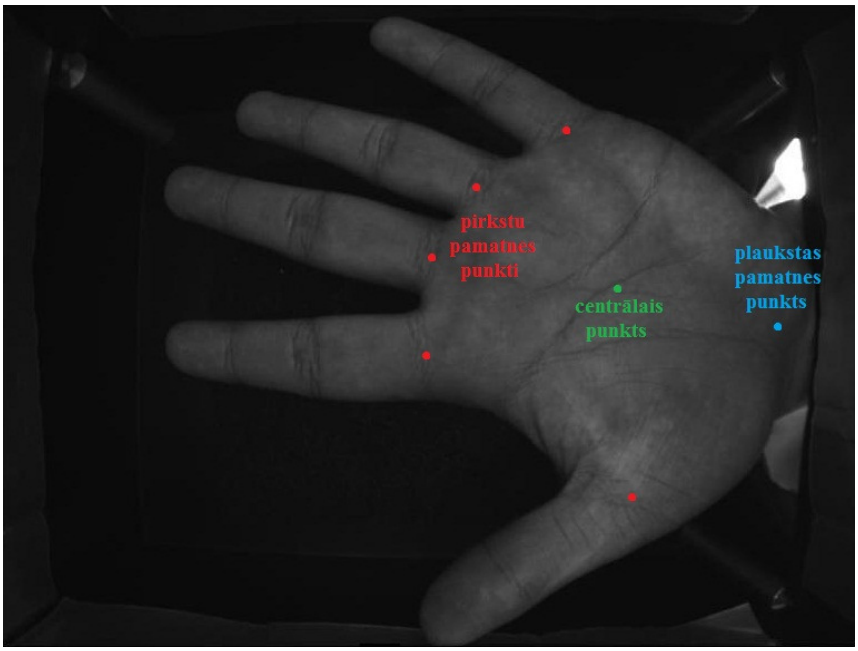
Turpmāk dokumentā ir detalizēti aprakstītas katras uzdevumu grupas pētniecības darbības un rezultāti.

Aktivitāte: Lietišķie pētījumi

Sejas un plaukstas attēlu iegūšanas metožu izpēte

Lielākā iespējamā riņķa ievilkšana plaukstas kontūrā.

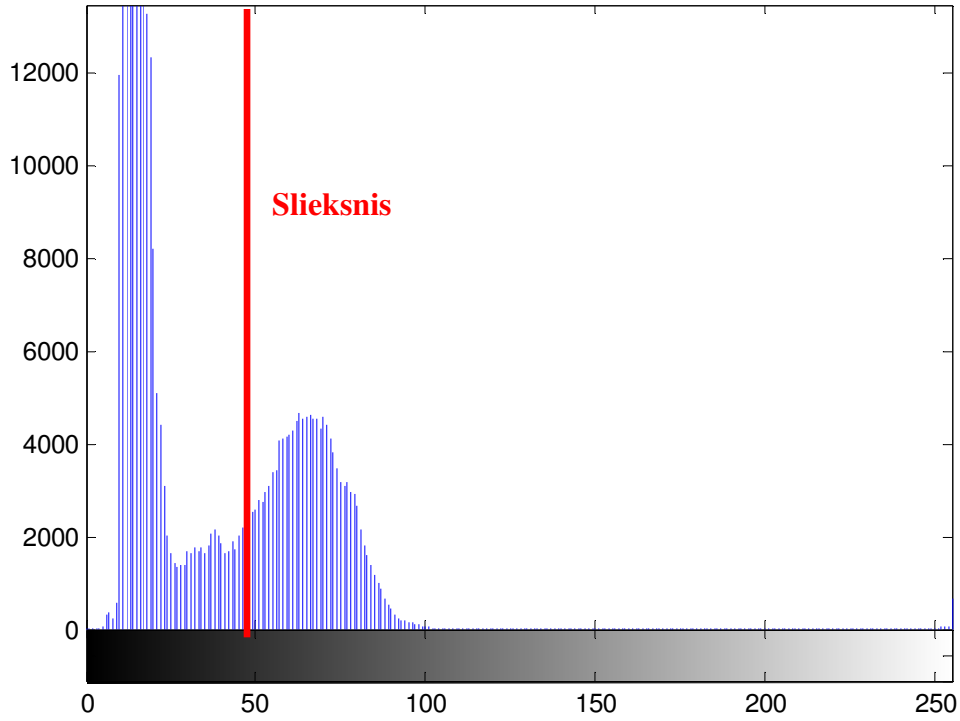
Lai precīzi noteiktu plaukstas ģeometriskos parametrus ir nepieciešams atrast stabilu atskaites punktu, no kura veikt plaukstas ģeometrisko parametru mērīšanu. Šim nolūkam var izmantot plaukstas pamatnes viduspunktu, pirkstu sākuma viduspunktu, vai arī plaukstas centrālo punktu.



Plaukstas attēls un iespējamie atskaites punkti ģeometrisko parametru reģistrācijai

Jāpatur prātā, ka veidojot bezkontakta plaukstas biometrisko parametru reģistrācijas iekārtu tai jāspēj reģistrēt ģeometriskie parametri, plaukstas nospiedums, kā arī asinsvadi. Atrodot unikālu atskaites punktu, to var izmantot ģeometrisko parametru relatīvai aprakstīšanai, kā arī kā riņķa līnijas centru, un pašu riņķi kā interesējošo reģionu, kurā nolasīt plaukstas nospieduma datus, kā arī asinsvadus.

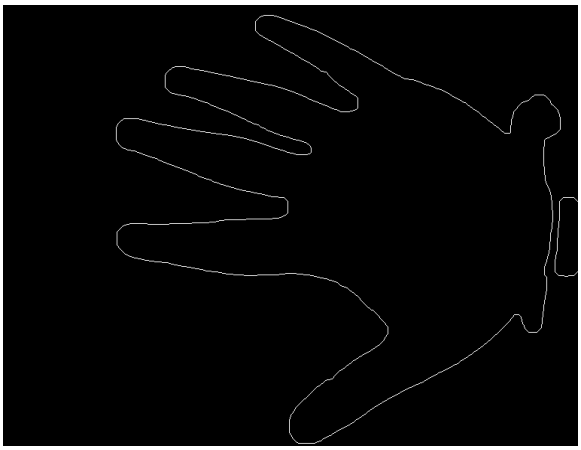
Lai plaukstas kontūrā ievilktu lielāko iespējamo riņķa līniju, un tādā veidā arī atrastu riņķa līnijas centru, no kura varētu veikt ģeometrisko parametru reģistrāciju, sākumā nepieciešamas iegūt plaukstas kontūra datus. Pirmkārt tiek veikta plaukstas attēla sliekšņošana izmantojot attēla histogrammas datus:



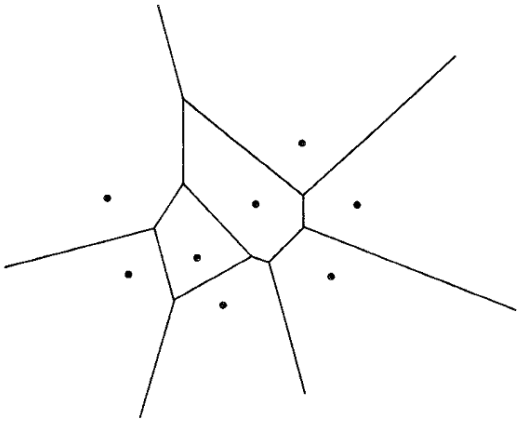
Attēla histogramma un slieksnis

Attēls pēc sliekšņošanas operācijas piemērošanas izskatās sekojoši:

Plaukstas attēls pēc kontūra detektēšanas:

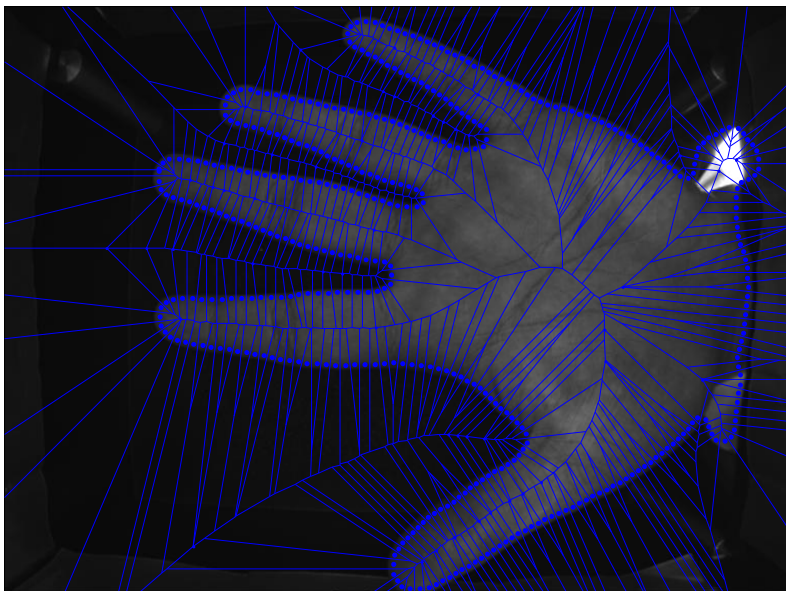


Kad plaukstas kontūra informācija ir iegūta, ir nepieciešams algoritms, kas izmantojot šos datus noteiks lielākā iespējamā riņķa pozīciju un tā centru plaukstas attēlā. Šim nolūkam var izmantot Voronoi diagrammas. Iepriekš definētai punktu kopai tiek piemeklēti reģioni kur katra punkta reģionā esošie punkti ir tuvāk reģiona atskaites punktam nekā citiem punktiem. Šos reģionus arī nosauc par Voronoi reģioniem vai šūnām. Attēlā redzamas 8 Voronoi šūnas un to sākuma punkti:



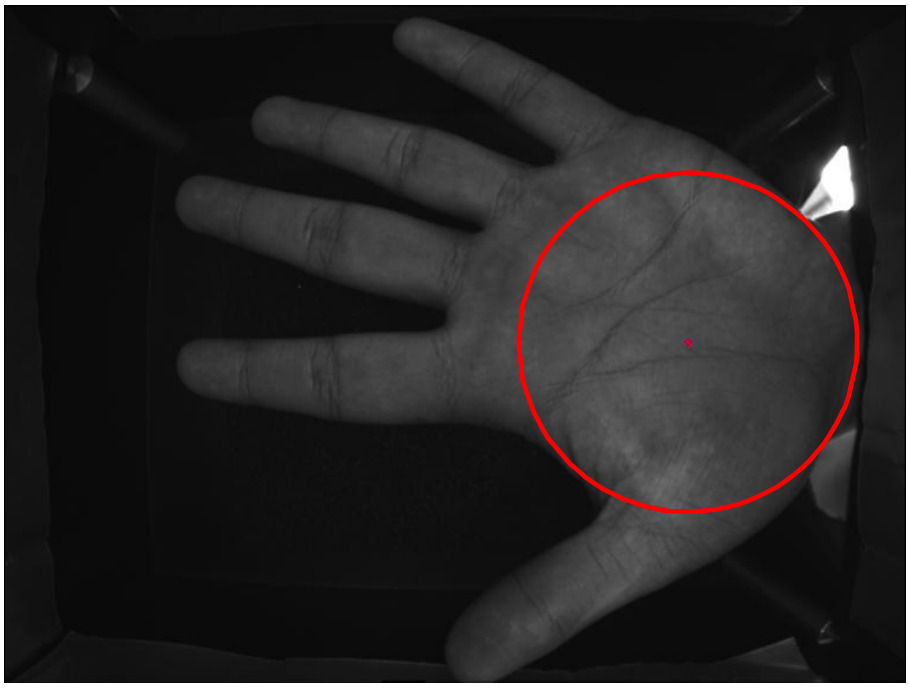
8 Voronoi šūnas

Lai efektīvi pielietotu Voronoi algoritmu plaukstas kontūra datiem, tos nepieciešams samazināt. Algoritmam tiek nodota katra peiktā nolase no plaukstas kontūra datu vektora. Rezultāts redzams attēlā:



Plaukstas kontūra aprakstīšana ar Voronoi šūnām

Lai ievilktu plaukstas kontūra iekšienē lielāko iespējamo riņķi ir nepieciešama šūnu virsotņu pārlase, kuras rezultātā iespējams atrast maksimālos attālumus līdz kontūra malām. Veicot pārlasi, tiek atrasta virsotne, kas tiek izmantota kā riņķa līnijas centrs.



Riņķa līnijas centrs var tikt izmantots kā atskaites punkts lai aprēķinātu plaukstas ģeometriskos parametrus un iegūtu relatīvos attālumus. Pati riņķa līnija var tikt izmantota kā plaukstas biometrisko parametru interesējošais reģions. Nolasot datus no šī reģiona polārajās koordinātēs ir iespējams iegūt rotācijas invarianti plaukstas asinsvadu un nospieduma datiem. Nepieciešams veikt papildus pētījumus lai novērtētu šī algoritma stabilitāti un iespēju izmantot to kā interesējošā reģiona izdalīšanas algoritmu.

Turpmāki plāni:

1. Algoritma stabilitātes novērtēšana;

2/ interesējošā reģiona izdalīšanas algoritma pilnveidošana.

Biometrisko datu apstrādes algoritmu darbības izvērtēšana

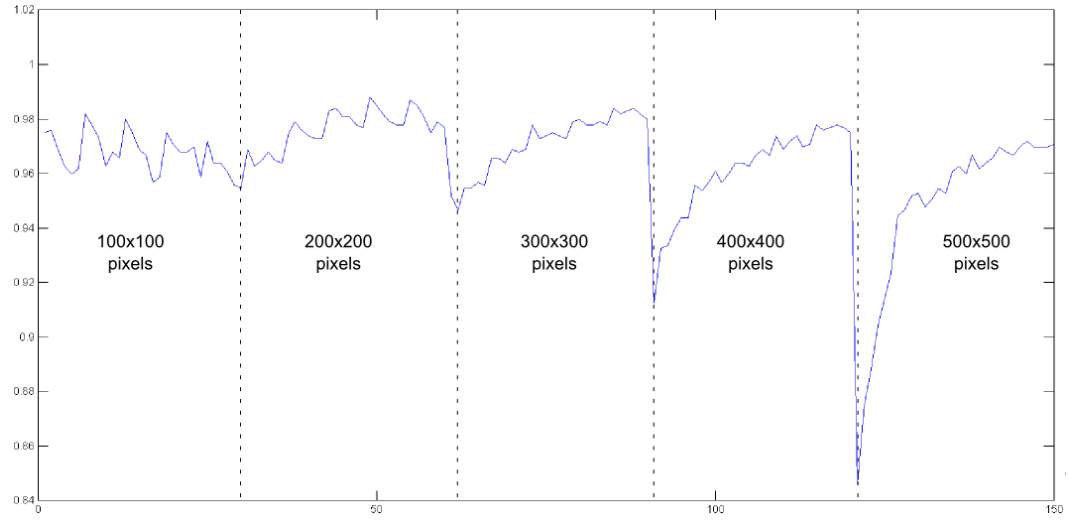
Šajā ceturksnī ir izstrādāta cilvēka sejas atpazīšanas prototipa iekārta. Tā ir iegulta sistēma uz Raspberry pi, kam ir pievienota USB kamera, ar kuras palīdzību tiek iegūti sejas attēli. Tālāk tie tiek apstrādāti ar LBP algoritmu un no iegūtā rezultāta tiek izteiktas sejas reģionu histogrammas. Histogramma tiek salīdzināta ar lokāli saglabāto datubāzi. Cilvēka sejas detektēšanai tiek izmantotas Hāra kaskādes. Programmatūras realizēšanai tiek izmantota valoda C++. Lai atvieglotu darbu ar attēliem, matricām un kaskādi, tiek izmantota OpenCV attēlu apstrādes bibliotēka.

Tika risināti arī vairāki jautājumi, kas ir saistīti ar automātisko sejas atpazīšanas algoritmu optimizāciju atbilstoši iegultas sistēmas resursiem un reāliem pielietojuma apstākļiem. Iegultiem risinājumiem raksturīgās īpašības ir sekojošas:

1) ierobežotie skaitļošanas resursi / ātrdarbība,
2) ierobežotas iespējas perifērijas kontrolē,
3) salīdzinoši nelieli atmiņas resursi.

Visas augstāk minētās nianses prasa kompromisu piemeklēšanu starp algoritmu sarežģītību un sistēmas iegulta risinājuma efektivitāti.

Pirmais aspekts, kas tika risināts ir ieejas attēla izšķirtspējas piemeklēšana. Sejas atpazīšanas process balstās uz *Local Binary Patterns* transformācijas (LBP), kas ļauj objektu reprezentēt/iekodēt kā LBP histogrammu. Histogramma ir statistiskā reprezentācija, kas uzkrāj stabilitāti pie attēla izšķirtspējas palielināšanas. Tomēr liela ieejas attēla izšķirtspēja būtiski palēnina atpazīšanas procesu, ka arī ierobežojumi perifērijas kontrolē ne vienmēr ļauj uzstādīt vēlamos parametrus. Tika detalizēti novērtēta sejas atpazīšanas algoritma precizitātes atkarība no sejas attēla izšķirtspējas. Iegūtās raksturlīknes ir sekojošas:



Lielākā uz LBP balstītā sejas atpazīšanas algoritma precizitāte (98.8% FERET datubāzei) ir sasniegta pie sejas attēla izšķirtspējas 200x200 pikseļi. Tādas sejas attēla izšķirtspējas iegūšanai ir nepieciešams ieejas attēls, kas ir vismaz divas reizes lielāks: 400x400 pikseļi, kas iegultas sistēmas gadījumā ir salīdzinoši daudz un nav efektīvi. Kompromiss tika sasniegts pie sejas attēla izmēriem, kas ir vienādi ar 100x100 pikseļi. Šajā gadījumā

atpazīšanas precizitātes vērtība ir 98.2%. Tas ļauj samazināt ieejas attēla izmērus līdz 320x240 pikseļiem, kas ir standarta izšķirtspēja daudzām USB kamerām un ļauj panākt gan pieņemamu sistēmas ātrdarbību un precizitāti, gan vienkāršu kameras aizvietošanu.

Vēlāk nācās papildināt LBP sejas atpazīšanas algoritmu ar komplicētāku klasifikācijas/atpazīšanas procesu, kas šobrīd balstās uz *Weighted Nearest Neighbour Classifier* (WNNC). WNNC izmantošanai ir nepieciešama efektīva svaru piemeklēšanas metode. Svaru piemeklēšanai tika izmantots „*Mini-batch Discriminative Feature Weighting*" algoritms, kas ļauj iteratīvi piemeklēt svarus WNNC klasifikatoram. Pēc svaru piemeklēšanas sejas atpazīšanas algoritma precizitāte paaugstinājās līdz 98.9% (FERET datubāzei, sejas attēla izšķirtspēja ir 100x100 pikseļi). Šis risinājums ir ļoti efektīvs, tad dod:

1) augstu atpazīšanas precizitāti: 98.9 %;
2) nelielu sejas attēla izšķirtspēju: 100x100 pikseļi, kas ļauj paātrināt gan sejas detektēšanas, gan sejas atpazīšanas procesus;
3) WNNC klasifikators ar LBP histogrammu svarošanas bloku līmenī prasa tikai 25 papildus reizināšanas operācijas, lai salīdzinātu ieejas attēlu ar *vienu* no attēliem datubāzē.

<u>Hāra kaskādes.</u>

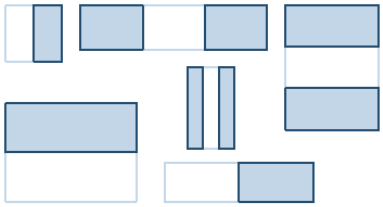
Hāra kaskādes ir pirmais klasifikators, kas spēj atrast cilvēka sejas reālajā laikā. Lai atrastu nepieciešamo objektu, algoritms veic pilnu pārlasi ar slīdošo logu, kam tiek mainīts izmērs un atrašanās vieta. Labi rezultāti tiek sasniegti izmantojot sekojošas lietas:

• vienkāršas taisnstūra veida iezīmes, ko sauc par Hāra iezīmēm (*Haar features*),

• tiek izmantots integrālais attēls aprēķinu veikšanai,

• vairāki vienkārši klasifikatori apvienoti kaskādē, kas ļauj apvienot vairākas vienkāršas iezīmes.

Hāra iezīmes apraksta spožuma jeb intensitātes atšķirību starp diviem vai vairākiem blakus esošiem taisnstūra laukumiem attēlā. Vizuāli tos var attēlot kā gaišākus un tumšākus taisnstūrus:



Pieņemsim, ka kāda no iezīmēm ir novietota virs attēla, kurā ir redzama meitene Lena:

Lai noskaidrotu iezīmes vērtību, tiktu summēta pikseļu intensitāte zem gaišās daļas un intensitāte zem zilās daļas, pēc tam tiktu aprēķināta starpība.

Lai uzlabotu veiktspēju un konstantā laikā būtu iespējams aprēķināt attēla summāro intensitāti izvēlētajā taisnstūrī, tiek izmantoti integrālie attēli. Šajā attēlā ir saglabātas visas iespējamās taisnstūra laukumu summas, kuras ir iespējams iegūt novelkot taisnstūri no viena un tā paša stūra (šajā gadījumā augšējā kreisā). Līdz ar to integrālā attēla punktā (x;y) ierakstītā vērtība ir summa, ko iegūst saskaitot visas pikseļu vērtības oriģinālajā attēlā no koordinātēm (x;y) līdz kreisam augšējam stūrim jeb koordinātēm (0;0). Oriģinālais un iegūtais integrālais attēls ir sekojošs:

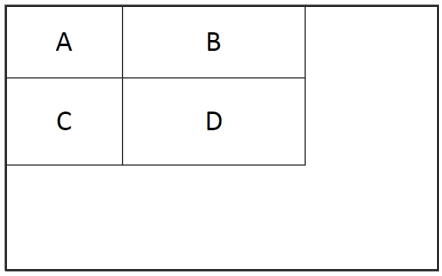
5	**2**	**3**	4	1
1	**5**	**4**	2	3
2	2	1	3	4
3	5	6	4	5
4	1	3	2	6

5	7	10	14	15
6	13	**20**	26	30
8	17	25	34	42
11	25	39	52	65
15	30	47	62	81

5 + 2 + 3 + 1 + 5 + 4 = 20

Lai noskaidrotu patvaļīga taisnstūra D (kas attēlots apakšējā attēlā) summu ir nepieciešams atņemt nevajadzīgo taisnstūru A; B; C summas. Vajadzīgo rezultātu var iegūt aprēķinot:

D = (A + B + C + D) - (A + B) - (A + C) + A



Patvaļīgā reģiona aprēķināšanas piemērs ir sekojošs:

5	2	3	4	1
1	**5**	**4**	**2**	3
2	**2**	**1**	**3**	4
3	5	6	4	5
4	1	3	2	6

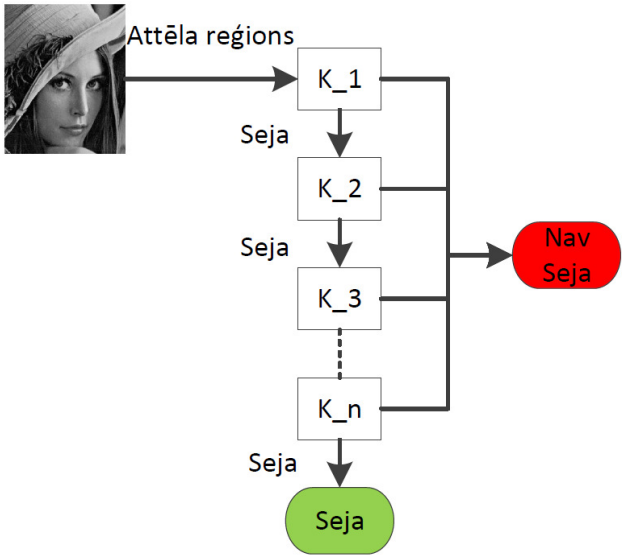
5	7	10	**14**	15
6	13	20	26	30
8	17	25	**34**	42
11	25	39	52	65
15	30	47	62	81

5 + 4 + 2 + 2 + 1 + 3 = 17 34 – 14 – 8 + 5 = 17

Šādas pieejas izmantošana ļauj vēlamā reģiona summu aprēķināt konstantā laikā. Lai noskaidrotu kādas iezīmes nepieciešams izmantot un kādas sliekšņa vērtības izmantot

atpazīšanā, ir nepieciešams pielietot mašīnmācības algoritmu *AdaBoost*. Šis algoritms apvieno vairākus vājus klasifikatorus, lai izveidotu vienu spēcīgu. Vāji tie ir tādā nozīmē, ka tie pareizu atbildi atgriež nedaudz biežāk nekā veicot minēšanu. *AdaBoost* izvēlas kopu ar vājiem klasifikatoriem, ko apvienot un piešķir katram svara koeficientu, šī kombinācija sniedz spēcīgu klasifikatoru. Izmantotie vājie klasifikatori ir Hāra iezīmes.

Pārbaudot vai izvēlētajā attēla reģionā ir meklējamais objekts, tiek pielietotas atlasītās iezīmes. Tās tiek pielietotas vien pēc otras, sākot ar vienkāršāko un beidzot ar sarežģītāko. Iegūtā iezīmju kaskāde izskatās sekojoši:



Līdz ar to atpazīšanas algoritms tiek saukts par Hāra kaskādi. Ja kaut vienai no iezīmēm summu starpība nepārsniedz vēlamo slieksni, tad meklēšana tiek pārtraukta un tiek pateikts, ka attēlā nav redzams meklējamais objekts. Ja ir pārbaudītas visas iezīmes un visām ir pārsniegts vajadzīgais slieksnis, tad tiek nolemts, ka apskatāmajā attēla reģionā ir meklējamais objekts.

Turpmākie plāni:

1. Novērtēt funkcionālo bloku nepieciešamību automātiskajā sejas atpazīšanas procesā ar mērķi izslēgt tos blokus, kas var pasliktināt atpazīšanas procesu un ievērojami samazināt ātrdarbību.
2. Pilnveidot un papildināt izveidoto programmu, novēršot darba procesā iegūtās kļūdas.

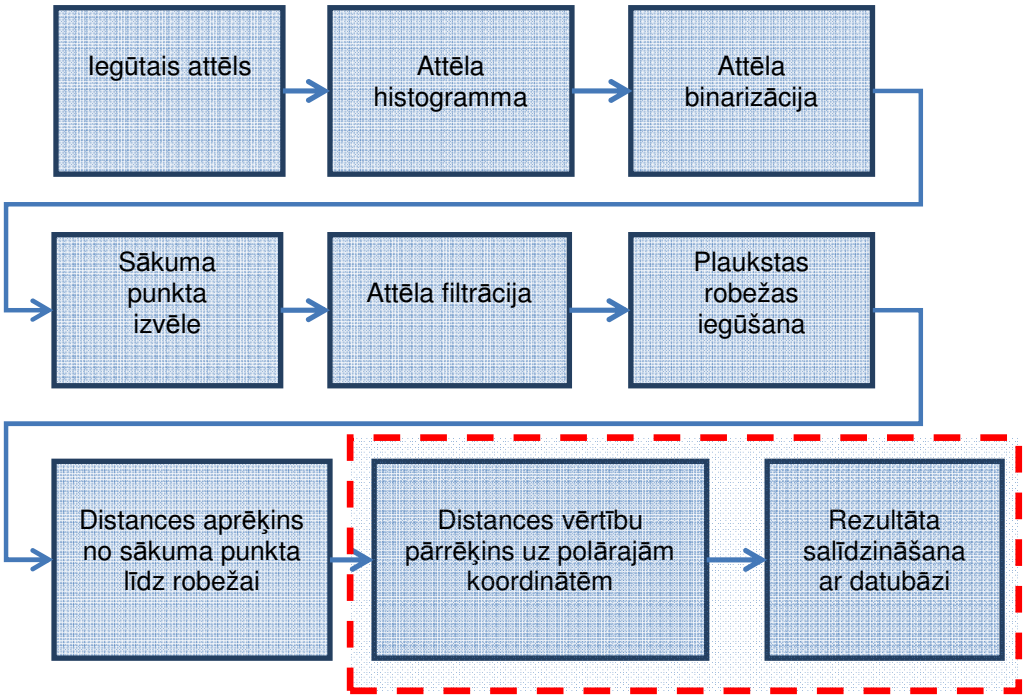
Biometrisko datu iegūšanas algoritmu paralelizācija un implementēšana programmējamos loģiskos masīvos

Plaukstas ģeometrijas algoritmu izstrāde un implementācija FPGA

Iepriekšējās atskaitēs tika aprakstīti jau izveidotie plaukstas ģeometrijas algoritmi, šajā ceturksnī tie tika testēti un uzlaboti, kā arī daži algoritmu izpildes posmi tika implementēti FPGA.

Plaukstas ģeometrijas algoritmu turpinājums:

Esošie algoritmi balstās uz viena darbības principa, kas parādīts sekojošajā shēmā:

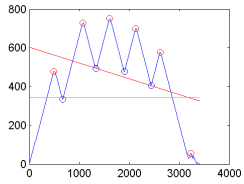
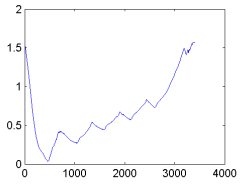
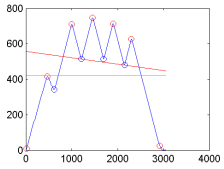
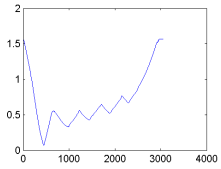


Algoritma katrs atsevišķais posms tika apskatīts iepriekšējās atskaitēs. Neskaidrais jauninājums ir ar sarkano rāmīti apvilktais posms, kurā distances vērtības tiek pārrēķinātas no Dekarta uz polārajām koordinātēm. Pārrēķina rezultātā katrai vērtībai mēs iegūstam leņķa vērtību(φ) un attālumu no sākuma punkta (r). Balstoties uz testēšanas rezultātiem tika secināts, ka leņķa vērtība(φ) „nes" sevī pietiekoši daudz informācijas, lai varētu veikt plaukstu salīdzināšanu. Šāda veida salīdzināšana ir nepieciešama, lai atsijātu tādus plaukstas ģeometrijas attēlus, kuri būtiski atšķiras no apskatāmās personas plaukstas attēla.

Turpmākie pētījumi:

1. Pirmais uzdevums saistīts ar distances vērtību spektra analizēšanu, kas varētu ļaut iegūt katras personas plaukstas ģeometrisko parametru unikālāko informāciju.
2. Sekojošais uzdevums ir algoritma testēšana un uzlabošana (tā, lai tas atvieglotu implementāciju FPGA).



Attēlos redzami divu dažādu personu plaukstas attēli, kuri tiek analizēti ar plaukstas ģeometrijas algoritmu. Augšējais attēls ir apstrādāts robežas iegūšanai. Apakšējie attēli ir attiecīgi: robežas leņķa vērtības(*φ)* un attālums no sākuma punkta (*r*).

Attēla histogrammas iegūšanas algoritms

Līdz šim attēla histogramma tika iegūta izmantojot MATLAB programmēšanas vides iekšējās funkcijas. Tā kā plaukstas ģeometrijas algoritms ir jārealizē FPGA, tad nācās izveidot savu histogrammas aprēķināšanas algoritmu.

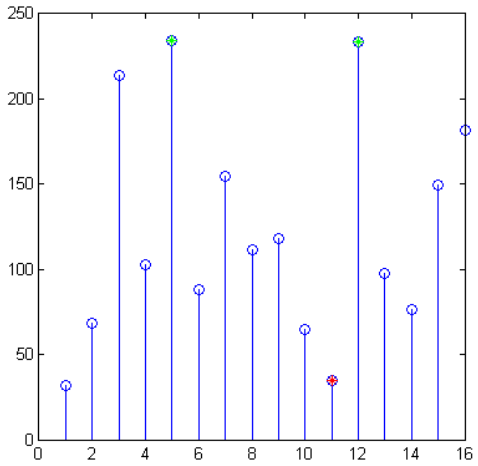
Algoritma izveide un testēšana MATLAB

Attēla histogramma ar L iespējamajiem intensitātes līmeņiem diapazonā no [0..G] ir definēta kā diskrēta funkcija:

$$h(r_k) = n_k,$$

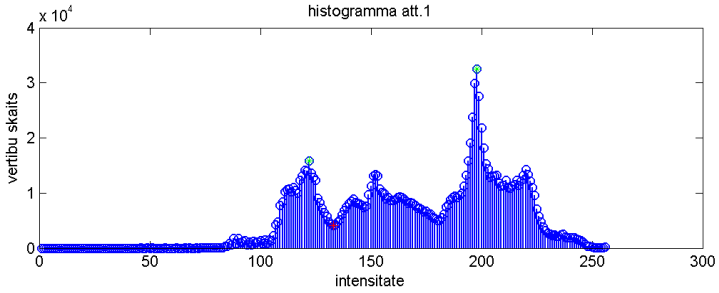
kur r_k – k-tie intensitātes līmeņi intervālā [0..G] un n_k - pikseļu daudzums konkrētajā attēlā. Mūsu sistēmā attēli tiek iegūti izmantojot 8 bitus uz katru pikseli ar izšķirtspēju 640x480, tādējādi histogrammai jāsastāv no L = 256 intensitātes līmeņiem un pikseļu daudzums ir k=307200.

Attēla histogramma iegūst informāciju par intensitātes līmeņu sadalījumu attēlā. Atrodot minimumu starp diviem intensitāšu pīķiem tiek iegūta *sliekšņa* vērtība, ar kuras palīdzību attēls tiek binarizēts, tādējādi plauksta izdalās no apkārtējā fona.



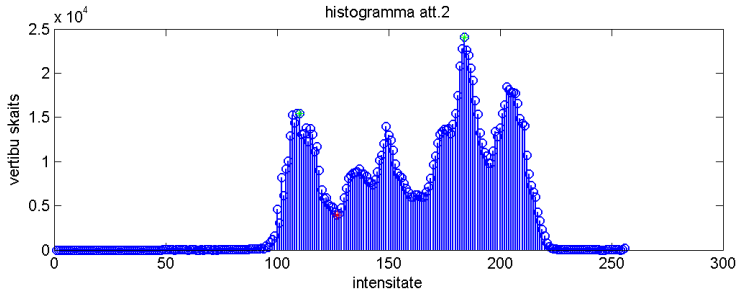
Šajā attēlā redzams piemērs kā izskatās histogramma, kad L=16 un k=256, kur dati ir *random* skaitļi. Ar zaļiem punktiem tiek apzīmētas divas maksimālo intensitāšu pīķu vērtības, bet ar sarkanu ir atzīmēta izrēķinātā *sliekšņa* vērtība.

Divu dažādu attēlu apstrāde histogrammas novērtēšanai un sliekšņošanai starp MATLAB iebūvēto funkciju (I=imhist(attēls); T = greytresh(attēls)) un mūsu realizēto.



Matlab slieksnis

Apreķinātais slieksnis



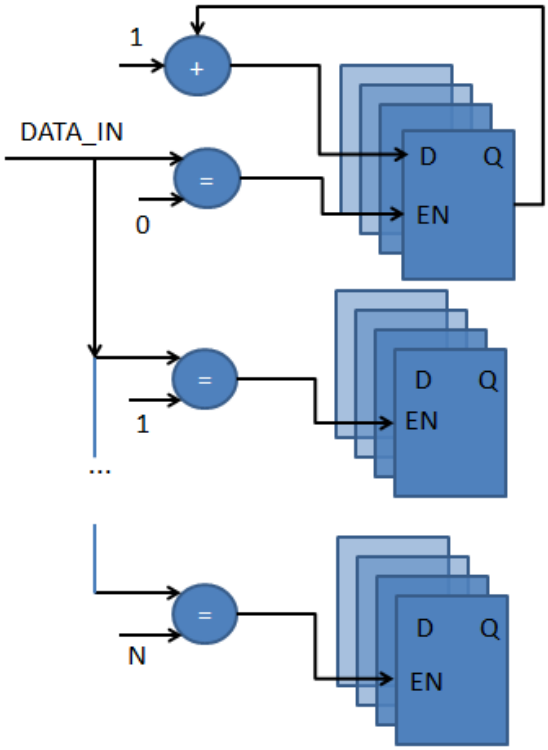
Matlab slieksnis

Apreķinātais slieksnis

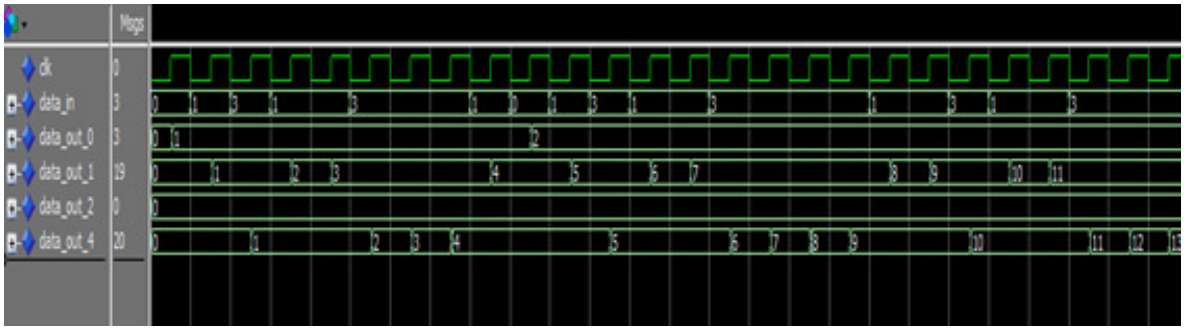
Histogrammas realizācija FPGA

Ar attēla histogrammas aprēķināšanas funkciju tiek uzsākta plaukstas ģeometrijas algoritma implementācija FPGA. Histogrammas analīze FPGA ļaus novērtēt vai esošie plaukstas ģeometrijas algoritmi spēj pildīt savas funkcijas pie dažādiem apgaismojumiem un dažādiem foniem (vai arī tas notiek tikai pie melna fona).

Histogrammas aprēķina realizācija :



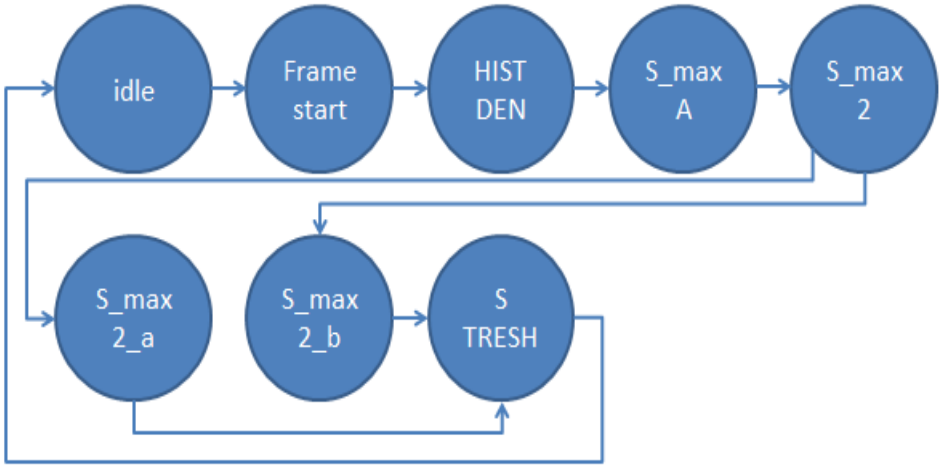
Attēlā redzama histogrammas aprēķina realizācija FPGA. Tiek izmantots ieejas signāls, kura vērtības (intensitātes līmeņi N=L) tiek salīdzinātas ar ciklā izsauktiem skaitļiem atkarībā no ieejas intensitātes diapazona. Atkarībā no intensitātes līmeņa, tai atbilstošajam atmiņas reģistram tiek palielināta viena vērtība.



Esošā pieeja ir simulēta MODELSIM simulācijas vidē. Programma darbojas korekti.

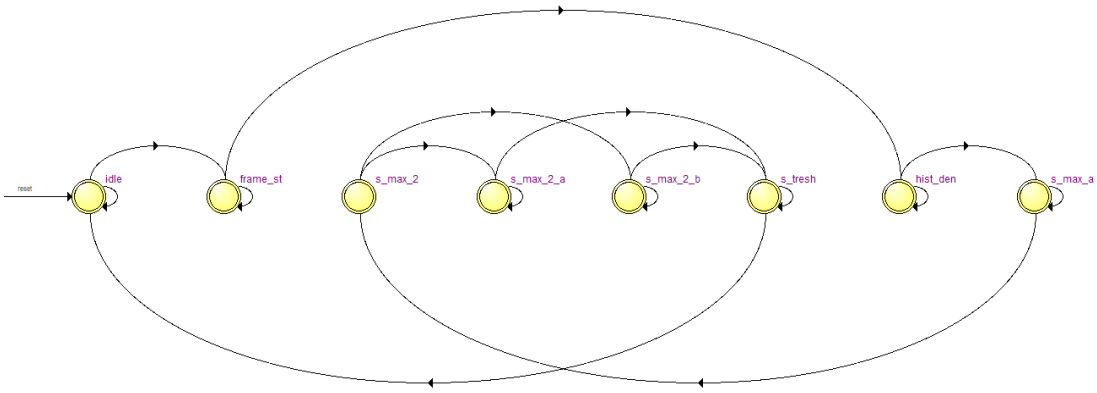
Sliekšņa vērtības aprēķins

Lai aprēķinātu sliekšņa vērtību FPGA, nepieciešams izmantot FSM – galīgu stāvokļu automātu:



Attēlā redzamais *Idle* ir sākuma stāvoklis, kur tiek gaidīts pēc atļaujas sākt histogrammas aprēķinu. *Frame start* tiek gaidīts pēc kadra sākuma signāla, kurš liecina, ka attēlu sensors sācis iegūt bildi, *HIST DEN* stāvoklī tiek sagaidītas kadra beigas un apturēts histogrammas aprēķins, *S_max A* tiek meklēta maksimālā vērtība starp histogrammas datiem. *S_max_2* sazarojas *S_max_2_a S_max_2_b* atkarībā no tā, kurā apgabalā (0..N/2 vai N/2..N) atrodas maksimums. *S TRESH* stāvoklī tiek atrasta minimālā vērtība starp diviem maksimumiem, kas ir arī attēla binarizācijas sliekšņa vērtība.

Iegūtais FSM QUARTUS II izstrādes vidē



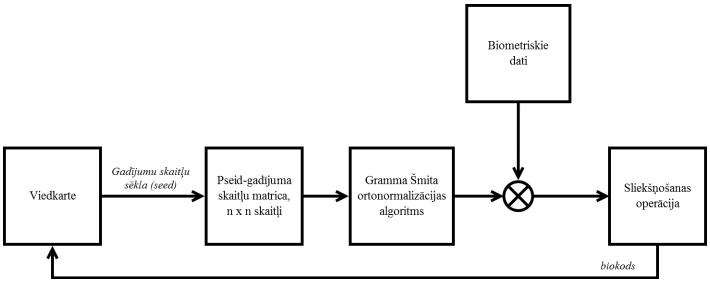
Turpmākie pētījumi:

1. Pabeigt un notestēt FSM sliekšņa vērtības;
2. Testi attēla histogrammai reālos apstākļos ar FPGA;
3. Turpināt plaukstas ģeometrijas algoritma posmu realizāciju FPGA

Biometrisko datu kriptēšanas, glabāšanas un lietojuma apakšistēmas izveide

`BIO HASH` algoritms

Līdz šim apskatītais `BIO HASH` kods tika iegūts, izmantojot 1. attēlā redzamo blokshēmu.



1. att. `BIO HASH` koda iegūšanas algoritma blokshēma

Šāda `BIO HASH` koda algoritma implementēšana nav optimāla, šādu apsvērumu dēļ:

- izpildot Gramma Šmita ortonormalizēšanas operāciju, pseido gadījuma skaitļu matricas katrai rindai v, sākot no 2. (šīs rindas varam uztvert kā vektorus n dimensiju telpā un apzīmēt kā v_i) ir jāatņem šīs rindas projekcija uz citām, iepriekšējajām ortonormētajām rindām (vektoriem) $e_{i<n}$, --- kas nozīmē, ka pēdējai rindai, ja $n = 512$, no tās būs jāatņem rindas projekcijas uz iepriekšējām 511 rindām, kas nozīmē, ka aprēķiniem nepieciešams *liels* bitu platums - Gramma Šmita operāciju pēdējai rindai varam aprakstīt ar formulu:

$$u_{512} = v_{512} - \sum_{i=1}^{511} proj_{e_i}(v_{512}) \tag{1}$$

  Kur

$$e_i = \frac{u_i}{||u_i||} = \frac{u_i}{\sqrt{\sum_{j=1}^{512} q_j^2}} \tag{2}$$

  u_i --- ar Gramma Šmita operāciju jau ortogonalizētas (pret iepriekšējām u_{i-1} rindām) bet nenormētas;
  q_j --- u_i rindas elements (koordināte).
  Tas nozīmē, ja aprēķinātā *individuālā projekcija* $proj_{e_j}(v_{512})$ tiek aprakstīta ar 8 *bitiem*, tad, lai *bez bitu pārplūdes* aprakstītu izteiksmes (1.) otro locekli, nepieciešami jau $8 + 511 = 519$ biti, kas nozīmētu, ka ar šādu bitu platumu ir jāglabā mainīgie u_i, kas nav *samērīgs* bitu platums, aprēķinu veikšanai. Līdzīga *problēma* veidojas, arī, veicot rindu normalizēšanas operāciju (2.), jo dalītāju veido 512 elementu (celtu kvadrātā) summa.

- Ja $n = 512$, tad, izpildot Gramma Šmita operāciju ir jāveic aptuveni $310'000'000$ reizināšanas un $310'000'000$ summēšanas operācijas. Ja reizināšanas operācijas veikšanai tiek implementēts procesors `FPGA` programmējamajā loģikā, un reizināšanas operāciju var veikt katrā taktī ar `clk = 125 Mhz` takts frekvenci, tad reizināšanas darbības vien aizņem 3 sekundes --- t.i. par šādu laiku tiks palēnināts autentifikācijas process.

- Minimālais atmiņas apjoms tikai datu glabāšanai, pieņemot, ka operācijas tiek veiktas ar 32 bitu platumu, ir

$$32 \cdot n^2 = 32 \cdot 512^2 = 8.39Mb$$

  Šāds atmiņas apjoms pieejams vienīgi izmantojot ārēju atmiņas avotu (nevis `FPGA` integrālās mikroshēmas iekšējos resursus), kas vēl vairāk palēnina sistēmas ātrdarbību, aizturu, kas veidojas, griežoties pie ārējās atmiņas, dēļ.

- Šādu sistēmu gandrīz neiespējami parametrizēt --- divkāršojoties biometrisko datu nolasēm n, glabājamo datu skaits pieaug četrkārši, bet izpildāmo operāciju skaits --- vēl *vairāk*.

Tādēļ tika meklēts cits `BIO HASH` operācijas implementēšanas algoritms --- precīzāk --- cits ortonormētu matricu iegūšanas veids --- tāds piedāvāts, izmantojot *rotācijas matricas*.

Vektora rotācija

Vektora, kuru varam uzdot ar koordinātām $X = (x, y)$ rotāciju iespējams veikt, izmantojot elementāro rotācijas matricu [1] $R(\Theta)$.

$$R(\Theta) = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \tag{3}$$

Veicot vektora rotāciju:

$$Y = R(\Theta) \cdot X = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \cdot \cos\Theta - y \cdot \sin\Theta \\ x \cdot \sin\Theta + y \cdot \cos\Theta \end{bmatrix}$$

Varam redzēt, ka jauniegūtā vektora Y elementus veido svērta ``ieejas" elementu reizinājums ar $\sin\Theta$ vai $\cos\Theta$. Ja mēs ieejas vektoru uzdodam kā:

$$X = \begin{bmatrix} \cos\Psi \\ \sin\Psi \end{bmatrix}$$

Tad:

$$Y = R(\Theta) \cdot X = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \cdot \begin{bmatrix} \cos\Psi \\ \sin\Psi \end{bmatrix} = \begin{bmatrix} \cos(\Psi + \Theta) \\ \sin(\Psi + \Theta) \end{bmatrix}$$

T.i. iegūtais vektors Y ir sākotnējais vektors X rotēts par leņķi Θ .

Elementārā rotācijas matrica $R(\Theta)$ ir ortogonāla un ortonormēta (to pierāda tas, ka matricas $R(\Theta)$ determinants $D = -1$, gan rindas, gan kolonnas elementu kvadrātu vērtību summa ir 1, bet ja mēs reizinām katras rindas elementus, un tad tos summējam, tad iegūstam 0), kas nozīmē, ka šīs matricas rindas ir elementāras bāzes funkcijas [2] --- tas nozīmē, ka mēs arī varam apgalvot, ka ieejas signāls X ir izvērsts Furjē rindā, kuras bāzes funkcijas ir $R(\Theta)$ rindas. Iegūtais signāla spektrs Y ir ieejas signāla X rotācija.

Vairākdimensiju vektoru rotācija

Avotā [2] un [3] piedāvāts izmantot kāpņveida rotācijas matricu --- Stairs-like Orthonormal Generalized Rotation Matrix (SOGRM). Matrica, ja N (rotējamā vektora nolašu skaits) ir 4, izskatās šadi:

$$B_4 = \begin{bmatrix} \sin\Theta & \cos\Theta & 0 & 0 \\ 0 & 0 & \sin\Theta & \cos\Theta \\ \cos\Theta & -\sin\Theta & 0 & 0 \\ 0 & 0 & \cos\Theta & -\sin\Theta \end{bmatrix} \tag{4}$$

Redzam, ka formulas (4.) *iekrāsotā daļa* ir jau iepriekš apskatītā *elementārā rotācijas matrica* (3.). Pie kam:

$$Y = H \cdot X = B_4^2 \cdot X = B_4 \cdot (B_4 \cdot X) \tag{5}$$

Vispārīgā gadījumā:

$$Y = B_N^{\log_2(N)} \cdot X = B_N \cdot (B_N \cdot (\ldots (B_N \cdot X))) \tag{6}$$

Tātad, vairākdimensiju vektora rotācijas gadījumā ir jāveic

$$K = \underbrace{\frac{N}{2}}_{\substack{\text{tik elementārās rotācijas} \\ \text{matricas kāpņveida matricā } B_N}} \cdot \underbrace{\log_2(N)}_{\text{tik kāpņveida matricas } B_N} \tag{7}$$

rotācijas.
Varam arī pārliecināties, ko tad īsti nozīmē darbība $B_4 \cdot B_4$, (5.) formulā:

$$B_4 \cdot B_4 = \begin{bmatrix} \sin\Theta & \cos\Theta & 0 & 0 \\ 0 & 0 & \sin\Theta & \cos\Theta \\ \cos\Theta & -\sin\Theta & 0 & 0 \\ 0 & 0 & \cos\Theta & -\sin\Theta \end{bmatrix} \cdot \begin{bmatrix} \sin\Theta & \cos\Theta & 0 & 0 \\ 0 & 0 & \sin\Theta & \cos\Theta \\ \cos\Theta & -\sin\Theta & 0 & 0 \\ 0 & 0 & \cos\Theta & -\sin\Theta \end{bmatrix} = \\ \begin{bmatrix} \sin\Theta^2 & \cos\Theta\,\sin\Theta & \cos\Theta\,\sin\Theta & \cos\Theta^2 \\ \cos\Theta\,\sin\Theta & -\sin\Theta^2 & \cos\Theta^2 & -\cos\Theta\,\sin\Theta \\ \cos\Theta\,\sin\Theta & \cos\Theta^2 & -\sin\Theta^2 & -\cos\Theta\,\sin\Theta \\ \cos\Theta^2 & -\cos\Theta\,\sin\Theta & -\cos\Theta\,\sin\Theta & \sin\Theta^2 \end{bmatrix} \tag{8}$$

T.i. divas atkārtotas reizināšanas darbības ar kāpņveida matricu B_4 aizvieto vienu sarežģītāku reizināšanas darbību.
Tā pat, kā iepriekš, matricas $B_N \ldots B_N^{\log_2}$ ir gan ortogonālas, gan ortonormētas, t.i. var kalpot kā bāzes funkcijas.
Izpētīsim, kā tad ieejas signāls X tiek ``rotēts'':

$$\begin{aligned} Y = H \cdot X &= B_4 \cdot (B_4 \cdot X) = \\ &= \begin{bmatrix} \sin\Theta & \cos\Theta & 0 & 0 \\ 0 & 0 & \sin\Theta & \cos\Theta \\ \cos\Theta & -\sin\Theta & 0 & 0 \\ 0 & 0 & \cos\Theta & -\sin\Theta \end{bmatrix} \cdot \begin{bmatrix} \sin\Theta & \cos\Theta & 0 & 0 \\ 0 & 0 & \sin\Theta & \cos\Theta \\ \cos\Theta & -\sin\Theta & 0 & 0 \\ 0 & 0 & \cos\Theta & -\sin\Theta \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \\ &= \begin{bmatrix} \sin\Theta & \cos\Theta & 0 & 0 \\ 0 & 0 & \sin\Theta & \cos\Theta \\ \cos\Theta & -\sin\Theta & 0 & 0 \\ 0 & 0 & \cos\Theta & -\sin\Theta \end{bmatrix} \cdot \begin{bmatrix} b\cos\Theta + a\sin\Theta \\ d\cos\Theta + c\sin\Theta \\ a\cos\Theta - b\sin\Theta \\ c\cos\Theta - d\sin\Theta \end{bmatrix} = \\ &= \begin{bmatrix} \cos\Theta\,(d\cos\Theta + c\sin\Theta) + \sin\Theta\,(b\cos\Theta + a\sin\Theta) \\ \cos\Theta\,(c\cos\Theta - d\sin\Theta) + \sin\Theta\,(a\cos\Theta - b\sin\Theta) \\ \cos\Theta\,(b\cos\Theta + a\sin\Theta) - \sin\Theta\,(d\cos\Theta + c\sin\Theta) \\ \cos\Theta\,(a\cos\Theta - b\sin\Theta) - \sin\Theta\,(c\cos\Theta - d\sin\Theta) \end{bmatrix} = \\ &= \begin{bmatrix} a\cdot\sin\Theta^2 & b\cdot\cos\Theta\,\sin\Theta & c\cdot\cos\Theta\,\sin\Theta & d\cdot\cos\Theta^2 \\ a\cdot\cos\Theta\,\sin\Theta & -b\cdot\sin\Theta^2 & c\cdot\cos\Theta^2 & -d\cdot\cos\Theta\,\sin\Theta \\ a\cdot\cos\Theta\,\sin\Theta & b\cdot\cos\Theta^2 & -c\cdot\sin\Theta^2 & -d\cdot\cos\Theta\,\sin\Theta \\ a\cdot\cos\Theta^2 & -b\cdot\cos\Theta\,\sin\Theta & -c\cdot\cos\Theta\,\sin\Theta & d\cdot\sin\Theta^2 \end{bmatrix} \end{aligned} \tag{9}$$

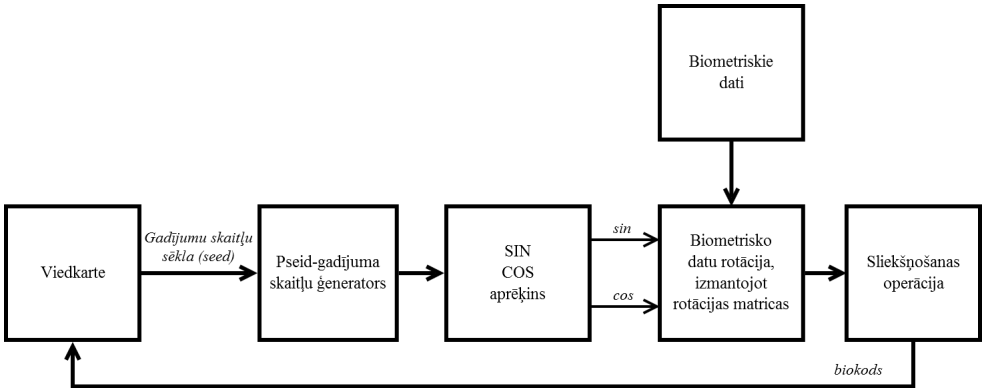
Vēlreiz varam pārliecināties, ka arī šeit ir spēkā asociatīvā īpašība, kā jau apskatīts formulā (5.), t.i. --- $H \cdot X = B_4^2 \cdot X = B_4 \cdot (B_4 \cdot X)$ --- to varam novērtēt, salīdzinot izteiksmes (9.) un (8.) --- izteiksmes (9.) gala rezultāts ir vienāds ar izteiksmi (8.), pareizinātu ar ieejas signālu X.
No piemēra (9.) arī redzams, ka, izmantojot konstantu rotācijas leņķi Θ, iegūtās kopējās rotācijas ortogonālā matricas $H = B_N^{\log_2(N)}$ koeficientus veido lielumu (skaitļu) --- $\sin\Theta, \cos\Theta$ un $-\sin\Theta$ galīga skaita kombinācijas, t.i. pašas rotācijas matricas (par ko mēs arī interesējamies) koeficienti atkārtojas, kā piemēru varam apskatīt matricu B_4^2, ja $\Theta = \pi/8$:

$$H = \begin{bmatrix} 0.0670 & 0.2500 & 0.2500 & 0.9330 \\ 0.2500 & -0.0670 & 0.9330 & -0.2500 \\ 0.2500 & 0.9330 & -0.0670 & -0.2500 \\ 0.9330 & -0.2500 & -0.2500 & 0.0670 \end{bmatrix}$$

Rotācijas matricas koeficientus var *vairāk sajaukt*, lietojot atšķirīgus rotācijas leņķus --- Θ, mainot tos katrā rotācijā, K reizes --- (7.) formula. Ja rotācijas leņķi $\Theta_1 = 10^o; \Theta_2 = 20^o; \Theta_3 = 30^o; \Theta_4 = 40^o$, tad:

$$
\begin{aligned}
B_4 \cdot B_4 &= \begin{bmatrix} \sin\Theta_1 & \cos\Theta_1 & 0 & 0 \\ 0 & 0 & \sin\Theta_2 & \cos\Theta_2 \\ \cos\Theta_1 & -\sin\Theta_1 & 0 & 0 \\ 0 & 0 & \cos\Theta_2 & -\sin\Theta_2 \end{bmatrix} \cdot \begin{bmatrix} \sin\Theta_3 & \cos\Theta_3 & 0 & 0 \\ 0 & 0 & \sin\Theta_4 & \cos\Theta_4 \\ \cos\Theta_3 & -\sin\Theta_3 & 0 & 0 \\ 0 & 0 & \cos\Theta_4 & -\sin\Theta_4 \end{bmatrix} = \\
&= \begin{bmatrix} 0.1736 & 0.9848 & 0 & 0 \\ 0 & 0 & 0.3420 & 0.9397 \\ 0.9848 & -0.1736 & 0 & 0 \\ 0 & 0 & 0.9397 & -0.3420 \end{bmatrix} \cdot \begin{bmatrix} 0.5000 & 0.8660 & 0 & 0 \\ 0 & 0 & 0.6428 & 0.7660 \\ 0.8660 & -0.5000 & 0 & 0 \\ 0 & 0 & 0.7660 & -0.6428 \end{bmatrix} = \\
&= \begin{bmatrix} 0.0868 & 0.1504 & 0.6330 & 0.7544 \\ 0.2962 & -0.1710 & 0.7198 & -0.6040 \\ 0.4924 & 0.8529 & -0.1116 & -0.1330 \\ 0.8138 & -0.4698 & -0.2620 & 0.2198 \end{bmatrix}
\end{aligned}
\tag{10}
$$

Redzam, ka skaitliskie matricas koeficienti vairs neatkārtojas --- tātad, kā ``gadījuma'' parametrs var tikt izmantots rotācijas leņķis K. Piemēram, ja tiek izmantotas $N = 512$ biometrisko datu nolases, tad izmantojamo *gadījuma* leņķu skaits $K = 9 \cdot 256 = 2304$, Tas nozīmē ka, `biokoda` iegūšanai var izmantot *optimizētu* algoritmu, kura blokshēma apskatāma 2. attēlā.



2. att. `BIO HASH` koda iegūšanas optimizētā algoritma blokshēma

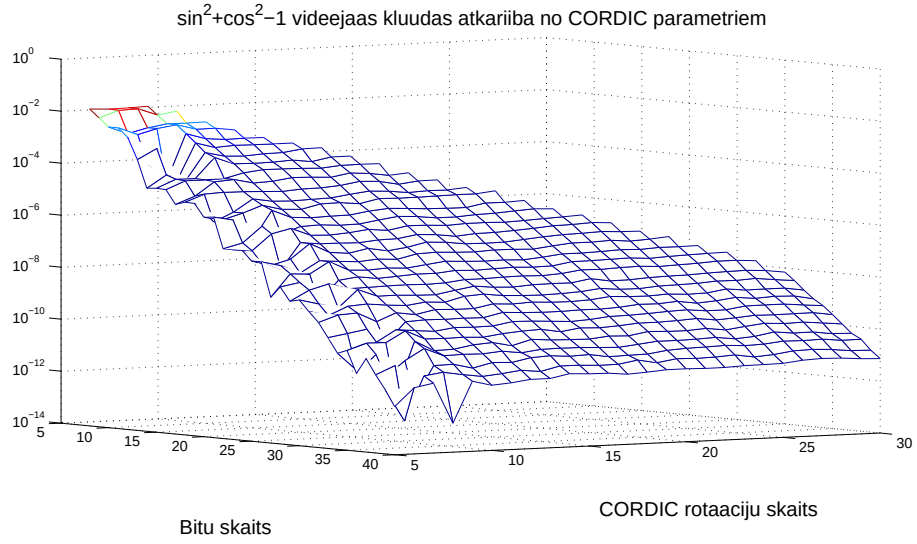
Vispārīgi nav nepieciešams rēķināt $\sin\Theta$ un $\cos\Theta$ vērtības, bet gan nepieciešams nosacījums, ka:

$$(a(\Theta))^2 + (b(\Theta^2)) - 1 \to 0 \tag{11}$$

kur $a(\Theta)$ un $b(\Theta)$ --- lielumi, kas atkarīgi no Θ, iegūti matemātisku darbību ceļā. No uzdotā nosacījuma (11.), iegūstam formulas:

$$\begin{cases} a = \Theta \\ b = \sqrt{1 - a^2} \end{cases}$$

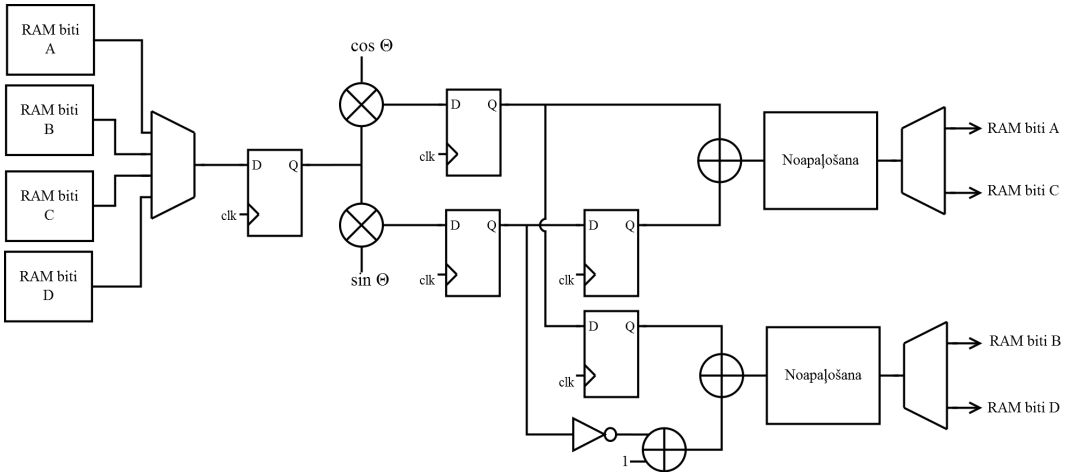
Bet, izmantojot `FPGA` programmējamos masīvus, ir sarežģīti veikt kvadrātsaknes *vilkšanas* operāciju, kā arī, izmantojot šādu lielumu a un b aprēķināšanas pieeju, būtu nepieciešams meklēt veidu, kā ar *pseidogadījuma* raksturu mainīt lieluma b zīmi, tādēļ izvēlēts lietot `CORDIC` algoritmu, $\sin\Theta$ un $\cos\Theta$ tuvinātai aprēķināšanai, jo šis algoritms izmanto tikai saskaitīšanas un atņemšanas, kā arī bitu nobīdes darbības, un ir jau izmantots iepriekš. Mainot `CORDIC` *rotāciju* skaitu un bitu skaitu starprezultātu glabāšanai, iespējams uzdot noteiktu izteiksmes $a^2 + b^2 - 1 = 0$ precizitāti --- (3.) attēls.



3. att. izteiksmes $a^2 + b^2 - 1 = 0$ vidējās kļūdas atkarība, no `CORDIC` algoritma implementācijas parametriem

Vektora rotācijas implementēšana `FPGA`

Iepriekš apskatīto vektora rotāciju, izmantojot rotācijas matricas, funkcionalitāti var implementēt `FPGA`, izmantojot blokshēmu, kas apskatāma (4.) attēlā.



4. att. Vektora rotācijas implementācijas blokshēma

Attēlā **nav** parādīta sistēmas kontroles shēma, kā arī tas, kā ieejas signāls X (biometriskie dati) tiek sākotnēji ierakstīts, un kā izejas signāls Y (*biokods*, pirms sliekšņošanas operācijas) tiek padots izejā, piemērs apskatīts, pieņemot, ka tiek izmantotas $N = 512$ biometrisko datu nolases.
Implementējamā struktūra sastāv no četriem `RAM bits` blokiem --- `A;B;C;D`. Katrā blokā iespējams saglabāt $512/2 = 256$ nolases. Ja pieņemam, ka veicot aprēķinus, katras nolases glabāšanai tiks atvēlēti 32 biti --- 16 biti veselajai daļai, bet 16 --- daļai aiz komata, tad tas nozīmē, ka kopējais nepieciešamais `RAM bitu` apjoms ir:

$$256 \cdot 4 \cdot 32 = 32'768 \text{ biti}$$

ko mēs varam nosaukt par *mazu* lielumu, salīdzinājumā ar to `RAM bitu` apjomu, ja būtu jāsaglabā visi rotācijas matricas H koeficienti.
Attēlā 4. parādītais implementācijas veids darbojas pēc šādas shēmas --- iterācijā n jau $n-1$

reizes rotētās ieejas signāla nolases atrodas `RAM bitu` blokos `A` un `B`, pie kam $0 \dots 255$ nolase tiek glabāta `A RAM bitu` blokā, bet $256 - 511$ nolase - `B RAM bitu` blokā. To *zin* vadības mehānisms, kurš ar multipleksora palīdzību izvēlas attiecīgos `RAM bitu` blokus.
Tālākā bloka darbība veidota, izmantojot ``*transposed* '' struktūru [4]. T.i. katra padotā signāla nolase *uzreiz* tiek reizināta gan ar sin Θ, gan ar cos Θ, bet sareizinātie rezultāti aizturēti, atkarībā no tā, vai tiek aprēķināta jaunā vērtība ar kārtas numuru $0 \dots 255$, vai $256 - 511$. (gadījumā, ja tiek aprēķināta jaunās nolases vērtība ar numuru $256 - 511$, tad jau iegūtā reizinājuma ar sin Θ vērtība tiek invertēta, un tai tiek pieskaitīts 1 - t.i. vērtība tiek reizināta ar -1.
Gatavās vērtības tiek noapaļotas, un, atkarībā no to kārtas numura, saglabātas vai nu `RAM bitu` blokā `C` vai `D`.
Kad ir rotētas visas 512 nolases, un tās saglabātas `RAM bitu` blokos `C` un `D`, tad process atkārtojas, tikai šoreiz aprēķinu bloka ieejā tiek pieslēgti `RAM bitu` bloki `C` un `D`, bet izejā - `A` un `B`. Kad izpildītas visas 9 rotācijas, tad nu jau iegūtais izejas signāls Y ir *gatavais* biokods, kam var izpildīt sliekšņošanas operāciju.

Literatūras saraksts

[1] Wikipedia, *Rotation matrix*, apskatams - `http://en.wikipedia.org/wiki/Rotation_matrix`

[2] P.Misāns, *Ortogonālie pārveidojumi*, Melnraksts 2009.10.07.

[3] P. Misans, U. Derums, *Introduction Into the Novel Two-Dimensional Discrete Orthogonal Transforms Based on Rotation Angles*

[4] Uwe Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*

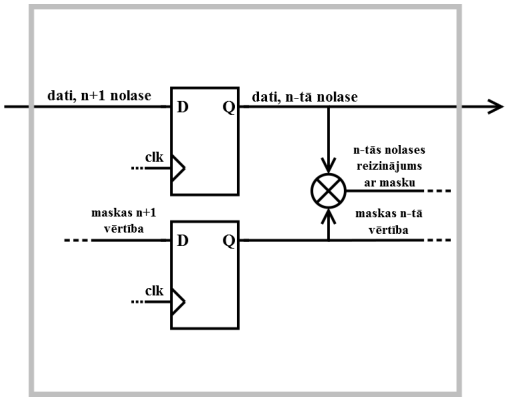
Aktivitāte: Eksperimentālā izstrāde

Algoritmu implementēšana eksperimentālajā maketā

Filtru implementācija `FPGA`

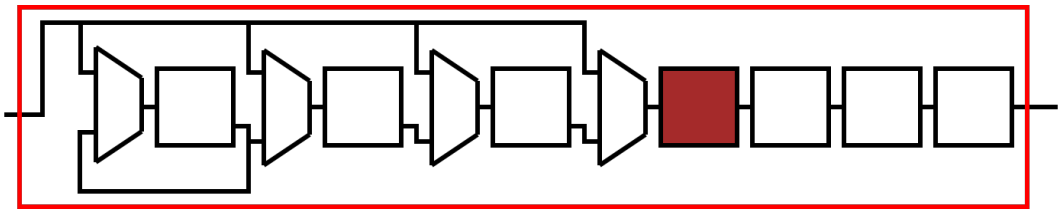
Horizontālais filtrs

Filtru dizainos, lai izveidotu parametrizējamus filtrus, kā mazākais filtru veidošanas elements izveidots vienas nolases filtrs (attēls 1.). Šī elementārā filtra sastāvdaļa veidota no diviem reģistriem --- viens paredzēts datu aizkavēšanai par vienu takti, otrs --- filtra maskas pikseļu vērtību glabāšanai, kā arī no reizinātāja, kas nepieciešams konvolūcijas veikšanai. Vienkāršības dēļ turpmākajos attēlos vienas nolases filtrs attēlots kā *kastīte*, kurai norādīta tikai datu ieeja --- (tiek ierakstīta signāla S_{N+1} nolase), un datu izeja --- tajā pieejama signāla S_N nolase, bet netiks apskatīti pārējie signāli, piem., takts signāls --- `clk`, ar filtra *masku* saistītie signāli, u.c.

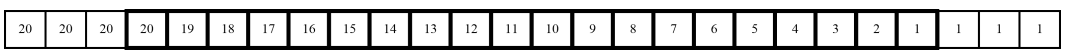


1. att. `Vienas nolases` filtra bokshēma

`Horizontālā` filtra blokshēma apskatāma 2. attēlā - šeit redzams piemērs nekauzālam horizontālajam filtram ar garumu 7 (tiek izmantoti 7 vienas nolases filtri), kas ir simetrisks (t.i. filtra maskas centra pozīcija (turpmāk --- ``filtra centrs'') ir 4). Šis parametrizējamais horizontālais filtrs veidots ar nosacījumu, ka jāveic attēla *papildināšana* ar attēla malējiem pikseļiem, nebojājot pašu attēlu. Piemērs šādai papildināšanai redzams attēlā 3. --- pieņemsim, ka attēlam platums (pikseļos) ir 20. Tā kā filtrs ir simetrisks un tā centrs ir 4, tad *mākslīgi* attēls ir jāpapildina ar 3 pikseļiem rindas sākumā, un 3 --- rindas beigās, lai, *novietojot* filtra masku attēla rindas galos, attēlam neizrūktu pikseļi, kas jāreizina ar maskas pikseļiem.

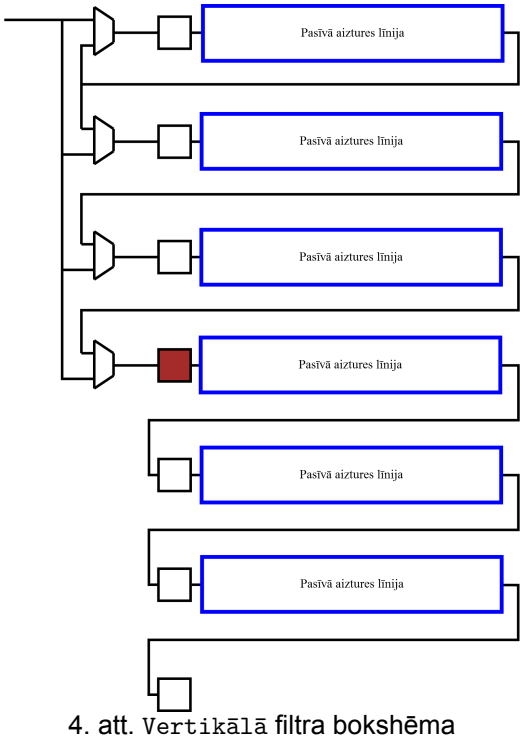


2. att. `Horizontālā` filtra bokshēma



3. att. Piemērs kadra rindas horizontālai *papildināšanai* --- tumši iezīmēta oriģinālā kadra rinda, bet gaišāk - mākslīgi radītie pikseļi, cipari apzīmē pikseļu kārtas numurus

Attēla papildināšana ar paša attēla malējiem pikseļiem nozīmē, ka rindas sākumā (kad ir pienācis rindas pikselis ar kārtas numuru 1) filtra daļa, kas atrodas pa *labi* no centrālā elementa (attēls 2.), ir jāaizpilda ar šo vērtību - `FPGA` programmējamajā loģikā horizontālais filtrs veidots tā, ka *sākoties katrai rindai* šī, pirmā vērtība tiek ierakstīta visos *vienas nolases filtros* uzreiz (t.i. multipleksori pārslēdzas ``uz augšu''). Brīdī, kad filtrā būs ierakstīts $4.$ pikselis, tad varam apgalvot, ka filtrs būs ``aizpildīts'', un tagad kontroles mehānisms varēs padot `start` signālu tālākajai ar masku sareizināto datu apstrādes loģikai. Jāpiemin, ka lai varētu notikt filtra ``aizpildīšana'', pirmajam multipleksoram jābūt pārslēgtam virzienā ``uz augšu'', bet pārējiem multipleksoriem - virzienā ``uz leju''.
Attēla rindas beigās, kad filtrā tiks ierakstīts rindas pēdējais - $20.$ pikselis, būs jāuzsāk rindas *beigu* papildināšana --- tagad pirmais multipleksors tiks ieslēgts virzienā uz leju - t.i. tas pirmajam *vienas nolases filtram* pados atpakaļ tā izejas vērtību (t.i. $20.$ pikseļa vērtību) `"filtra centrs - 1"` reizi. Šāda filtra papildināšanas implementācijas arhitektūra gan nozīmē to, ka maskas *vidus punktam* vienmēr jāatrodas filtra vidū, un pieprasa nosacījumu, ka filtra garums - nepāra skaitlis. Pašlaik gan tas nesagādā problēmas, jo nepieciešamajiem Gausa horizontālās izpludināšanas, Gausa vertikālās izpludināšanas un $NH - CMF2$ filtriem ir šādas maskas.



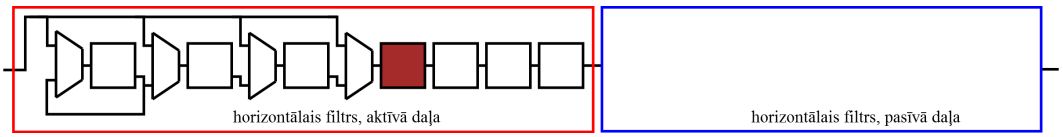
4. att. `Vertikālā` filtra bokshēma

Vertikālais filtrs

`FPGA` programmējamajā loģikā izveidots parametrizējams vertikāls filtrs, kas nodrošina attēla kadra papildināšanu ar paša attēlu --- 4. attēlā redzams šāds filtrs, ar garumu 7. Šāda filtra darbība ir tāda pati, kā `horizontālā` filtra gadījumā, tikai tagad ir jāpapildina nevis attēla rindas ar `(filtra centrs - 1)` pikseļiem, bet gan jāpapildina attēls ar `(filtra centrs - 1)` rindām.
Galvenā implementācijas atšķirība no `horizontālā` filtra ir tāda, ka tagad ir *jāseko* līdzi nevis attēla kadra rindām, bet gan paša kadra sākumam un beigām.

$2D$ parametrizējams filtrs

Tā kā projektā izveidotais $NH - CMF2$ ir $2D$ filtrs, tad tika veidots parametrizējams $2D$ filtrs, kas spēj veikt attēla papildināšanu. Lai apskatītu šādas implementācijas filtra darbības algoritmu, jau sadaļā *horizontālais filtrs* apskatīto filtru (attēls 2.), savieno ar pasīvo aiztures līniju - 5. attēls.



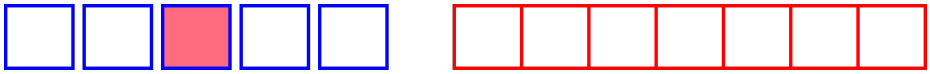
5. att. Horizontālā filtra virknes slēgums ar pasīvo aiztures līniju

Vienkāršības dēļ tālāk apskatīsim, ka filtra aktīvās daļas garums ir nevis $f = 7$, bet gan $f = 5$ pikseļi, kas nozīmē, ka maskas centra pozīcija, lai nodrošinātu simetriju, $c = 3$, bet

visas rindas garums ir $r = 10$ pikseļi. Tad pasīvās daļas garums aprēķināms pēc formulas:

$$r - f + (c - 1) = 10 - 5 + (3 - 1) = 7 \quad (1)$$

Vizuāls filtra aktīvās / pasīvās daļas attēlojums apskatāms 6. attēlā. Pieņemsim, ka 1. rindas



6. att. Filtra aktīvās daļas savienojums ar pasīvo daļu, zilie kvadrāti attēlo filtra aktīvās daļas, rozā kvadrāts – filtra centru, bet sarkanie kvadrāti – filtra pasīvo daļu

pikseļu vērtības ir sakārtotas secībā no labās puses uz kreiso:

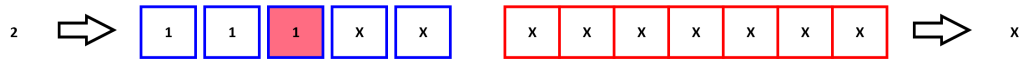
$$10,9,8,7,6,5,4,3,2,1$$

bet 2. rindas:

$$20,19,18,17,16,15,14,13,12,11$$

Apskatīsim dažus mums interesējošus momentus:

- Līdz šim filtrs (gan tā aktīvā, gan pasīvā daļa) aizpildīti ar *troksni* --- vērtībām, kuras mums neinteresē. Pienākot 1. rindas 1. pikselim, tas tiek ierakstīts filtrā, kā arī *nokopēts* vēl divas reizes --- līdz filtra *centram* --- 7. attēls.



7. att. Filtra rindas *aizpildījums* kadra rindas sākumā

- Dati tiek ievadīti filtrā, pienāk pēdējais 1. rindas pikselis --- 8. attēls.

8. att. Filtra rindas *aizpildījums* kadra rindas beigās

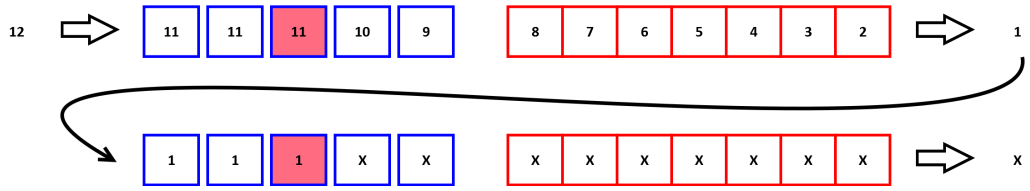
- Tiek ieslēgts ģenerators, kas attēla rindu *izvada* no filtra (tagad vairs ieejas pikseļi netiek lasīti, jo ir iestājies laika moments „starp rindām", kad nākošā attēla rinda no attēla sensora vēl nav pienākusi --- 9. attēls.

9. att. Filtra rindas *aizpildījums* kadra rindas beigās, ieslēdzoties ģeneratoram

- Pēdējā ieejas vērtība tiek nokopēta otro reizi - šajā gadījumā, kad $f = 5$, tā arī ir pēdējā reize, kad ir jāpapildina kadra rindas --- 10. attēls.

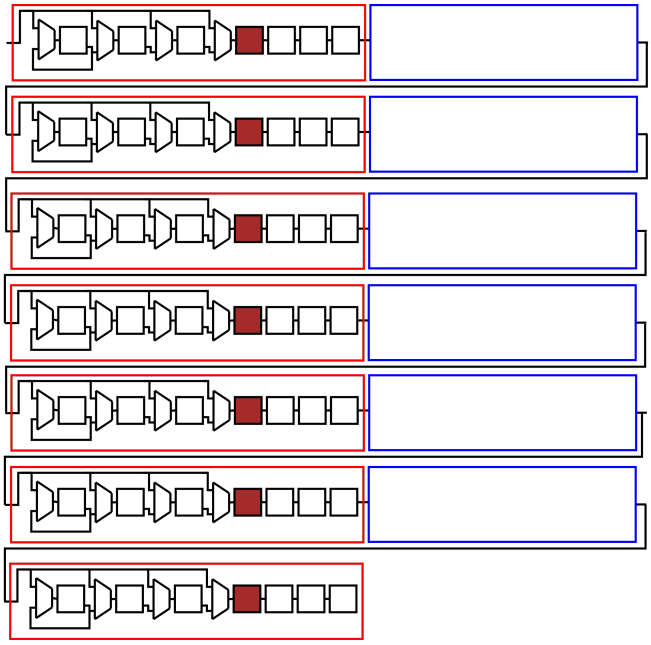
10. att. Filtra rindas *aizpildījums* kadra rindas beigās, strādājot ģeneratoram

- Pienāk otrās rindas 1. pikselis - vērtība 11 - šajā momentā no pasīvās aiztures līnijas tiek izbīdīta attēla 1. rindas 1. pikselis (vērtība 1), un iebīdīta nākošajā filtra rindā. Abu filtra rindu *individuālajos* filtros tiek veikta attēla rindu sākuma papildināšana - ieejas vērtība tiek *kopēta* līdz centram --- 11. attēls.



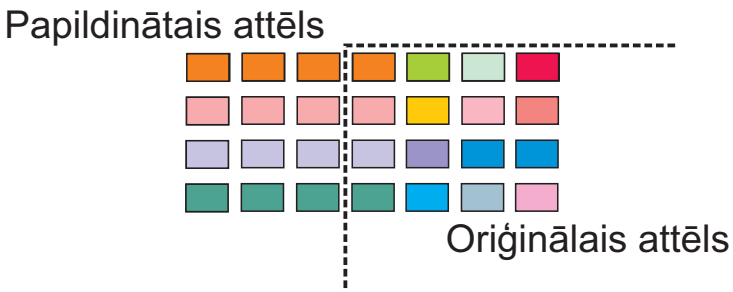
11. att. Filtra rindu *aizpildījums*, filtram saņemot 2. rindas 1. pikseli

Varam redzēt, ka izvēloties filtra pasīvās daļas garumu, kuru apraksta formula 1., attēls netiek sabojāts, veicot tā papildināšanu - *papildinātās* nolases *dzēšas*, ieslēdzot *ģeneratoru* un *izbīdot* attēlu no filtra. Tas nozīmē, ka šāda $2D$ filtra rinda neiespaido oriģinālo attēlu - tālāk attēls var tikt atkārtoti apstrādāts
Līdz šim esam apskatījuši tikai vienu no $2D$ filtra rindām (attēls 5.) - tā kā katra individuālā filtra rinda ar pasīvo aiztures līniju attēlu *nebojā*, tad varam tās savienot, piem., izveidojot $2D$ filtru ar aktīvās daļas izmēru - $7 \cdot 7$ pikseļi ---12. attēls (pēdējai rindai vairs nav nepieciešams pievienot aiztures līniju, jo nav nepieciešams saglabāt pēdējās filtra rindas izejas sinhronizāciju ar iepriekšējām filtra rindu izejām).



12. att. Filtra *rindu* savienojums, kas veido $2D$ filtru ar aktīvās daļas izmēru $7 \cdot 7$ pikseļi

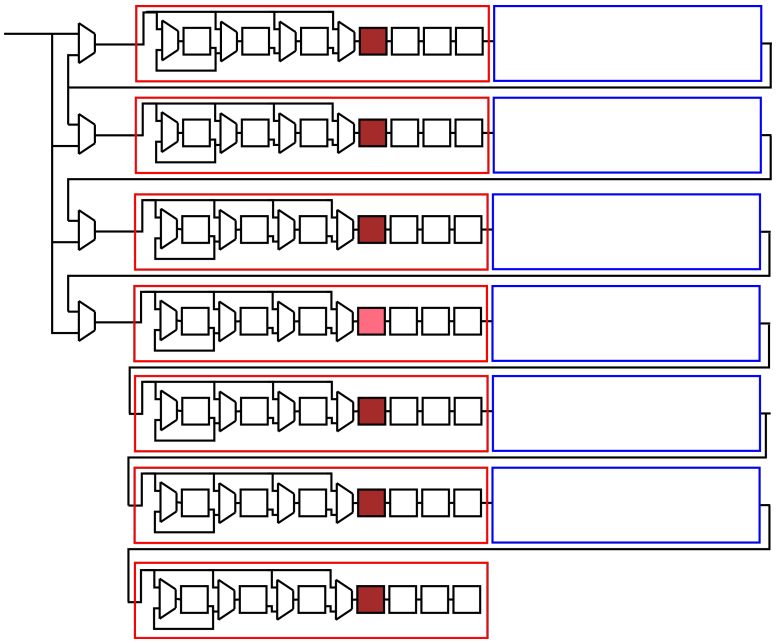
Redzams, ka apskatītais filtrs nodrošina attēla *papildināšanu* no malām, bet ne no attēla augšas un apakšas - 13. attēls.



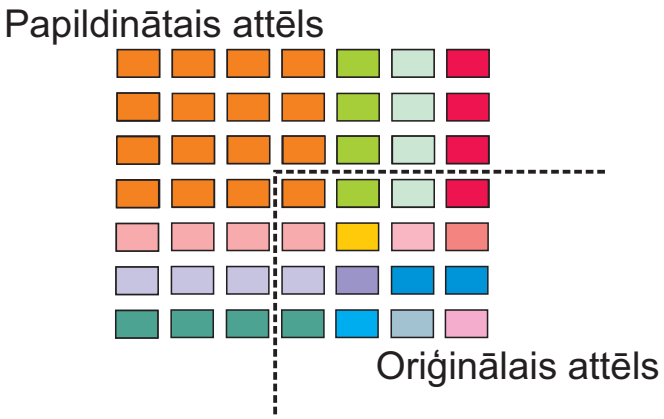
13. att. Attēla papildināšana, kādu nodrošina filtru slēgums, kas redzams 12. attēlā

Attēla papildināšanas jautājumu var risināt, individuālās filtra rindas *vadot* ar tādu pašu

mehānismu, kā tas tika darīts vertikālā filtra gadījumā --- 4. --- pilnīgi funkcionējošs $2D$ filtrs apskatāms 14. attēlā, bet šāda filtra attēla *papildināšanas* fragments redzams 15. attēlā.



14. att. *Pilnvērtīgs* $2D$ filtrs



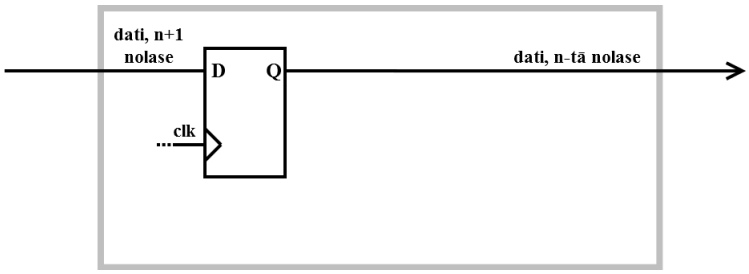
15. att. Attēla papildināšana, kādu nodrošina filtru slēgums, kas redzams 14. attēlā

NH-CMF2 filtrs

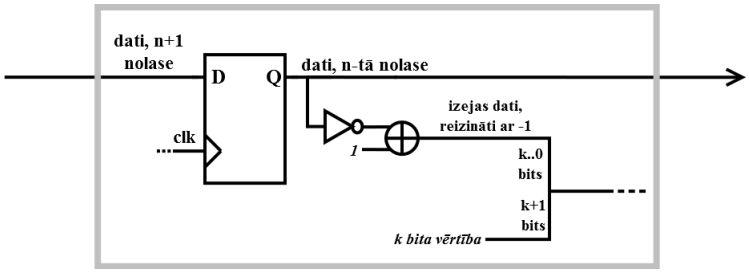
NH-CMF2 veidots tā, lai nebūtu nepieciešams izmantot reizinātājus. Tādēļ NH-CMF2 filtra implementācija veidota, izmantojot apskatīto $2D$ filtru, tikai tagad šis filtrs neizmantos vienas nolases filtrus (apskatīti iepriekš), bet tā vietā filtrs veidots tā, ka ar maskas parametru tabulas palīdzību --- (tabula 1.) iespējams izsaukt šādus filtra elementus:

- 0 konkrētā attēla pikseļa vērtības apstrāde nav nepieciešams, tiek implementēts reģistrs datu aizturei par vienu takti --- attēls 16.;
- 1 ne tikai oriģinālie dati tiek aizturēti par vienu takti, bet izejā padota arī pikseļa vērtība, reizināta ar -1 --- attēls 17.;
- 2 ne tikai oriģinālie dati tiek aizturēti par vienu takti, bet izejā padota arī pikseļa vērtība, reizināta ar 2 --- attēls 18.;

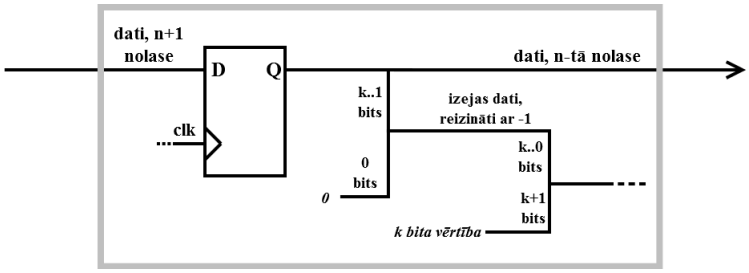
Filtrs arī izveidots tā, lai tas minimizētu izmantojamo reģistru skaitu - katras maskas rindas labās puses elementi, kuri nav nepieciešami attēla filtrācijai --- veic tikai pikseļu *aiztures* funkciju --- t.i. šādus elementus varam apzīmēt ar ``0'' --- netiek implementēti, izmantojot reģistrus, bet gan tiek pievienoti pasīvajai aiztures līnijai, kas izmanto RAM bitus --- tabulas 1. zaļā krāsā iekrāsotās rūtiņas --- šāda funkcionalitāte gan netiek piešķirta filtra pēdējai rindai, jo pēc tās neseko pasīvā aiztures līnija.



16. att. Implementēts reģistrs datu aizturei par vienu takti



17. att. Implementēts reģistrs datu aizturei par vienu takti, dati reizināti ar -1 un papildināti ar zīmes bitu



18. att. Implementēts reģistrs datu aizturei par vienu takti, dati reizināti ar 2 un papildināti ar zīmes bitu

1. tabula. `NH-CMF2` maskas parametri

0	0	0	0	0	0	1	0	0	2	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	2	0	0	0	1	0	2	0	1	0	0	0	2	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	2	0	0	0	0	0	0	0	2	0	0	0	0	0
1	0	0	0	0	0	0	0	1	2	1	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	2	0	0	0	2	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
2	0	0	2	0	0	2	0	0	0	0	0	2	0	0	2	0	0	2
0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	2	0	0	0	2	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	1	2	1	0	0	0	0	0	0	0	1
0	0	0	0	0	2	0	0	0	0	0	0	0	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	2	0	0	0	1	0	2	0	1	0	0	0	2	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	2	0	0	1	0	0	0	0	0	0

Secinājumi

Projekta „BiTe" pētnieciskā darbība „Lietišķo pētījumu" aktivitātē un eksperimentālā darbība „Eksperimentālās izstrādes" aktivitātē sekmīgi turpinās arī septītajā periodā. Noris izstrādājamās tehnoloģijas atsevišķu moduļu, algoritmu un elektrisko principiālo shēmu izpēte un izveide.

Sejas un plaukstas attēlu iegūšanas metožu izpētē turpmāk tiks veikti papildus pētījumi, lai novērtētu algoritma stabilitāti un iespēju izmantot to kā interesējošā reģiona izdalīšanas algoritmu.

Sejas atpazīšanas sistēmas izveidei tiek plānota funkcionālo bloku nepieciešamības novērtēšana automātiskajā sejas atpazīšanas procesā, ar mērķi izslēgt tos blokus, kas var pasliktināt atpazīšanas procesu un ievērojami samazināt tā ātrdarbību. Plānots arī papildināt izveidoto programmu, novēršot radušās kļūdas.

Biometrisko datu iegūšanas algoritmu paralelizācijas un implementēšanas programmējamos loģiskos masīvos aktivitātes ietvaros nākotnē paredzēts pabeigt un novērtēt FSM sliekšņa vērtības, ar FPGA testēt attēla histogrammas reālos apstākļos un plānots turpināt plaukstas ģeometrijas algoritma posmu realizāciju FPGA

Biometrisko datu kriptēšanas, glabāšanas un lietojama apakšsistēmas izveides aktivitātē turpmāk tiek plānots turpināt pētīt plaukstas detektēšanas iespējas.

Tehnoloģijas demonstratora eksperimentālā maketa montēšanas apakšaktivitātē maketa detalizēts izveides apraksts ir sniegts iepriekšējos progresa pārskatos. Turpmāk veiktie iekārtas papildinājumi un uzlabojumi tiks aprakstīti nākošajos progresa pārskatos.

Algoritmu implementēšana eksperimentālajā maketā turpināsies un ir paredzēts FPGA platformā implementēt BioHash algoritmu, kas iegūst un nosūta biokodu uz viedkarti salīdzināšanai.

Projekta pētnieciskie rezultāti tiek apspriesti izpildītāju iknedēļas sanāksmēs. Šīs sanāksmes tiek veidotas, lai projekta izpildītāji kopīgi izdiskutētu par pētījumiem un problēmām, kas jārisina, lai aktivitātes veiksmīgi attīstītos. Septītajā progresa pārskata periodā notika sanāksmes, kurās tika pārrunāti dažādi jautājumi saistībā ar projekta izpildi, svarīgākie no tiem: biometrijas iekārtas maketa uzlabošanas iespējas; parametrizējama filtra izstrāde uz 2D filtra bāzes; ātrā NH-CMF filtra realizācija FPGA; sejas detektēšanas algoritmu realizācija uz Raspberry Pi; plaukstas ģeometrisko izmēru algoritmu izveide, u.c. jautājumi.

Iepriekšējos periodos sagatavots zinātniskais raksts „Biohashing and Fusion of Palmprint and Palm Blood Vein Biometric Data", kas prezentēts starptautiskā IEEE organizētā konferencē (ICHB2011), Honkongā. Projekta izpildītāji piedalījās arī Starptautiskajā izgudrojumu izstādē „MINOX 2012", kurā ieguva „Dienas Biznesa" speciālbalvu. Septītajā pārskata periodā tika iesniegti divi zinātniskie raksti publicēšanai: „Palmprint Image Processing with NON-HALO Complex Matched Filters for Forensic Data Analysis" un „FPGA Implementation of CMF for Embedded Palm Biometric System"

Turpinās darbs aktivitātes „Intelektuālā īpašuma tiesību aizsargāšana" ietvaros, ir noslēgts līgums ar patentpilnvaroto SIA „PĒTERSONA PATENTS" un iesniegts starptautiska patenta pieteikums. Šobrīd tiek gatavoti papildus skaidrojumi.

Ir noteikti nākamā perioda pētniecisko darbu uzdevumi un sasniedzamie rezultāti. Par projekta pētnieciskiem sasniegumiem tiek plānots informēt arī piedaloties ar referātiem un publikācijām konferencēs un semināros.