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.6. par periodu no 01.07.2012.g. līdz 31.10.2012.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
2012. 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 sestajā periodā (01.07.2012.- 31.10.2012) 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 aktivitātes „Lietišķie (rūpnieciskie) pētījumi" pētījumos:

- Sejas un plaukstas attēlu iegūšanas metožu izpēte;
- Biometrisko datu apstrādes algoritmu darbības izvērtēšana;
- Biometrisko datu kriptēšanas, glabāšanas un lietojama apakšsistēmas izveide;
- Atsevišķu sistēmas komponenšu un bloku izstrāde un izpēte.

Turpinās darbi aktivitātes „Eksperimentālā izstrāde" ietvaros:

- Multimodālas biometrijas tehnoloģijas koncepta definēšana;
- Tehnoloģijas demonstratora eksperimentālā maketa montēšana;
- Algoritmu implementēšana eksperimentālajā maketā.

Rezultātu kopsavilkums

Projekta sestajā periodā (01.07.2012.-31.10.2012.) veikto pētniecības uzdevumu un 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; uzsākts 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 apskatīts jauns sākuma punkta noteikšanas algoritms, kas tiek izmantots robežu pikseļu vektora konstruēšanai un turpinās darbs pie algoritmu izstrādes MATLAB vidē.

- Biometrisko datu apstrādes algoritmu darbības izvērtēšana – Iepriekšējos periodos tika realizēta 1. versija 2D sejas atpazīšanas algoritmam (sejas detektēšana, lokalizācija un atpazīšana), kas implementēts iegultā sistēmā; veidots automātisks sejas atpazīšanas algoritms signālprocesoram TMS320C6416; pievienoti galvenie elementi kameras modulim; izveidotas programmas attēla vizualizācijai un kvalitātes novērtēšanai; un analizētas sejas atpazīšanas algoritmu implementācijas iespējas. Sestajā pārskata periodā apskatīti trīs risinājumi sejas atpazīšanas izstrādei iegultās sistēmās un izpētīta TMDXEVM6678LE attēla apstrādes testa programma.
- 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 (konvolūcijas veikšanai FPGA platformā); veikta attēla filtrēšana reālā laikā; izveidota jauna sistēmas (125MHz) arhitektūra; izpildīta jaunā Aptina MT9V024 attēlu sensora montāža un apskatīta datora saskarne ar prototipu. Šajā periodā tika izpētīta un izveidota attēlu filtrācija FPGA platformā; apskatīti varianti kā samazināt dubulto leņķi un uzsākta LCD kontroliera izveide uz FPGA platformas.
- 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; apskatīta Biohash algoritma papildus informācija, atpazīšanas uzlabošanai un izveidota viedkartes saskarne. Sestajā periodā ir izveidota viedkartes programma, kurā ir iespējams ierakstīt cilvēka datus.
- 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 2. versijas montāža; tika programmēti Aptina MT9V024 attēlu sensora reģistri; projektēts biometrijas sistēmas ietvars; izveidota iespiedplate viedkaršu saskarnes nodrošināšanai ar FPGA izstrādes rīku; programmēts mikrokontrolieris MSP430 un realizēti LBP, LDP algoritmi MATLAB vidē. Šajā pārskata periodā ir veikta fāzes korelācijas algoritma izpēte un datorprogrammas izveide šīs metodes pārbaudei ar plaukstas biometrijas datiem.

Aktivitātes „Eksperimentālā izstrāde" rezultātu kopsavilkums ir sekojošs:

- Multimodālas biometrijas tehnoloģijas koncepta definēšana – iepriekšējā periodā tika aprakstīts prototipa funkcionālo bloku darbības princips 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, izmantojot esošo prototipu un apstrādāti attēli tajā.
- Algoritmu implementēšana eksperimentālajā maketā – iepriekšējā pārskata periodā veiktas CMF modifikācijas; nomainīts maskas izmērs no 9x9 uz 15 x15; izmainīta pati maska un gūts priekšstats par VHDL parametrizējamām ķēdēm. Šajā pārskata periodā ir uzsākta Fast NH-CMF2 realizācija (vertikālās izpludināšanas filtra, horizontālās izpludināšanas filtra un NH-CMF2 aproksimācijas filtru realizācija).

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

Aktivitāte: Lietišķie pētījumi

Sejas un plaukstas attēlu iegūšanas metožu izpēte

Atskaites periodā, daudzu eksperimentu rezultātā tika konstatēts, ka nepieciešams mainīt sākuma punkta – SP noteikšanas algoritmu, kas tiek izmantots, robežas pikseļu vektora (BPV – Bourder Pixels Vector) konstruēšanai. Tas saistīts ar to, ka agrāk piedāvātais algoritms SP koordinašu noteikšanai izmantoja tikai plaukstas locītavu. Apģērbs vai citi priekšmeti radīja kropļojumus iepriekšējā algoritma darbības rezultātos, kas šādos gadījumos deva nepieņemamus rezultātus. Tāpēc tika izstrādāts jauns algoritms, kas izmanto dažu plaukstas parametru ģeometriskās sakarības.

Izskatīsim jaunā SP koordinašu noteikšanas algoritma darbību pa etapiem. Kā sākotnējie dati mums ir plaukstas attēls:

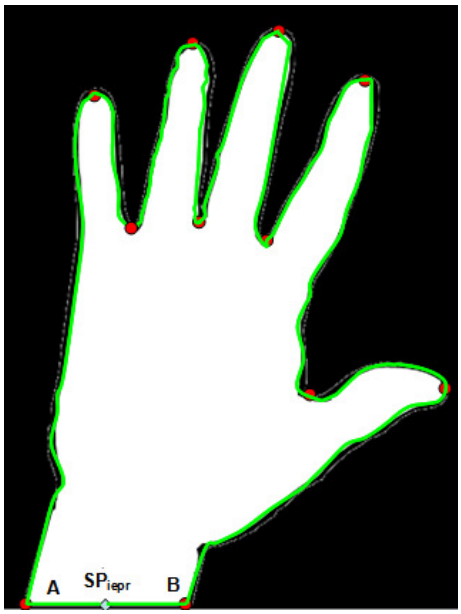
Pirmais etaps: attēla iepriekšējā apstrāde.

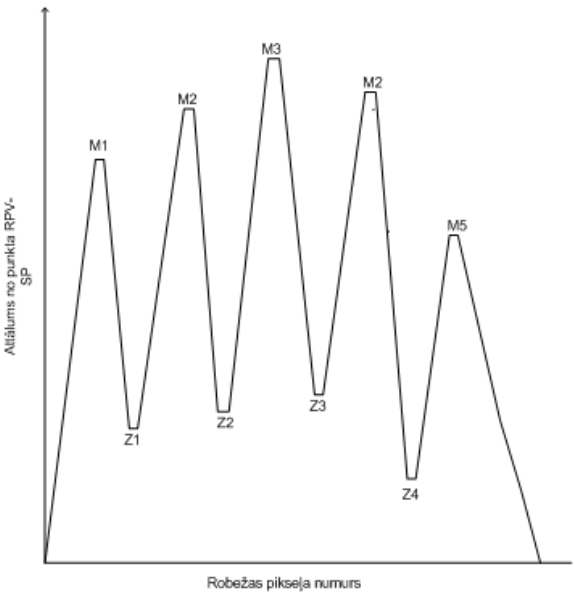
Vispirms, ja nepieciešams, attēls tiek pārveidots no krāsaina uz melnbaltu jeb uz pelēkās skalas formātu (gray scale). Tad, izmantojot speciālus filtrus, tiek uzlabota attēla kvalitāte (kontrasts, asums, utt.), bet pēc attēla histogrammas tiek noteikts binarizācijas slieksnis un veikta attēla binarizācija:

Otrais etaps: primārā BVP konstruēšana.

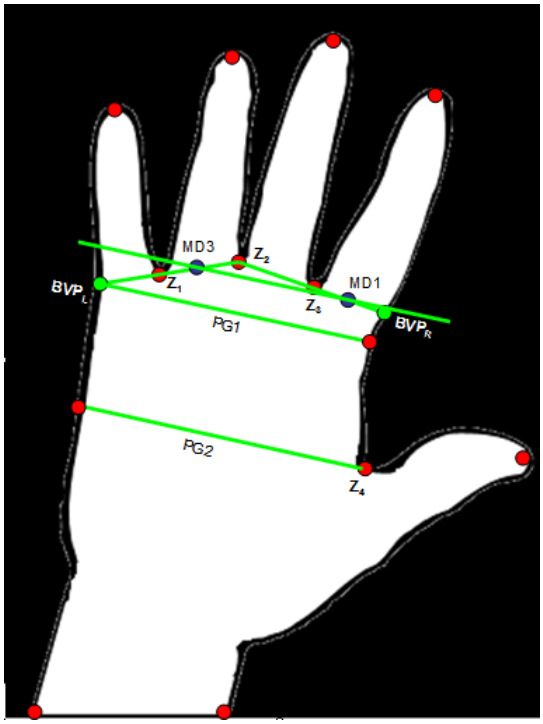
Primārais BVP tiek konstruēts sekojoši: vispirms nosakām nogriežņa AB koordinātes, kurš veidojas plaukstas attēlam šķērsojot attēla apakšējo robežu (apakšējais attēls), tad nosakām nogriežņa AB viduspunktu SP_{iepr} , kas būs sākumpunkts primārā BVP_{iepr} konstruēšanai.

BVP konstruēšanas procedūra aprakstīta iepriekšējās atskaitēs. Rezultātā iegūstam robežas pikseļu kopu, kas veido BVP. Shematiski BVP_{iepr} pikseļi parādīti zaļā krāsā. Blakus attēlā parādīta šo pikseļu diagramma, kurā uz abscisas ass atlikts BVP_{iepr} pikseļa numurs, bet uz ordinātas ass tā attālums līdz sākumpunktam SP_{iepr}. Pikseļu numerācija sākas ar SP_{iepr} un pakāpeniski palielinās apliecot plaukstas robežu pa kreisi no SP_{iepr}. Maksimumi M_i un minimumi Z_i atbilst pikseļiem, kas atrodas pirkstu virsotnēs un starp pirkstiem.





Trešais etaps: plaukstas platuma mērīšanas vietas atrašana un plaukstas platuma mērīšana. Plaukstas mērīšanas pirmā punkta atrašanai, vispirms novelkam līniju, kas iet caur pikseļiem Z_2 un Z_3 līdz tā krusto plaukstas robežu, precīzāk, ar vienu no BVP_{iepr} pikseļiem, kas atrodas pa labi no pikseļa Z_3. Apzīmēsim šo pikseli kā BVP_R. Tālāk nosakām nogriežņa Z_3 - BVP_R viduspunktu. Šī nogriežņa viduspunktu apzīmēsim ar MD_1. Pēc tam nosakām punkta MD_3 koordinātes. Novelkam līniju caur punktiem Z_1 un Z_2 līdz tā šķērso plaukstas robežu pa kreisi no pikseļa Z_1. Pikseli, kurā šī līnija krustojās ar plaukstas robežu, apzīmēsim ar BVP_L. Punkts MD_3 nosakāms kā nogriežņa Z_1 - Z_2 viduspunkts. Visbeidzot novelkam nepieciešamo līniju starp punktiem MD_1 un MD_3. Plaukstas platums tiks mērīts paralēli iegūtajai līnijai. Kā atskaites punktu plaukstas platuma mērīšanai piedāvājam izmantot punktu BVP_L :

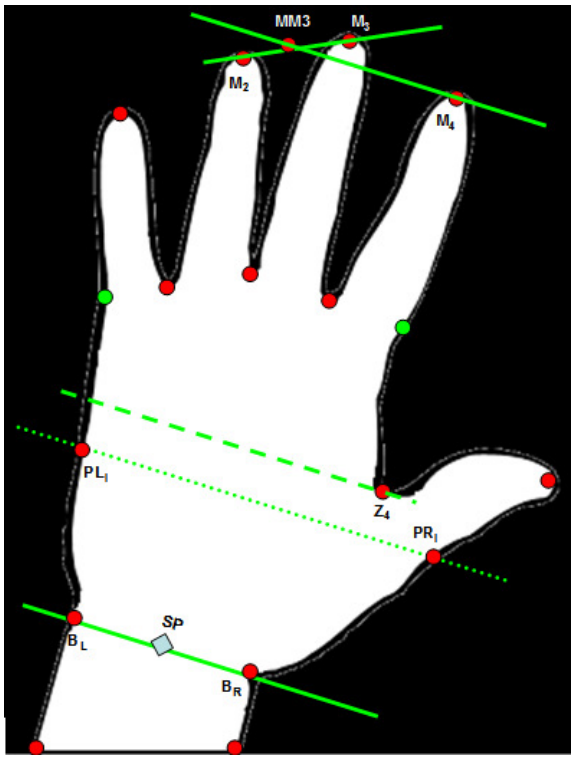


Plaukstas platuma mērīšanai novelkam līniju, paralēli līnijai, kas iet caur punktiem MD_1 un MD_3 un kas sākās punktā BVP_L. Šajā vietā izmērīto plaukstas platumu apzīmēsim ar PG1. Vēl vienu plaukstas platuma mērījumu veiksim punkta Z_4 līmenī. Šim nolūkam novilksim līniju, paralēli līnijai, kas iet caur punktiem MD_1 un MD_3 un kas sākās punktā Z_4. Šajā vietā izmērīto plaukstas platumu apzīmēsim ar PG2.

Ceturtais etaps: īstā sākuma punkta SP (start point) koordinātu noteikšana, īstā BVP konstruēšanai un tālāko plaukstas mērījumu veikšana.

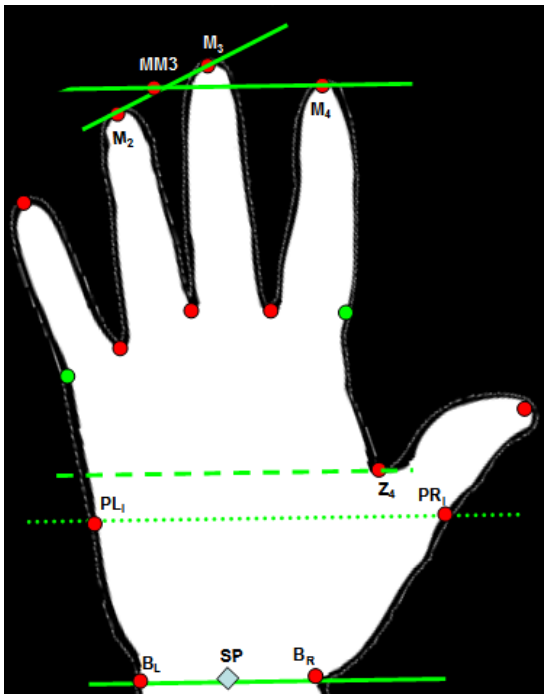
Novilksim līniju, kas iet caur punktiem M_2 un M_3, noteiksim nogriežņa M_2 - M_3 viduspunktu un apzīmēsim to ar MM_3. Tad novilksim līniju, kas iet caur punktiem MM_3 un M_4. Novilksim paralēli šai līnijai (MM_3 - M_4) līniju, kas šķērso punktu Z_4 apakšējā attēlā apzīmēta ar raustītu līniju. Novirzām šo līniju N pikseļus (N - parametrs) uz leju (paralēli sev) un mēram nogriežņa garumu, ko veido šīs līnijas krustpunkti ar kreiso un labo BVP robežu. Šie punkti i-tajai iterācijai apzīmēti ar PL_i un PR_i. Nomērām un saglabājam nogriežņa PL_i – PR_i garumu. Atkārtojam ciklu, kamēr kārtējā līnija (paralēlā MM_3 - M_4 līnijai) nav sasniegusi attēla apakšējo

malu. No visiem iegūtajiem nogriežņu garumiem, izvēlamies minimālo (attēlā tas apzīmēts B_L - B_R). Nogriežņa B_L - B_R viduspunkts SP tiek pieņemts par īsto sākuma punktu lai konstruētu īsto BVP. Nogriežņa B_L - B_R garumu apzīmēsim ar PG3



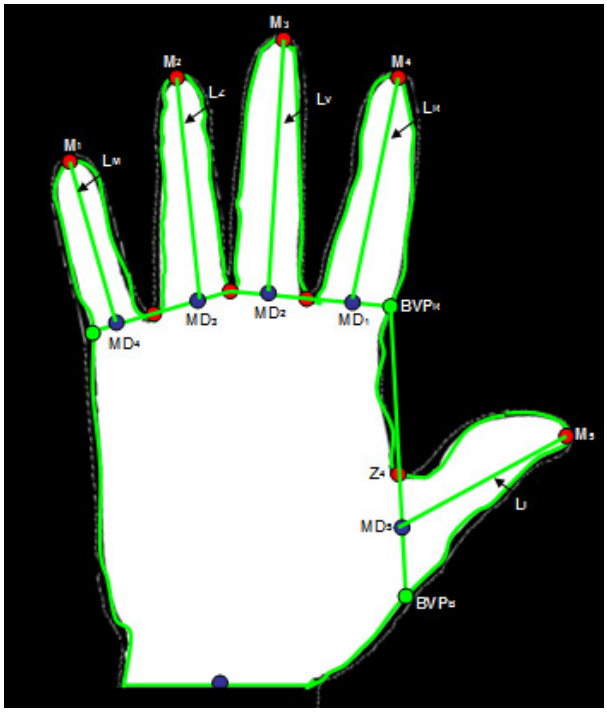
Piektais etaps: Pirkstu garuma noteikšana.

Tālākās analīzes atvieglošanai, plaukstas attēlu pagriežam pretēji pulksteņa rādītāja kustības virzienam tā, lai līnija, kas iet caur punktiem B_L un B_R kļūtu horizontāla:



Sākot no punkta SP, tiek konstruēts īstais BVP un noteiktas vairāki plaukstas ģeometriskie raksturlielumi.

Noteiksim vairākus palīgpunktus, kas nepieciešami pirkstu garumu noteikšanai. Punktus MD_1 un MD_3 mēs noteicām iepriekš. Punkts MD_2 tiek noteikts, kā nogriežņa $Z_2 – Z_3$, viduspunkts, bet punkts MD_4 kā nogriežņa $Z_1 – BVP_L$ viduspunkts. Mazā pirkstiņa garums tiek noteikts, kā attālums starp punktiem M_1 un MD_4. Apzīmēsim to ar L_M:



Zeltneša garums tiek noteikts kā attālums starp punktiem M_2 un MD_3. Apzīmēsim to ar L_Z. Vidējā pirksta garums tiek noteikts kā attālums starp punktiem M_3 un MD_2. Apzīmēsim to ar L_V. Visbeidzot, rādītājpirksta garums tiek noteikts, kā attālums starp punktiem M_4 un MD_1. Apzīmēsim to ar L_R. Lai noteiktu īkšķa garumu, vispirms nepieciešams novilkt līniju, kas iet caur punktiem BVP_R un Z_4 līdz tā krusto BVP vektora robežu (punkts BVP_B), bet pēc tam atrast nogriežņa Z_4 - BVP_B viduspunktu. Apzīmēsim šo punktu ar MD_5 un savienosim ar punktu M_5. Nogriežņa $MD_5 – M_5$ garums tiek pieņemts par lielā pirksta garumu - L_I. Sāksim formēt plaukstas īpašību vektoru PIV.

Šajā etapā PIV var tikt izteikts sekojoši:

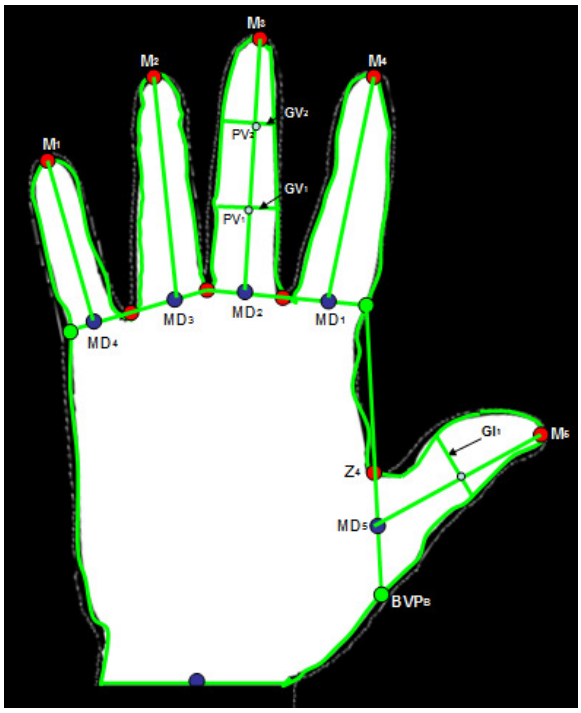
$$PIV = \{PG1, PG2, PG3, L_M, L_Z, L_V, L_R, L_I\}$$

Atzīmēsim, ka piedāvātās plaukstas raksturlielumu noteikšanas metodes neparedz nekādu patoloģiju esamību (piemēram, mazāku pirkstu skaitu vai to nedabīgu garumu). Turpmāk paredzēts izstrādāt šo metožu modifikācijas, kas spētu apstrādāt vairākas iespējamās patoloģijas.

Sestais etaps: pirkstu platuma noteikšana..

Četru pirkstu platums (izņemot īkšķi) mērāms divās vietās, īkšķa platums vienā vietā. Lai nomērītu pirksta platumu, nepieciešams noteikt mērījuma vietu. Piedāvājam sekojošu metodiku. Nogrieznis, kas nosaka pirksta garumu tiek sadalīts trīs vienādās daļās (pirmajiem četriem pirkstiem), bet nogrieznis, kas nosaka īkšķa garumu tiek sadalīts divās vienādās daļās. Piemēram, punkti LV_1 un LV_2 uz vidējā pirksta. Caur šiem punktiem tiek vilkti perpendikuli pret līniju MD_2 - M_3, kas šķērso pirksta robežu (precīzāk, krustojās ar

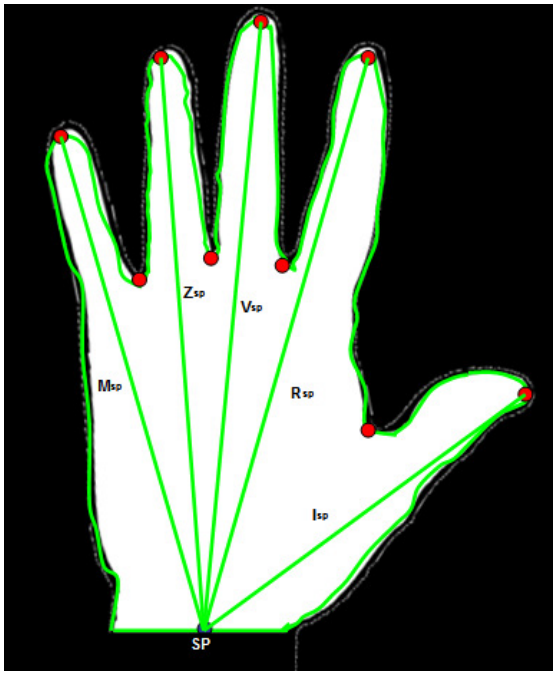
attiecīgajiem BVP pikseļiem) pa labi un pa kreisi. Iegūtā nogriežņa garums tiek pieņemts par attiecīgā pirksta platumu, punktā V_1, nomērīto apzīmējam ar GV_1, bet punktā V_2 nomērīto apzīmējam ar GV_2 :



Analoģiski atrodam arī citu pirkstu platumu: GM_1, GM_2, GZ_1, GZ_2, GR_1 un GR_2. Īkšķa platumu atrodam tikai vienā vietā. Noteikšanas punkts atrodas nogriežņa MD_5 - M_5 centrā. Īkšķa platumu apzīmējam ar GI_1. Pēc šiem mērījumiem plaukstas īpašību vektors izsakāms šādi::

PIV = {PG1, PG2, PG3, L_M, L_Z, L_V, L_R, L_I, GM_1, GM_2, GZ_1, GZ_2, GV_1, GV_2, GR_1, GR_2, GI_1}

Pēdējā mērījumu grupa nosaka attālumu no sākuma punkta (SP) līdz pirkstu virsotnēm:



Šo izmēru apzīmējumi ir sekojoši: M_{SP}, Z_{SP}, V_{SP}, R_{SP}, I_{SP}.

Pēc šīm izmaiņām PIV izskatīsies šādi:

PIV = {PG1, PG2, PG3, L_M, L_Z, L_V, L_R, L_I, GM_1, GM_2, GZ_1, GZ_2, GV_1, GV_2, GR_1, GR_2, GI_1, M_{SP}, Z_{SP}, V_{SP}, R_{SP}, I_{SP}}

Turpmākie plāni:

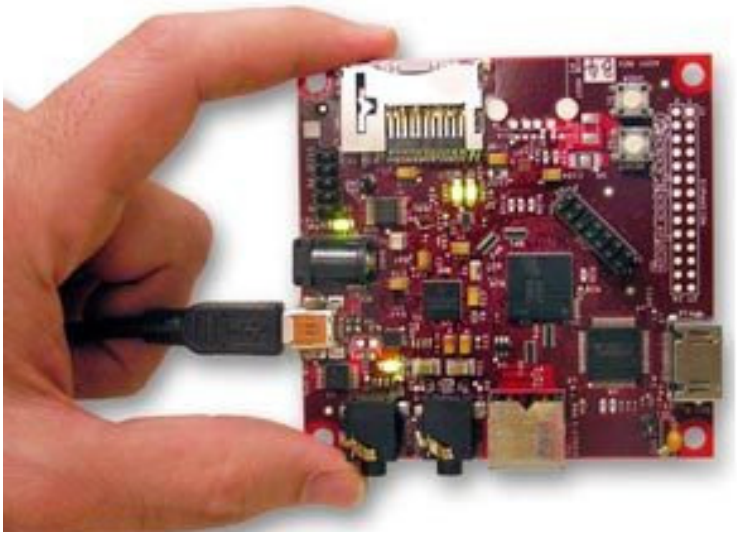
- Izstrādāt jaunus algoritmus plaukstas īpašību noteikšanai
- Sākuma punkta noteikšanas algoritma izstrāde MATLAB vidē;
- Plaukstas ģeometrisko izmēru metodes aprakstīšana;
- Algoritma izstrāde un eksperimenti ar esošo datubāzi algoritma darbības pārbaudei, uzlabošanai.

Biometrisko datu apstrādes algoritmu darbības izvērtēšana

Ir paredzēts izstrādāt sejas atpazīšanu uz iegultas sistēmas. Tika apskatīti vairāki risinājumi un uz doto mirkli ir izvēlēts risinājums, kuru ir visērtāk izstrādāt un tas sniegtu ērtāku piekļuvi resursiem. Trīs apskatītām iekārtām ir ARM procesors. Kā pirmais solis ir izvēlēts web kameras pieslēgšana pie plates un kādu gatavu algoritmu palaišana sejas detektēšanai (OpenCV bibliotēka). Tika apskatīta arī TMDXEVM6678L attēla apstrādes testa programma.

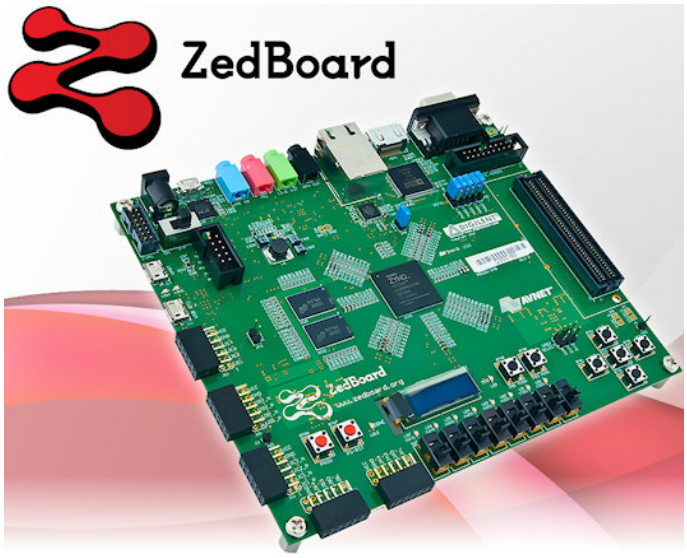
Beagle board iekārta

Pirmā apskatītā iekārta ir Beagle board revīzija Bx plate (skatīt sekojošajā attēlā). Procesors ir 600 MHz ARM8. Platei tika pārprogrammēta flash atmiņa, un uz SD kartes tika uzstādīts Ubuntu. Platei nav ethernet porta un nav arī wi-fi kartes. Līdz ar to linux'a konfigurēšana un vajadzīgo pakotņu uzstādīšana ir problemātiska.



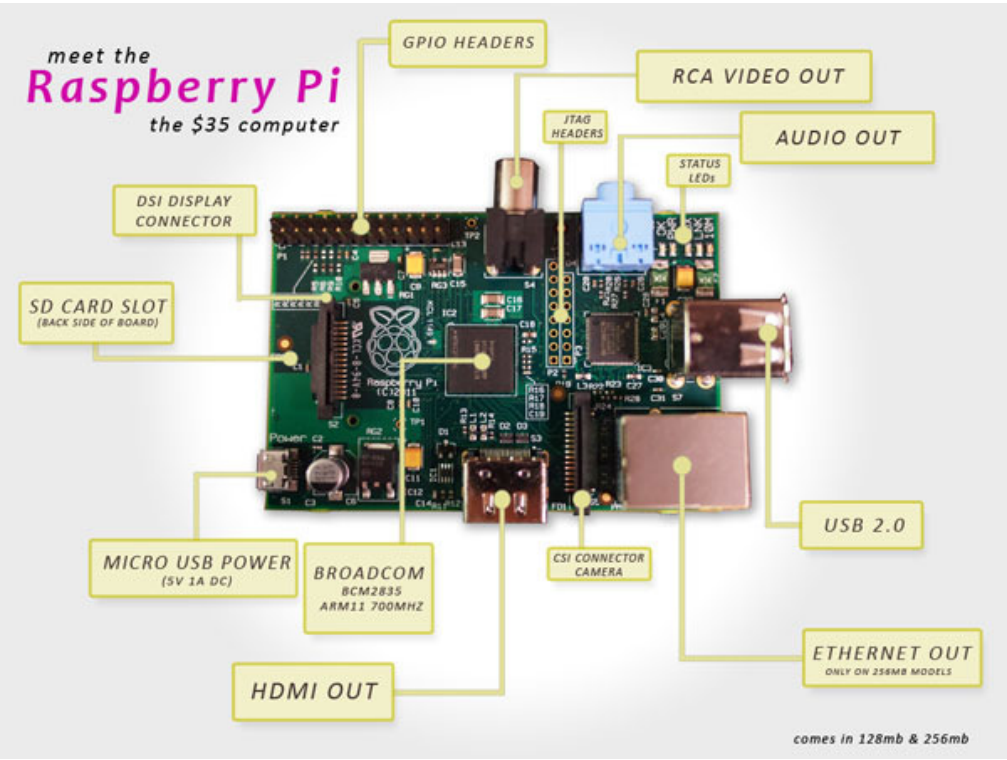
Zed Board izstrādes rīks

Otrā apskatīta iekārta ir izstrādes rīks Zed Board (skatīt sekojošajā attēlā), kuram uz čipa ir gan FPGA, gan ARM procesors. Iekārtu ir nepieciešams konfigurēt caur Xilinx izstrādes rīkiem, kādi porti tiek kādā veidā savienoti ar FPGA. Uz iekārtas ir iespējams nokompilēt un palaist C kodu, bet ne C++. Ubuntu ir iespējams uzlikt uz SD kartes un daļu no FPGA izmantot video kartei, kas uzreiz samazina pieejamos resursus. Arī visiem pārējiem portiem būtu nepieciešami kontrolieri, kas tiktu veidoti uz FPGA.



Raspberry Pi iekārta

Trešā apskatītā iekārta ir Raspberry Pi. 700 MHz ARM11 procesors. Uz viņas bez problēmām tika uzstādīta specializēta Debian distribūcija. Kartei ir ethernet ports, kas ļauj bez problēmas piekļūt internetam. Elementāri tika uzstādīts OpenCV un notestētas testa programmas. Tika arī iegūti attēli no web kameras.



Tika uzsākta MATLAB koda portēšana. Kā pirmā daļa tika pārrakstīta LBP histogrammas aprēķināšana. Lai to būtu iespējams notestēt, kods tika veidots tā, lai būtu iespējams izsaukt funkciju no MATLAB vides.

TMDXEVM6678L attēla apstrādes testa programma

Testa programmas darbība ir saņemt attēlu no personālā datora; saņemt norādes, cik kodolus izmantot; izmantojot *sobel* operatoru, apstrādāt attēlu; atgriezt attēlu uz personālā datora. Apstrādē patērētais laiks ir atkarīgs no kodolu skaita, kas tiek izmantoti. Apstrādājot 16 MB lielu attēlu un izmantojot 8 kodolus apstrādes laiku var samazināt 6 reizes, salīdzinot pret 1 kodolu.

Šajā programmā tiek izmantota attēla pārsūtīšana uz TMDXEVM6678LE izmantojot THCP un HTTP protokolus. Šie protokoli ir izveidoti izmantojot Texas Instruments NDK piedāvātos risinājumus, kā arī EMAC. Lai uzlabotu sistēmas ātrdarbību, attēls tiek glabāts DDR ārējā atmiņā un tālākās procedūras var tikt sadalītas pa kodoliem. Komunikācijas izveidei starp kodoliem tiek izmantots IPC protokols, ar kuru tiek kontrolēta attēla apstrāde.

Izveidots pagaidu algoritms, lai varētu novērot sistēmas darbību uz vairākiem kodoliem. Attēlam tiek pielietots *sobel* operators un pēc tam sliekšņošana. Kad apstrāde ir pabeigta, galvenais kodols gaida atbildes rindu no pārējiem un tiek salikta bilde. Attēls tiek pārsūtīts atpakaļ uz personālo datoru. Vienkāršoti algoritms ir attēlots attēlā.

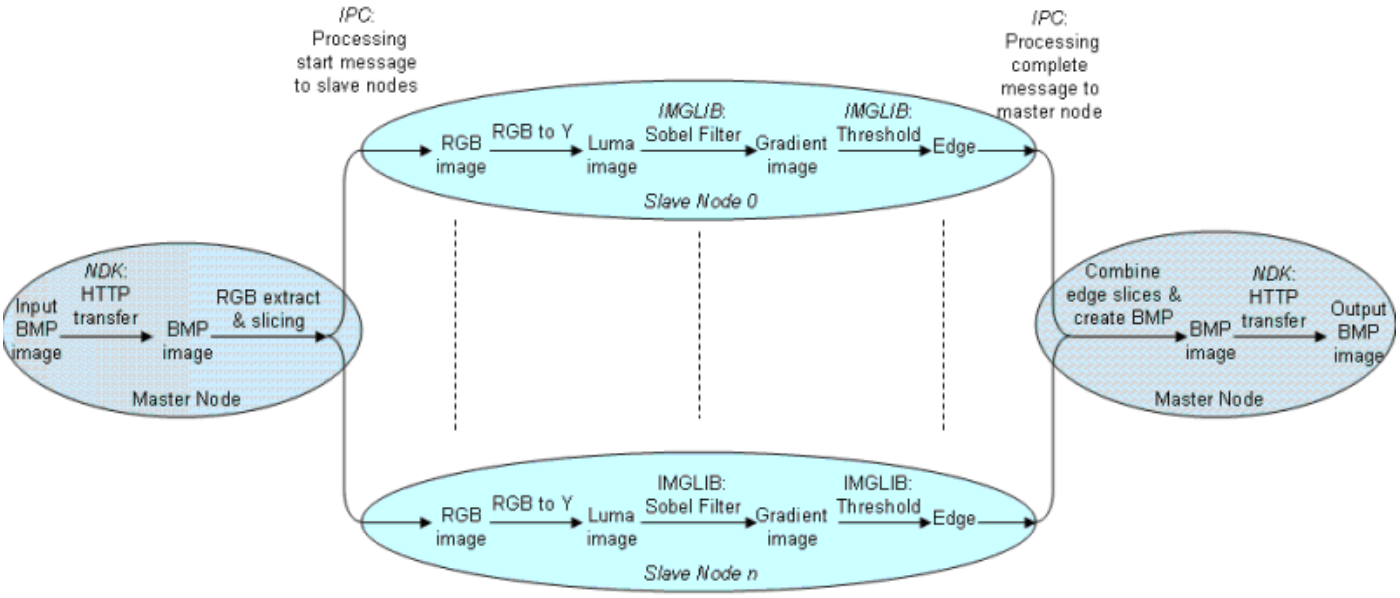


Image Processing Demonstration Pipeline

Turpmākie plāni:

- MATLAB koda pārveidošana uz C++ kodu, lai programma var darboties patstāvīgi bez MATLAB;
- Programmas ieviešana iegultā sistēmā;
- Cilvēka attēla iegūšana no kameras un pārbaudes testi;
- TMDXEVM6678L attēla apstrādes programmas pielietošana sejas atpazīšanas algoritmu realizācijā.

Biometrisko datu iegūšanas algoritmu paralelizācija un implementēšana programmējamos loģiskos masīvos

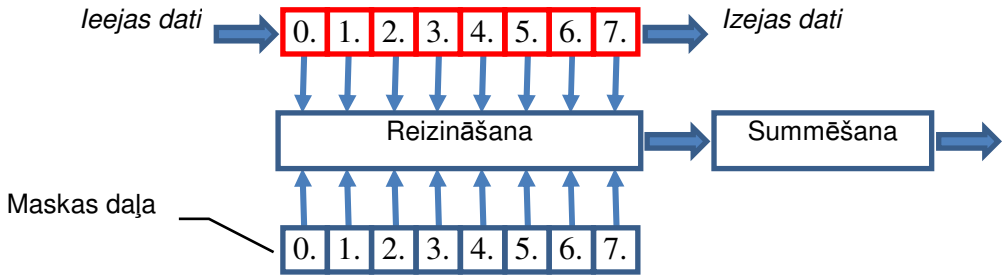
Attēlu filtrācija FPGA platformā

Filtrācijas veikšanai reālā laikā FPGA platformā filtram ir nepieciešams nodrošināt iespēju piekļūt pie vesela ieejas attēla fragmenta ar tikpat lielu ātrumu kā nāk katrs ieejas attēla pikselis. Šo funkcionalitāti nav iespējams realizēt, izmantojot ārējās RAM mikroshēmās, bet ir iespējams nodrošināt ar FPGA iekšējiem resursiem, īslaicīgi glabājot vajadzīgos attēla fragmentus un filtrējot datus attēla ieguves laikā.

FPGA 2D filtri tiek veidoti no t.s. aktīvām un pasīvām daļām (abi ir bīdes reģistri, bet ar dažādu fizisko realizāciju), slēdzot tos virknē. Ieejas dati tiek pārvadīti caur šiem bīdes reģistriem virknes formā, tādejādi tiek ģenerēti izejas dati, kas arī ir virknes formā. Process var norisinies t.s. reālā laikā, jeb ar maksimālu frekvenci, ar kuru spēj strādāt attēlu sensors, bez ārējas atmiņas piesaistīšanas.

1. Filtra daļas

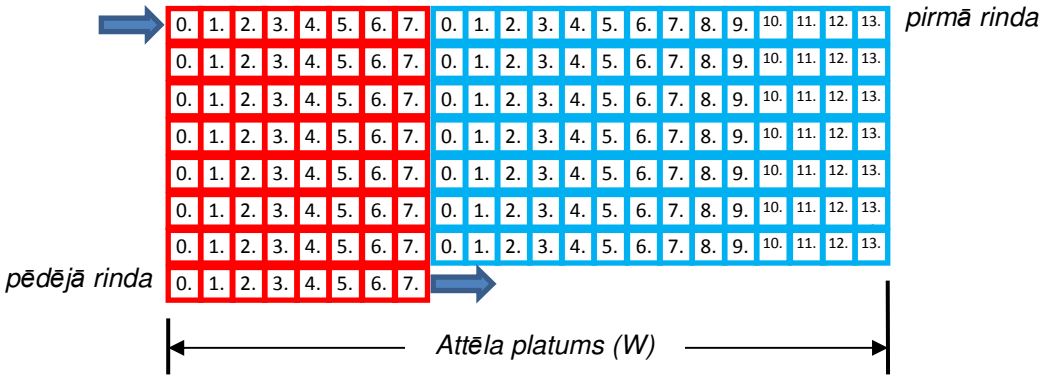
Filtra aktīvās daļas ir bīdes reģistri, kas tiek realizēti ar FPGA reģistriem, neizmantojot RAM bitus. Šādi veidojot katru filtra aktīvo daļu tiek nodrošināta vienlaicīga piekļuve pie visiem šajā aktīvajā daļā saglabātajiem pikseļiem. Filtra aktīva daļa ir tā, kas veic filtrāciju, un, tāpēc, tā tiek vienmēr integrēta kopā ar atmiņu, kurā glabā filtra masku kā redzams sekojošajā attēlā (punkts pie numura norāda uz pozīciju bīdes reģistrā, attēlā – aktīvas daļas, kā arī maskas garums ir 8; sarkanā krāsa turpmāk norādīs uz aktīvās daļas atmiņas šūnām).



Filtra pasīvās daļas ir bīdes reģistri, kas tiek realizēti ar FPGA RAM bitu palīdzību. Tie tiek izmantoti tikai pikseļu glabāšanai, bet netiek izmantoti datu apstrādei.

Zilā krāsa turpmāk norādīs uz pasīvās daļas atmiņas šūnām; attēlā pasīvās daļas garums ir 14.

Filtra daļas tiek slēgtas kaskādē, veidojot dažādas atmiņas šūnu kombinācijas. Šīs šūnu kombinācijas tiek attēlotas telpiski, atkarībā no tā, kādu attēla struktūru tās glabā:



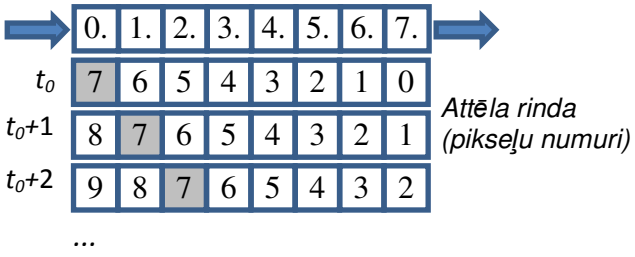
Šādas atmiņu kombinācijas slēdz struktūrās ar platumu, kas ir vienāds ar attēla platumu. Šeit ir attēlota atmiņas šūnu kombinācija, kas ir paredzēta attēlam ar platumu W=22, taču reāliem attēliem šis parametrs ir lielāks, piemēram, W=640.

Abstrahējoties no datu apstrādes (reizināšanas/summēšanas) turpmāk apskatīsim tikai datu bīdīšanu.

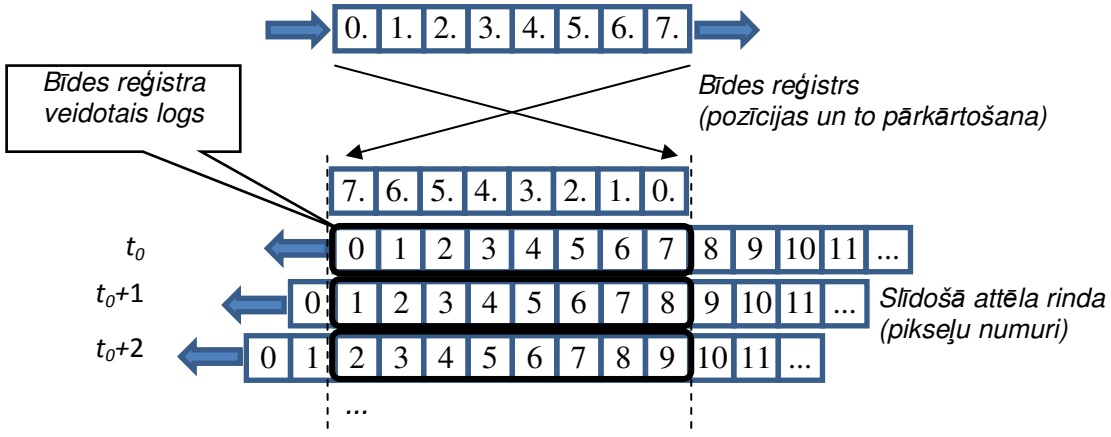
2. Datu bīdīšana

Attēla pikseļi, kas apvienoti rindās, no attēlu sensora nāk virknē. Katrā rindā pikseļi ir sakārtoti virzienā no kreisās puses uz labo, bet pašu rindu secība ir no augšas uz leju (piezīme: pikseļu numuriem netiek likts punkts pie numura; H apzīmē attēla augstumu).

Bīdot datus aktīvajā vai pasīvajā daļā katrs nākamais pikselis (ar lielāku numuru) iebīdās reģistra kreisajā pusē, bet pārējie, jau saglabātie pikseļi, nobīdās pa labi kā parādīts attēlā:

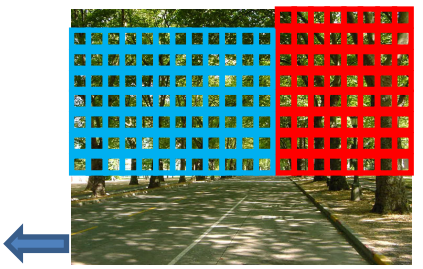


Procesa attēlošanai var uzskatīt, ka bīdes reģistrs veido „logu" (ar pretēji pārkārtotām pozīcijām), kuram apakšā slīd attēla rinda:



Turpmāk šis process tiks attēlots kā filtrējamā attēla pārbīde zem bīdes reģistra veidotā loga:

Vairāku rindu gadījumā, katra rinda slīdēs zem sava loga – visas ar vienādu ātrumu, tāpēc logu var veidot arī kompleksās struktūrās – vairāku daļu veidotās kaskādes, saglabājot attēlu pārbīdīšanas metodi notiekošo procesu attēlošanai. Līdzīgi kā ar viena bīdes reģistra pozīciju pārkārtošanu, veidojot kompleksās atmiņas šūnu struktūras, datu attēlošanai rindu secība arī ir jāsamaina uz pretējo:



Katru filtra atmiņas rindu veido tikpat garu kā attēla rindu (garums ir W). Tā kā dati, kas tiek izbīdīti ārā no kādas rindas (teiksim, ar indeksu *i*) tiek iebīdīti nākamajā rindā *i*+1, tad datu nobīdes procesu varētu attēlot kā *attēlu kopas* pārbīdi zem filtra rindu veidotā loga:

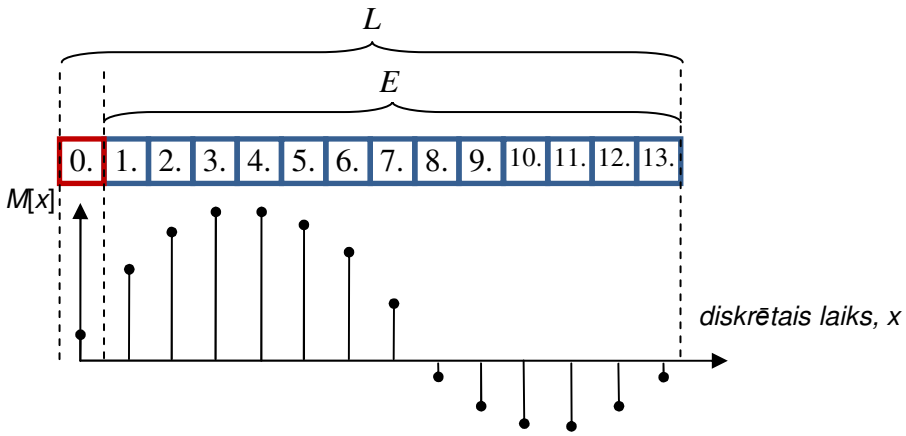
Šādā attēlu kopā katrs nākamais attēls ir novietots 1 rindu augstāk par iepriekšējo, kas šajā attēlošanas modelī simulē *i*-tās filtra rindas datu iebīdīšanu *i*+1jā rindā. Procesi, kas notiek aiz attēla malām tiek apskatīti vēlāk un ir saistīti ar attēla papildināšanu, jeb *padding*.

3. Filtru tipi

Pastāv dažāda tipa filtri – *kauzālie* (izejas dati pieejami jau kopā ar ieejas datiem) un *nekauzālie* (izejas dati tiek saņemti ar aizturi pret ieejas datiem), kaut gan kauzalitāte filtriem tiek definēta pēc citas pazīmes. Pēdējiem ir spēkā gadījums, kad veidojamā aizture ir maksimāli liela (*antikauzālie* filtri), bet tie netiek apskatīti. Šajā nodaļā nekauzalitātes ietekme uz filtra darbību tiek demonstrēta, izmantojot 1D signālus, taču iegūtās zināšanas var pielietot arī 2D filtriem.

Kauzālie filtri

Pieņemsim, ka ir 1D FIR ar sekojošu maskas M[x] (arī sauc par filtra impulsa reakciju) struktūru:

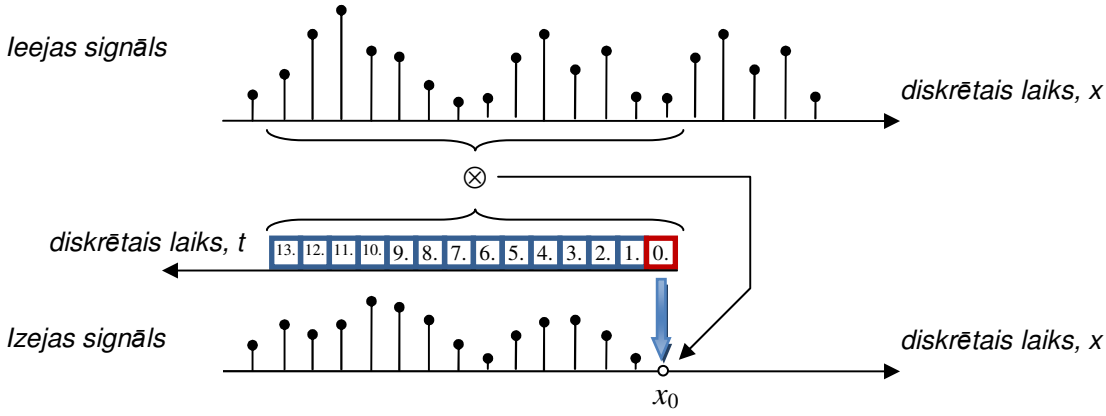


Filtra maskas garumu apzīmē ar L. Šajā piemērā L = 14, bet var būt arī cits lielums. Maskas atskaites punktu, kas turpmāk tiks saukts par centru – maskas centru, aktīvās daļas centru, u.tml. – apzīmē ar parametru C un šajā piemēra C = 0 (turpmāk attēlos tiek apzīmēts ar tumši-sarkanu krāsu).

Maskas sākumu (nolašu skaitu pa kreisi no centra) apzīmē ar S, un aprēķina kā S = C. Šajā piemērā S = 0. Maskas galu E aprēķina kā E = L – (C + 1), tātad šeit tas ir 13.

Veicot konvolūciju, maska tiek izvietota virs signāla ar apgrieztu laika asi (pēc argumenta *t*), kā ir uzdota pēc izteiksmes:

$$g[x_0] = \left(f[x] \otimes M[x]\right)_{x=x_0} = \sum_{t=0}^{L-1} f[x_0 - t] \cdot M[t]$$

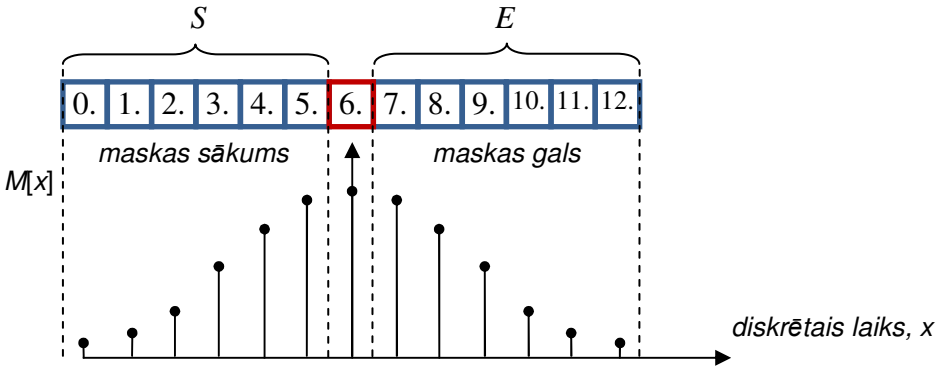


Var redzēt, ka maskas ievietošanas punkts sakrīt ar iegūstamo filtrēto nolasi pēc pozīcijas x_0; filtrētā vērtība ir atkarīga tikai no aktīvajā daļā iepriekš iebīdītām ieejas signāla vērtībām. To var redzēt pēc tā, ka centrālajam punktam (šajā gadījumā – 0.) labajā pusē nav neviena cita atmiņas elementa.

Šādus filtrus (ar C = 0) sauc par ***kauzāliem*** un to implementācijai nav nepieciešami nekādi datu aiztures mehānismi – katra ieejas signāla vērtība ļauj uzreiz iegūt tai atbilstošu izejas signāla vērtību.

Nekauzālie filtri

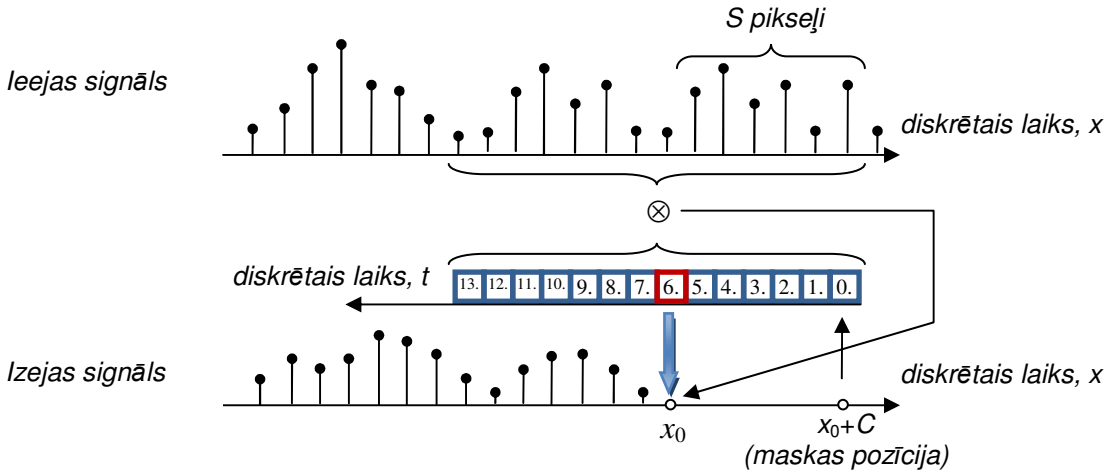
Kauzālo filtru pretstats ir ***nekauzālie filtri*** ar C > 0. Šādu filtru maskas struktūra ir sekojoša:



Šajā piemērā: maskas garums L = 13 nolases; maskas centrs C = 6; S = C = 6 nolases veido maskas sākumu, bet E = L – (C + 1) = 6 nolases veido maskas galu.

Veicot konvolūciju ar šādu filtru, S nolasēm ir jābūt zināmām pirms ir iespējams dabūt izejas datu pirmo nolasi:

$$g[x_0] = \left(f[x] \otimes M[x]\right)_{x=x_0} = \sum_{t=0}^{L-1} f[x_0 + S - t] \cdot M[t]$$

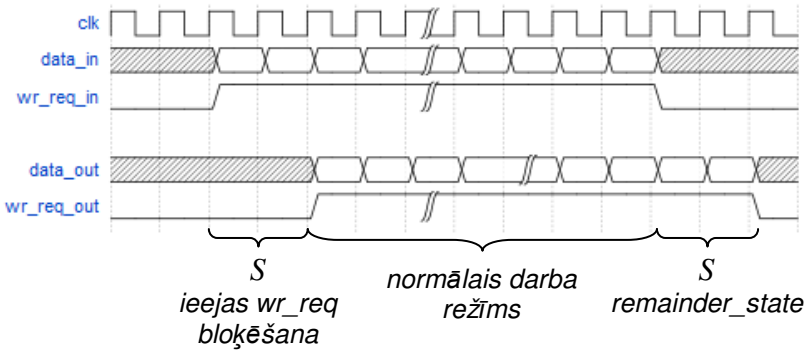


Šādus filtrus realizē ar aizturēm vai arī katram signāla fragmentam:

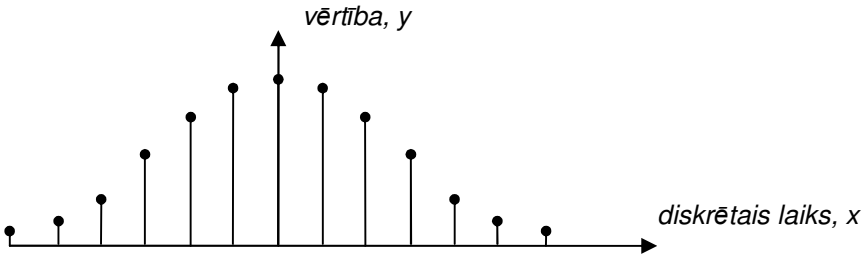
- vispirms S = 6 nolasēm tiek nobloķēts (netiek padots uz izeju) rakstīšanas pieprasījuma wr_req signāls, kaut arī pats filtrs šādu signālu ieejā neignorē;
- S nolases tiek iebīdītas bīdes reģistrā līdz centra pozīcijai;
- tālāk filtru simulē kā kauzālo filtru;
- lai saglabātu datu apjomu, pēc signāla beigām filtrs (konkrētāk – filtra daļa, ko sauc par *ģeneratoru*) pats saģenerē izejā tikpat daudz wr_req impulsu, cik sākumā bija nobloķēts, tādejādi, dati tiek izbīdīti ārā no filtra reģistriem.

Veicot šādu operāciju, ir svarīgi, lai ģenerators paspētu saģenerēt visus nepieciešamos impulsus pirms atnāk jaunais signāla fragments (piemēram, rindas pikseļi). Ja tomēr jaunie dati atnāk ātrāk, tad ģenerators pārslēdzas sekošanas režīmā (*remainder state*) un seko līdzi ieejas datu rakstīšanas pieprasījumiem, pašam vairs wr_req neģenerējot (vajadzīgs, lai neizjauktu sinhronizāciju). Taču šajā režīmā izejas dati tiek būtiski izkropļoti (tāpēc, ka ieejas

dati maina bīdes reģistru saturu). Teiktais attiecas pret jebkuru koordinātu asi, piemēram, X ass virzienā teiktais attiecās pret rindas pikseļiem, bet Y ass virzienā nekauzalitāte nozīmē, ka ir jāveido S rindu lielu aizturi. Tipiskā laika diagramma varētu izskatīties sekojoši:



Redzams, ka S=2 un ģenerators paspēj izbīdīt datus no filtra bīdes reģistriem
Attēla izpludināšanas filtrs ir nekauzāls, jo tā maskas nolases ir simetriski izvietotas pret Y asi:



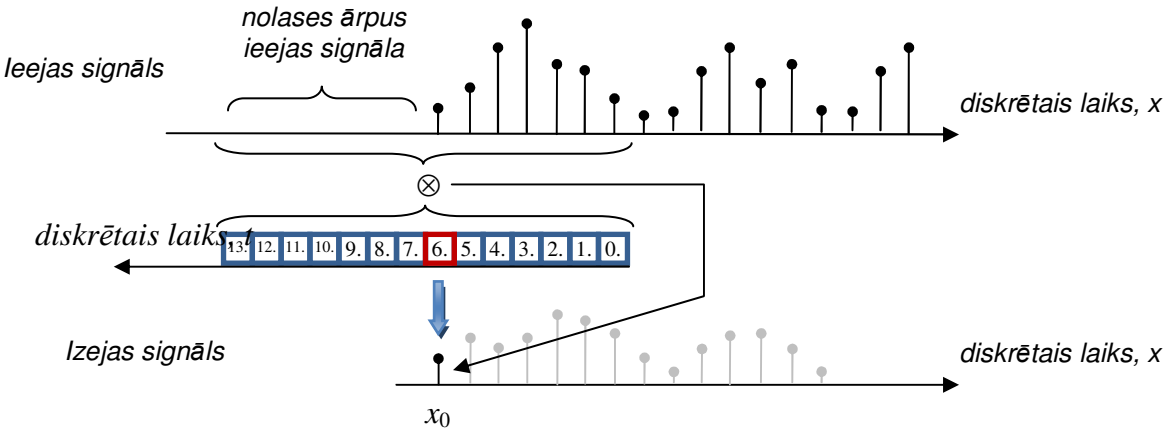
Tas nozīmē, ka šāda filtra realizācijai ir nepieciešams ievērot izejas datu aizturi par S pikseļiem vai S rindām (atkarīgs no filtra tipa – vertikāls, vai horizontāls).

4. Signāla papildināšana

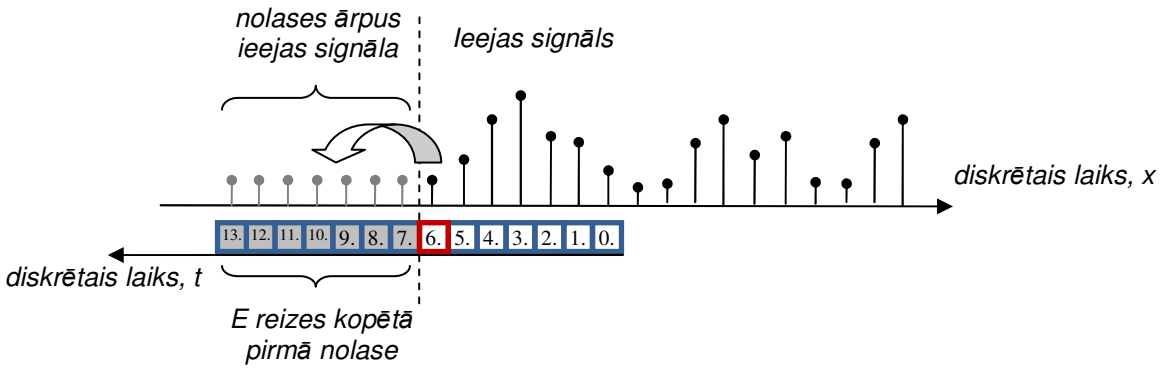
Aiztures veidošana nav vienīgā problēma, kas parādās veidojot filtrus. Otrā problēma ir filtrēto datu kropļojumi pie signāla (konkrētāk – attēla) malām. Tie veidojas tāpēc, ka, iegūstot pirmās un pēdējās nolases, daļa no maskas nolasēm pārklāj laukumu ārpus ieejas signāla.

Signāla papildināšana tā sākumā

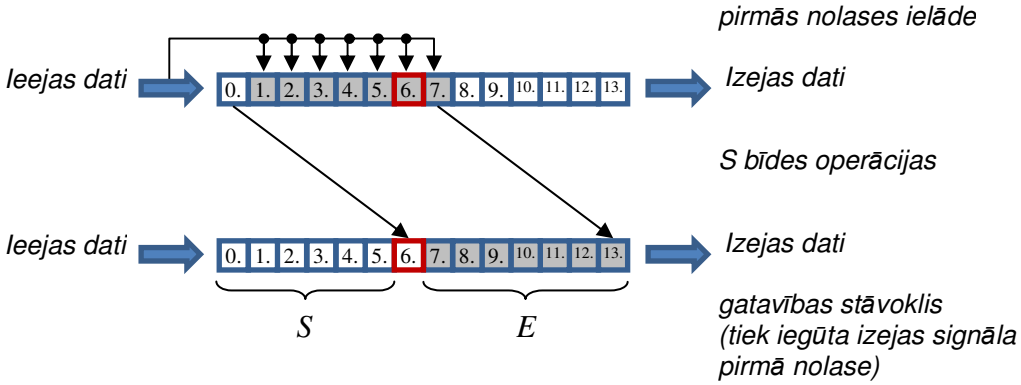
Tipiskā situācija, uzsākot signāla apstrādi daļa no filtra maskas tiek uzklāta virsū reģionam, kurā ieejas signāla vērtības nav definētas:



Kā kompromiss starp realizācijas vienkāršību un filtrētā signāla kvalitāti, tukšo nolašu vietās (maksimāli šādas nolases var būt E = L – (C + 1) un šajā piemēra E = 7) tiek nokopēta signāla pirmā nolase:



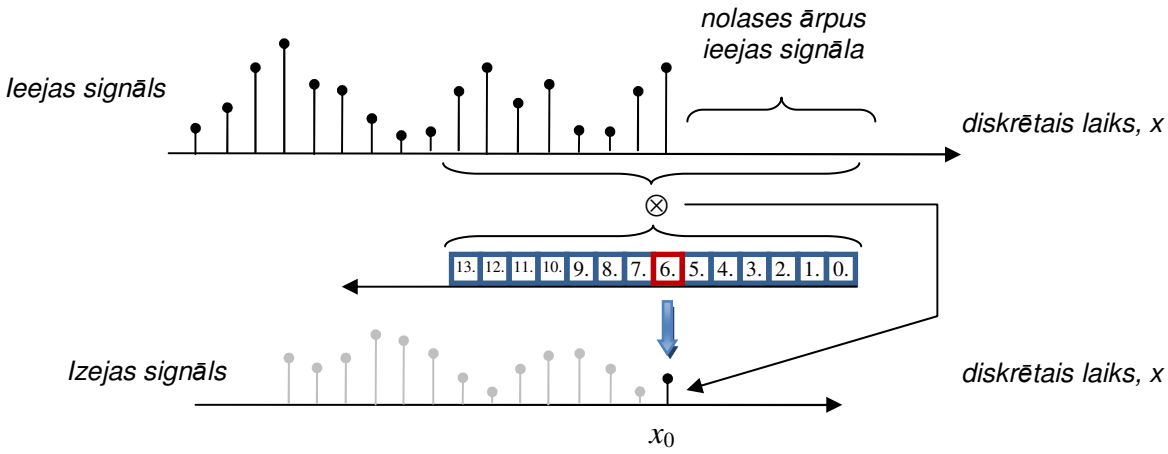
Lai realizētu šādu iespēju ir nepieciešams paralēlais bīdes reģistrs ar iespēju pirmajām E+1 šūnām paralēli ielādēt vienu un to pašu vērtību (pirmo nolasi un tās E kopiju).



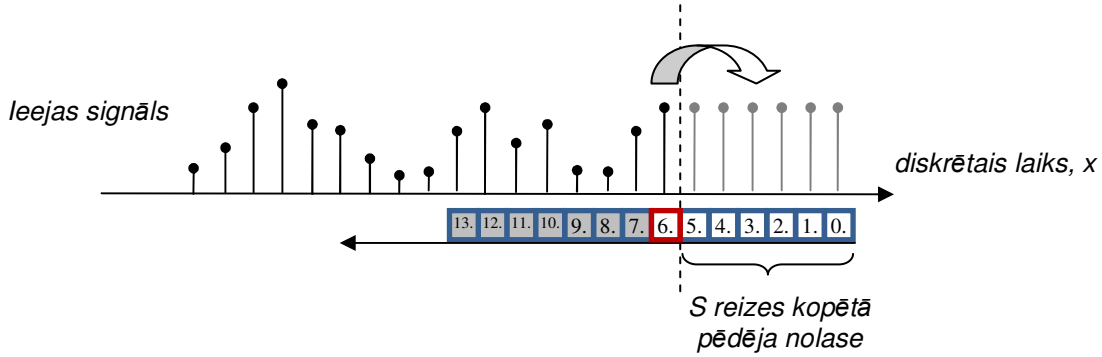
Ielādējot signāla (piemēram, attēla rindas) sākumā E+1 vērtības, pēc S bīdes operācijām tās atradīsies signāla papildināšanas reģionā (attēlā – bīdes reģistra pozīcijas 7.-13.), bet bīdes reģistrs būs gatavības stāvoklī. Tieši šajā momentā var iegūt izejas signāla pirmo nolasi.

Signāla papildināšana tā beigās

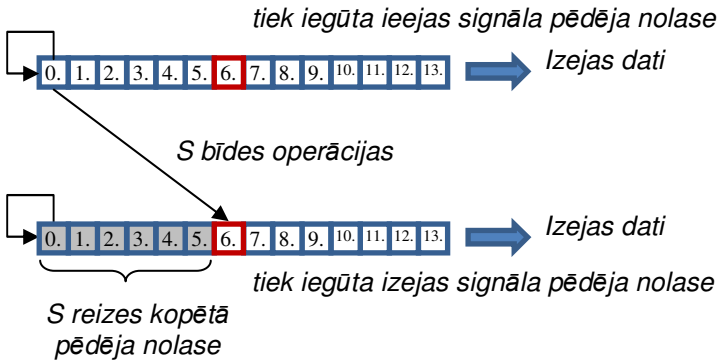
Pabeidzot signāla apstrādi veidojas sekojoša situācija, ka daļa no maskas ir izvietota ārpus ieejas signāla nolasēm:



Kā kompromiss, starp realizācijas vienkāršību un filtrētā attēla kvalitāti, tukšo nolašu vietās (maksimāli šādas nolases var būt S un piemērā S = 6) tiek nokopēta attēla pēdējā nolase.



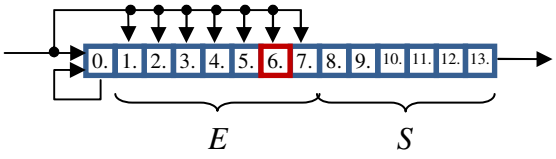
Šāda bīdes reģistra realizācija ir nepieciešama, lai varētu nolasīt 0. pozīcijā ierakstītā pikseļa vērtību un ielādētu to bīdes reģistra ieejā (citi ieejas dati tiek nobloķēti).



Apkopojums

No teiktā var secināt, ka:

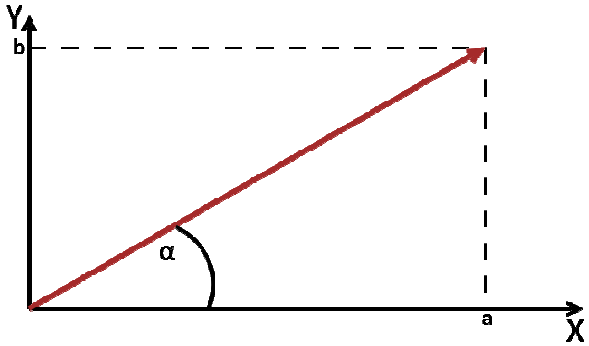
- Filtra maskas parametri ietekmē filtra atmiņas organizāciju:
  - E elementiem, sākot no otrā, tiek realizēta paralēlā piekļuve rakstīšanai (raksta ieejas datus tikai iegūstot pirmo nolasi ieejā);
  - pirmajam elementam realizēta datu kopēšana (saglabā šajā elementā rakstīto saturu visām nolasēm pēc signāla beigām).



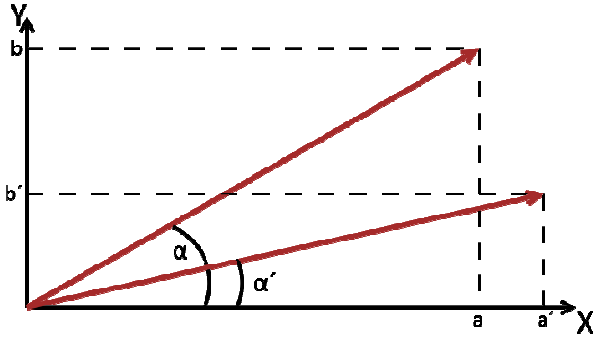
- Filtra maskas parametri ietekmē tā uzvedību (nekauzālu filtru gadījumā):
  - parametrs E nosaka kopējamo datu apjomu signāla (kadra/rindas) sākumā,
  - parametrs S nosaka kopējamo datu apjomu signāla (kadra/rindas) beigās, kā arī ieejas datu ignorēšanas un ģeneratora darbības ilgumu.

Dubultā leņķa samazināšana

Uzdevumā ir dots vektors a, b, ar garumu r=a2+b2 kurš ar x asi veido leņķi α. (skatīt attēlā).



Jāatrod vektors a', b' tā, lai tā garums r=a2+b2, bet leņķis ar x asi α'=α2 (skatīt attēlā).



Lai veiktu leņķa samazināšanu uz 12, tiek pielietoti divi CORDIC (***CO****ordinate* ***R****otation* ***DI****gital* ***C****omputer*) algoritmi – pāreja no Dekarta uz polārajām koordinātām un pāreja atpakaļ no polārajām uz Dekarta koordinātām.

Apskatīsim pāreju no Dekarta, uz polārajām koordinātām. Pieņemsim, ka a=100, b=60, bet ir jāatrod leņķis α, un vektora garums r.

Izmantojot CORDIC algoritmu, gan leņķis, gan vektora garums tiek meklēts, izmantojot visvienkāršāk implementējamās matemātiskās darbības – saskaitīšanu, atņemšanu un bitu nobīdes. Lai vēl vairāk samazinātu veicamo darbību skaitu, šajā implementācijā algoritms tiek izpildīts noteiktu reižu skaitu, šajā piemērā 15 reizes.

Tiek izveidoti 3 mainīgie - x, y, α, sākotnēji mainīgajiem tiek piešķirtas šādas vērtības:

x=a= 100; y=b= 60; α=0.

Katrā no algoritma iterācijām tiek salīdzināts vai y ir lielāks vai mazāks par nulli. Ja y ir lielāks par nulli, tad tiek veiktas šādas darbības:

$$x = x + y2n - 1$$

$$y = y - x2n - 1$$

$$\alpha = \alpha + k(n)$$

kur n – iterācijas kārtas numurs, bet k – koeficentu matrica, kuru koeficenti iegūti:

$$k(n) = 285{,}95 \cdot 180\ \pi \cdot \text{arctg}2 - n - 1$$

Arktangensa vērtības skaitļiem 1,12,14,18,…. tiek aprēķinātas šīs vērtības pareizinot ar 180π un iegūst iegūtas leņķu vērtības φn=45°, 26.6°, 140°, 7.1°…

Koeficienti reizināti ar skaitli 285.95 tādēļ, lai koeficents k15=1.

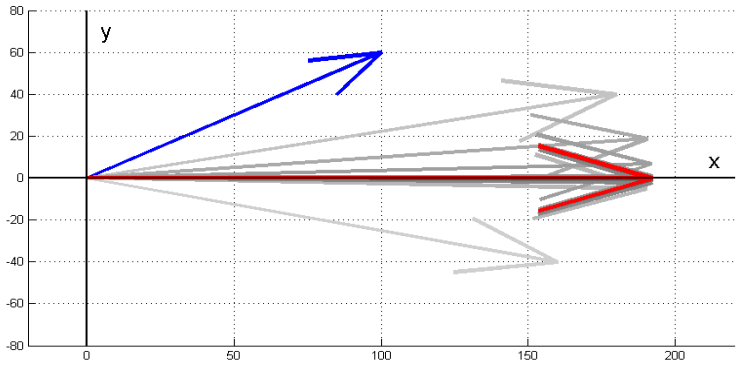
Ja y ir mazāks par nulli, tad tiek veiktas šādas darbības:

$$x = x - y2n - 1$$

$$y = y + x2n - 1$$

$$\alpha = \alpha - k(n)$$

Izpildot CORDIC algoritmu pāriešanai no Dekarta, uz polāro koordinātu sistēmu, sākotnējais vektors tiek rotēts 15 reizes, katru reizi pārrēķinot tā x un y koordinātes:



Attēlā zilais vektors ir dotais sākuma vektors (100, 60), bet sarkanais vektors ir rotētais vektors, izmantojot 15 CORDIC algoritma rotāciju. Pēc tam, kad CORDIC algoritms ir izpildīts 15 reizes, pie ieejas vērtībām a=100, b=60, mainīgo x, y, α vērtības ir šādas:

$$x = 192.04$$

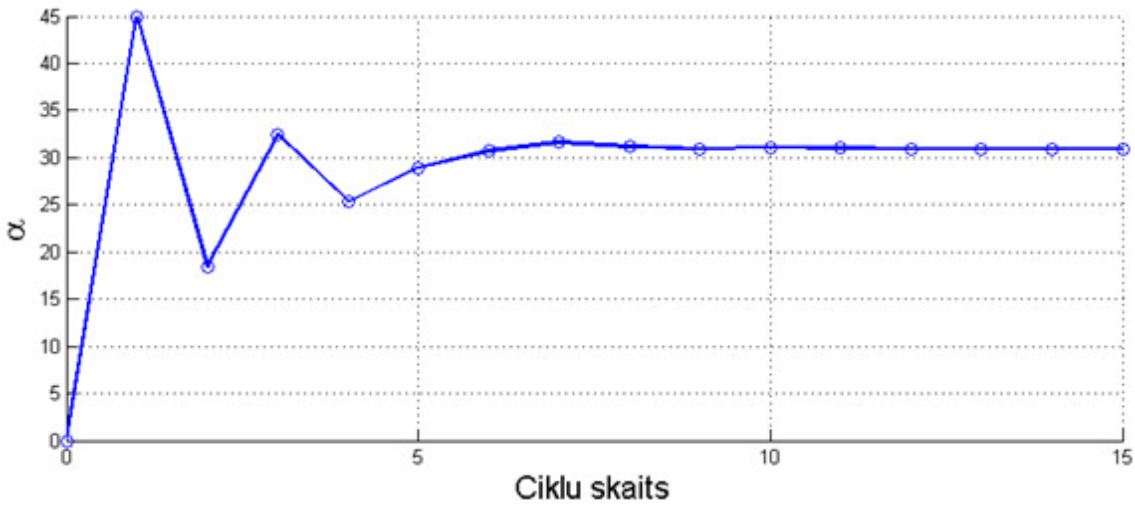
$$y = 0.010415$$

$$\alpha = 8855$$

y vērtība raksturo kļūdu – ja y būtu 0, tad sākotnējās vektors būtu ideāli „pierotēts” pie x ass. α raksturo leņķi (grādos), tikai šī vērtība ir jāizdala ar skaitli 285.95:

$$\alpha = 8855/285.95 = 30.967°$$

Leņķa vērtības izmaiņu katrā CORDIC algoritma taktī var apskatīt zemāk redzamajā attēlā



x vērtība raksturo sākotnējā vektora a,b garumu. Katrā CORDIC algoritma izpildes reizē x vērtība tiek palielināta par 1cosφ, tādēļ pēc N algoritma izpildes reizēm mums vērtība ir jāpareizina ar skaitli:

$$c = n = 1 \cdot N \cdot \cos(\varphi n) \approx 0.60725$$

$$r = x \cdot c = 192{,}04 \cdot 0{,}60725 = 11625$$

patiesās vērtības α un r ir:

$$r = a2 + b2 = 1002 + 602 = 116.62$$

$$\alpha = 180\pi \cdot \operatorname{arctg}(ba) = 30{,}948$$

Pāriešana atpakaļ no polārajām uz Dekarta koordinātēm notiek līdzīgi, tikai tagad tiek salīdzināts, vai tikko aprēķinātais leņķis α2 ir lielāks vai mazāks par uzdoto sākuma leņķi α', bet a un b vērtības tiek pārrēķinātas pretēji tām kā aprakstīts iepriekš.

Dubultā leņķa samazināšana FPGA implementēta tā, kā parādīts apakšējā attēlā redzamajā shēmā. Bloka, kas rūpējas par dubultā leņķa samazināšanu, ieejā tiek padotas vektoru a un b koordinātās, kā arī nepieciešamie kontroles signāli (netiks apskatīti). Pirmā operācija ir kvadranta noteikšana, kurā atrodas vektors, un vektora pagriešana. Ja vektors atrodas I vai IV kvadrantā, parametri tiek nodoti tālāk bez izmaiņām:

$$a1 = a;$$

$$b1 = b;$$

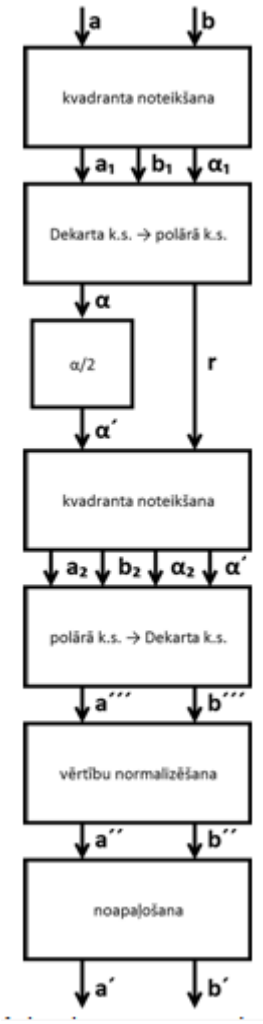
$$\alpha 1 = 0,$$

Ja ieejas vektors atrodas II kvadrantā, tad tas tiek transponēts uz I kvadrantu, tālāk nododot šādas vērtības:

$$a1 = b$$

$$b1 = - a$$

$$\alpha 1 = 285{,}95 \cdot 90°$$

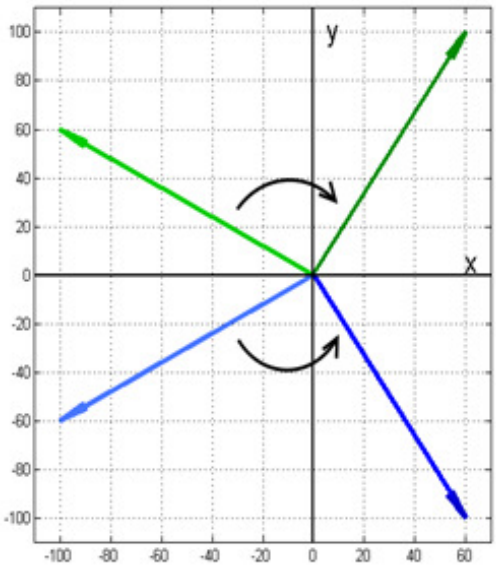


Attēlā ar zaļo līkni uzskatāmi parādīta ieejas vektora rotācija no II uz I. Bet ieejas attēla transportēšana no III uz IV kvadrantu parādīta ar zilo līkni un tā vērtības ir:

$$a1 = - b$$

$$b1 = a$$

$$\alpha 1 = - 285{,}95 \cdot 90°$$



Tālāk notiek pāriešana no Dekarta koordinātu sistēmas uz polāro koordinātu sistēmu, izmantojot jau iepriekš aprakstīto CORDIC algoritmu tiek izpildīta vektora rotēšana 15 reizes. Rezultātā tiek iegūts leņķis α un vektora garumu r.

Tiek veikta leņķa dalīšana uz divi - α'=α2.

Izmantojot jauniegūtās vērtības -α' un r, atkal tiek noteikt kvadrants, kurā atrodas vektors, kuram nu jau leņķis ir samazināts par 12. Ja α' vērtība ir lielāka par 285,95 · 90° (t.i., ja vektors atrodas II kvadrantā), tad nākošajam CORDIC algoritmam tiek nodotas šādas vērtības: a2 = - r un α2 = 285,95 · 90°.

Šeit vērtība α2 ir vērtība, kurai katrā CORDIC algoritma, ar kura palīdzību tiek pāriets no polārās koordinātu sistēmas uz Dekarta koordinātu sistēmu, iterācijā tiek pieskaitīta vai atņemta kāda kn vērtība (skatīt iepriekš).

Ja α' vērtība ir mazāka par -285,95 · 90° (t.i., ja vektors atrodas III kvadrantā), tad nākošajam CORDIC algoritmam tiek nodotas vērtības: a2 = - r un α2 = - 285,95 · 90°.

Ja - 285,95 · 90° < α' < 285,95 · 90°, tad tālāk tiek nodotas sekojošas sākuma vērtības: a2 = r un α2 = 0°

Vērtība α' ir augšējā attēlā iegūtā pusleņķa vērtība un tā tiek saglabāta visu iterāciju laiku – ar šo vērtību tiek salīdzināta α2 vērtība. Bet mainīgā b2 sākumvērtība vienmēr ir nulle - b2 visās iterācijās glabā informāciju par b''' vērtību (nenormalizēta un nenoapaļota vektora a', b' x koordinātes vērtība).

Pēc 2. CORDIC algoritma kaskādes tiek iegūtas vērtības a''' un b'''. Tās tiek reizinātas ar skaitli 24'167 (rezultātā iegūstot a'', b'' vērtības), tad noapaļotas, un dalītas ar 216:

$$a' = a''' \cdot 24'167 / 216 = a''' \cdot 0{,}36876$$

Skaitlis 0.36876 ir tuvināta vērtība vērtībai c2:

$$c2 = n = 1N\cos\varphi n\ 2 = 0.607252 = 0.36875$$

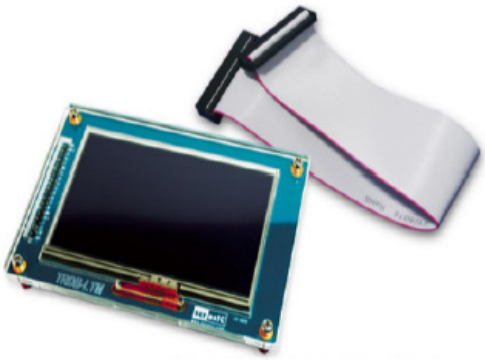
Tā ir nobīde, kas radusies CORDIC algoritma dēļ.

Apkopojums

- Bloks, kas rūpējas par dubultā leņķa samazināšanu, izveidots kā aiztures līnija (*pipeline*).
- Kopumā bloks veic 37 secīgas darbības, katra darbība tiek veikta, izmantojot divas 125 Mhz clk signāla taktis. Tas nozīmē, ka padodot bloka ieejā pirmos datus (a,b), rezultāts a',b' bloka izejā parādīsies pēc 74 clk signāla taktīm.
- Ja dati ieejā tiks bīdīti katrā otrajā taktī, tad izejā katrs nākošais rezultāts parādīsies katrā otrajā taktī.

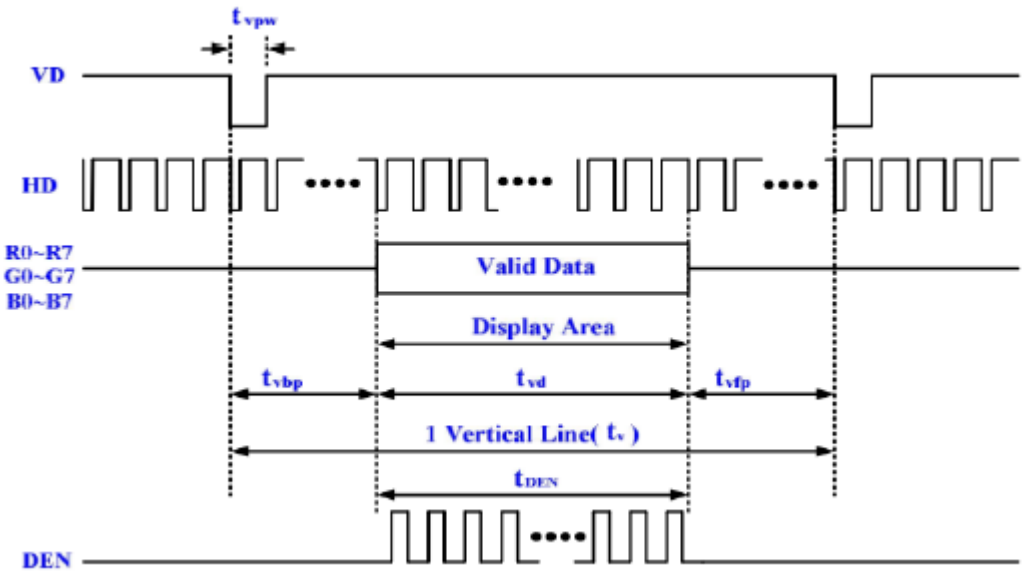
Šķidro kristālu displeja (LCD) kontroliera izveide FPGA platformā

Projekta pirmais demonstrators sevī iekļauj LCD, kurš ir paredzēts, attēlu sensora datu attēlošanai. Līdz šim dati tika attēloti uz VGA, tāpēc nācās izveidot LCD kontrolieri FPGA platformā. Izmantotais LCD displejs – *TRDB-LTM* no *Terasic* satur *AD7843* mikroshēmu un vadības moduli ar reģistriem. Tika pētīts LCD darbības princips un laika diagrammas, lai tās varētu implementēt FPGA.



Darbības princips

LCD darbības princips ir līdzīgs VGA, tādējādi nācās tikai pilnveidot esošo VGA kontrolieri. Atšķirība ir papildus signālā (DEN – data enable) un sinhronizācijas daļā. Pārrakstot LCD vadības reģistrus iespējams mainīt izšķirtspējas: [800x480, 480x272, 400x240] un tām atbilstošās darba frekvences: [33.3, 9, 8.3] MHz.

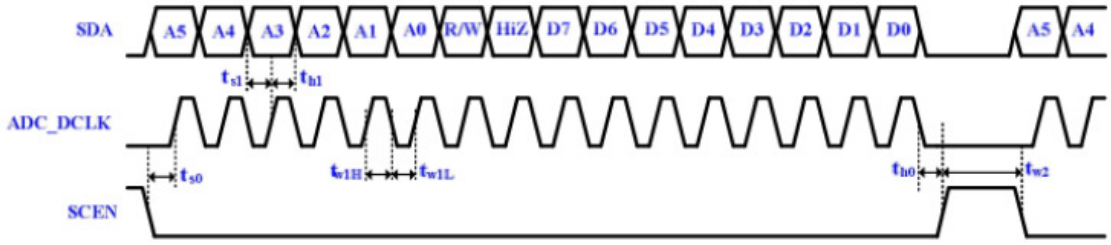


Problēmas

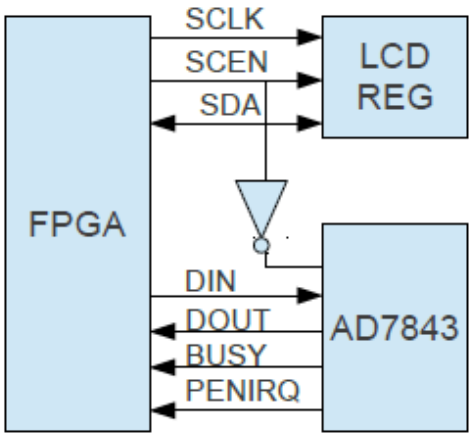
1. Ar demonstratoru iegūtais attēls, kurš attēlots uz LCD ir samērā tumšs, jo attēlu sensors ir dziļi ierīcē. Vērojot displeju plauksta nebija redzama, tāpēc nācās palielināt LCD spilgtumu. Lai to izdarītu bija nepieciešams izveidot LCD reģistru kontrolieri FPGA platformā, kurš sazinātos ar LCD iekšējiem reģistriem.
2. Pēc noklusējuma LCD darbojas ar 33.3 MHz lielu frekvenci un 800x480 lielu izšķirtspēju. Tas rada papildus problēmas, jo attēlu sensors iegūst datus ar 25 MHz. Lai pie šādiem nosacījumiem viss darbotos būtu jāievieš vēl papildus funkcijas projektā. Tika nolemts LCD uzstādīt 400x240 izšķirtspēju ar 8.3 MHz frekvenci, lai to izdarītu jābūt 1.punktā aprakstītajam LCD reģistru kontrolierim.

LCD reģistru kontrolieris

Atbilstoši uzdotajām laika diagrammām tika izveidots LCD reģistru kontrolieris, kurš izmanto virknes interfeisu saziņai starp FPGA un LCD.



Ar realizētā kontroliera palīdzību tika nomainīta LCD izšķirtspēja un atrastas optimālās gammas un spilgtuma vērtības. Apakšējā attēlā redzams reģistru kontrolieris un mikročips AD7843, kurš atbild par skārienjūtīgo funkciju.



Attēlošana

Tā kā LCD un VGA kontrolieri ir līdzīgi, tad projektā nebija jāievieš kardinālas izmaiņas. Tika ņemts vērā fakts, ka VGA darbojas ar 25 MHz, bet LCD ar 8.33 MHz (starpība 3x). Līdz ar to, kopējā projektā visiem kontroles signāliem bija jāievieš aiztures.

Sākotnēji demonstratora LCD attēlojot datus ekrānā neietvēra plaukstu, kura novietota uz ierīces. Tika izmainīts lineārās adreses aprēķins(eksperimentāli tika noteikta sākuma vērtība, pie kuras LCD sāk attēlot mums interesējošo apgabalu), kura nolasa datus no ASRAM.

Turpmākie plāni:

- izpētīt filtra maskas parametru ietekmi uz filtru;
- turpināt darbu pie dubultā leņķa samazināšanas metodes;
- problēmu novēršana, ko radīja šķidrā kristāla displeja kontroliera izveide FPGA platformā

Biometrisko datu kriptēšanas, glabāšanas un lietojama apakšsistēmas izveide

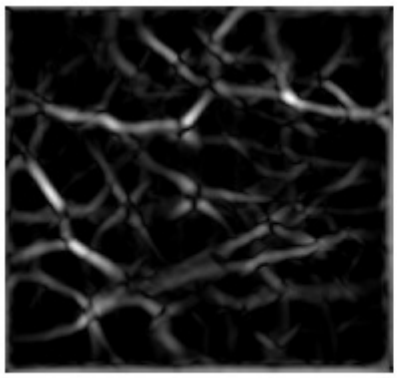
Darbs ar viedkarti

Ir izveidota viedkartes programma, kurā ir iespējams ierakstīt cilvēka datus. Uz karti ir iesējams aizsūtīt cilvēka vektorus, kas tiek salīdzināti ar kartē glabātajiem. Karte kā rezultātu

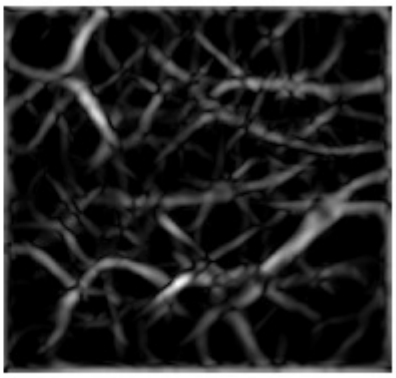
atgriež abu paraugu līdzību. Dati, kurus vajag nosūtīt uz karti ir daudz lielāki nekā ir iespējams nosūtīt vienā reizē, tāpēc viņi tiek sadalīti daļās, lai var aizsūtīt. Uz kartes šīs daļas tiek saņemtas un no viņām tiek rekonstruēts sākotnējais buferis. Aizsūtītie dati tiek interpretēti kā vektori un, lai salīdzinātu datus, tiek veikta vektoru salīdzināšana. Rezultāts tiek atgriezts kā skaitlis ar 64 iespējamām vērtībām, līdz ar to ļaujot veikt salīdzināšanu ar 1.5% izšķirtspēju. Kartēm ir arī izstrādāta funkcionalitāte, ar kuras palīdzību ir iespējams pārbaudīt vai sūtītie dati tiek pareizi sūtīti un vai pareizi tiek rekonstruēti uz kartes.
Turpmāk tiek plānots pilnveidot un papildināt izveidoto programmu, novēršot kļūdas, kas rodas darba procesā.

Atsevišķu sistēmas komponenšu un bloku izstrāde un izpēte

Šajā laika periodā svarīgākie veiktie darbi ir *Phase only correlation* algoritma realizācija un izpēte. Ir veikta fāzes korelācijas algoritma izpēte un datorprogrammu izveide šīs metodes pārbaudei ar plaukstas biometrijas datiem.
Izmantojot fāzes korelāciju ir iespējams salīdzināt divus attēlus. Ja ir divi dažādi attēli, f(x,y) un g(x,y), to attiecīgās 2D Furjē transformācijas ir F(u,v) un G(u,v). Zemāk redzami divi plaukstas asinsvadu attēli, kas tiks izmantoti šajā piemērā.



(a)



(b)

Attēlu f(x,y) un g(x,y) Furjē transformācijas F(u,v) un G(u,v) aprēķina sekojoši:

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)},$$

$$G(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} g(x,y)e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)},$$

Kur f(x,y) un g(x,y) ir iepriekš minētie M x N izmēra attēli. F(u,v) un G(u,v) var tikt pierakstīti arī kā

$$F(u,v) = A_F(u,v)\cdot e^{j\theta_F(u,v)}$$

$$G(u,v) = A_G(u,v)\cdot e^{j\theta_G(u,v)}$$

Kur $A_F(u,v)$ un $A_G(u,v)$ ir amplitūdu komponentes un $e^{j\theta_F(u,v)}$ un $e^{j\theta_G(u,v)}$ ir F(u,v) un G(u,v) attiecīgās fāzes komponentes. $R_{FG}(u,v)$ ir šķērsspektrs starp F(u,v) un G(u,v), ko aprēķina sekojoši:

$$R_{FG}(u,v) = F(u,v)G(u,v) = A_F(u,v)A_G(u,v)e^{j\theta(u,v)},$$

kur $\overline{G(u,v)}$ ir G(u,v) kompleksi saistītais un $\theta(u,v)$ ir fāzes starpība $\theta_F(u,v) - \theta_G(u,v)$. Korelācijas funkciju $r_{fg}(x,y)$ var aprēķināt kā 2D inverso Furjē transformāciju no $R_{FG}(u,v)$:

$$r_{fg}(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} R_{FG}(u,v)\, e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

Šķērs-fāzes spektru vai arī normalizēto šķērsspektru $\ddot{R}_{FG}(u,v)$ aprēķina kā

$$\hat{R}_{FG}(u,v) = \frac{F(u,v)G(u,v)}{\left|F(u,v)\overline{G(u,v)}\right|} = e^{j\theta(u,v)}$$

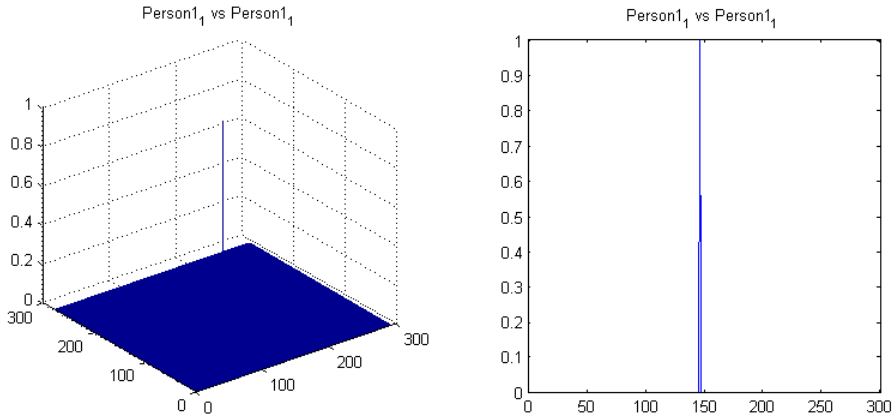
Fāzes korelācijas funkcija $\hat{r}_{fg}(x,y)$ ir 2D inversā Furjē transformācija no $\ddot{R}_{FG}(u,v)$:

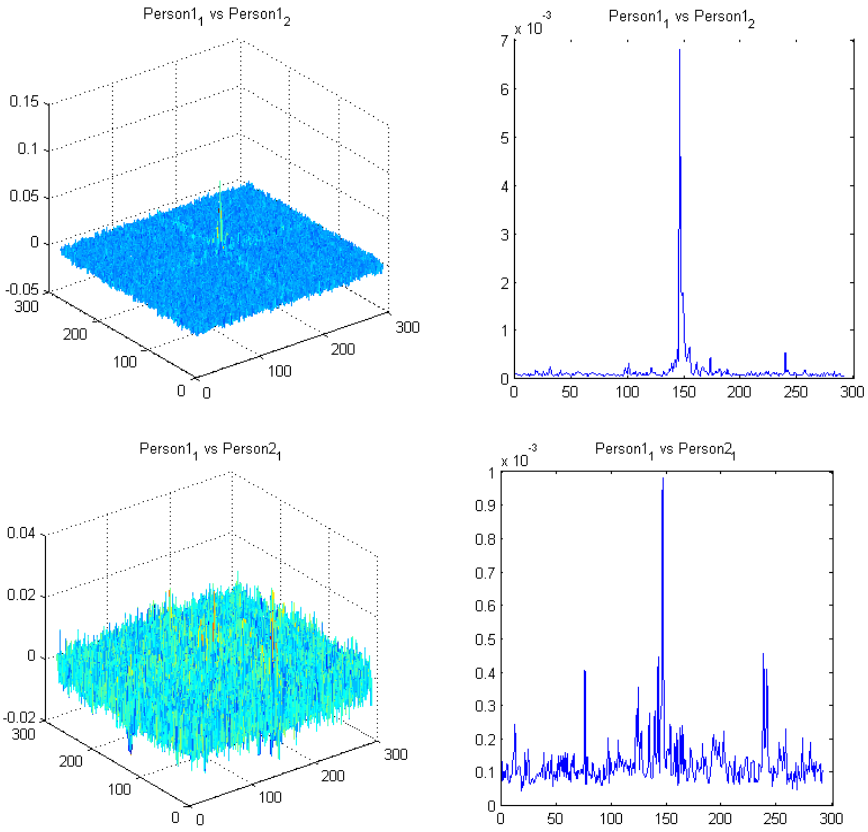
$$r_{fg}(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} \hat{R}_{FG}(u,v)\, e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

Kad f(x,y) un g(x,y) ir viens un tas pats attēls (f(x,y) = g(x,y)), tad fāzes korelācijas funkcija ir

$$\hat{r}_{ff}(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} = \delta(x,y)$$
$$= \begin{cases} 1 & ja\ x = y = 0 \\ 0 & citos\ gadījumos \end{cases}$$

No iepriekšējās izteiksmes izriet, ka fāzes korelācijas funkcija starp diviem vienādiem attēliem ir Kronekera delta funkcija $\delta(x,y)$. Līdzīgu attēlu fāzes korelācijas funkcijai būs raksturīgs pīķis, kas nosaka attēlu līdzības pakāpi. Ja attēli savā starpā ir nobīdīti pa x vai y asi, tad attiecīgi fāzes korelācijas funkcijas pīķis tiks nobīdīts no centra. Šādā veidā var noteikt attēlu savstarpējo nobīdi un to līdzību. Zemāk redzams piemērs kur tiek salīdzināts plaukstas asinsvadu testa attēls pats ar sevi, kā arī vienas personas divi dažādos laika momentos iegūti plaukstas asinsvadu attēli, kā arī divu dažādu personu plaukstas asinsvadu attēli.





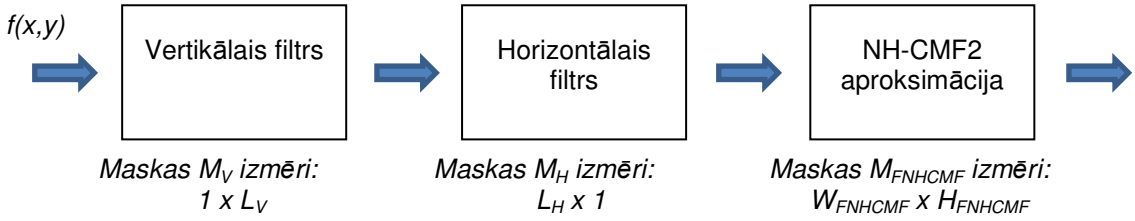
Aktivitāte: Eksperimentālā izstrāde

Algoritmu implementēšana eksperimentālajā maketā

Filtrs ir aprakstīts atskaites lietišķo pētījumu sadaļā „Sejas un plaukstas attēlu iegūšanas metožu izstrāde", šajā sagaļā tiks aprakstīta filtra realizācija.

Fast NH-CMF2 realizācija

Veidojamo filtru sadala uz vairākiem filtriem, kas ir slēgti kaskādē:



Filtri realizē konvolūcijas asociatīvo īpašību:

$$f(x,y) \otimes M_{NHCMF} \approx ((f(x,y) \otimes M_V) \otimes M_H) \otimes M_{FNHCMF}$$

kur $M_{NHCMF} \approx M_V \otimes M_H \otimes M_{FNHCMF}$ ir oriģinālā NH-CMF2 maska, tiek aproksimēta ar vienkāršāk realizējamo filtru masku konvolūciju.

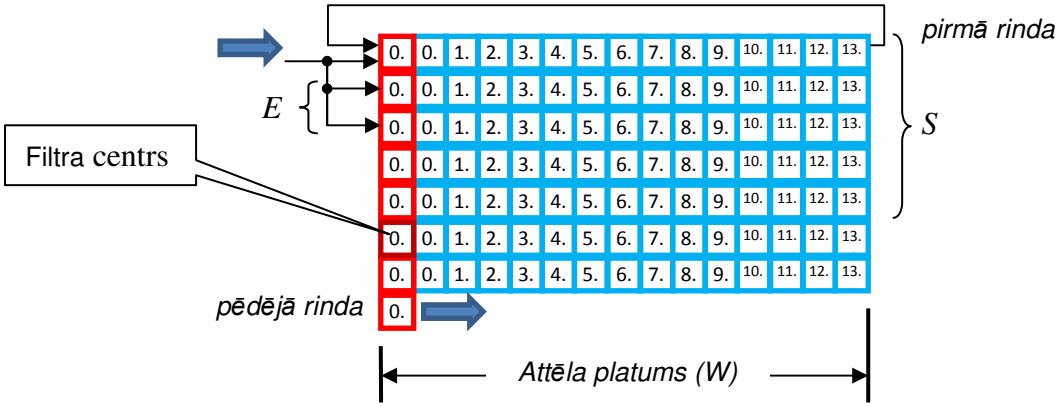
Sekojošas struktūras realizācijai ir jāizveido trīs filtri:

1. Vertikālais izpludināšanas filtrs M_V,

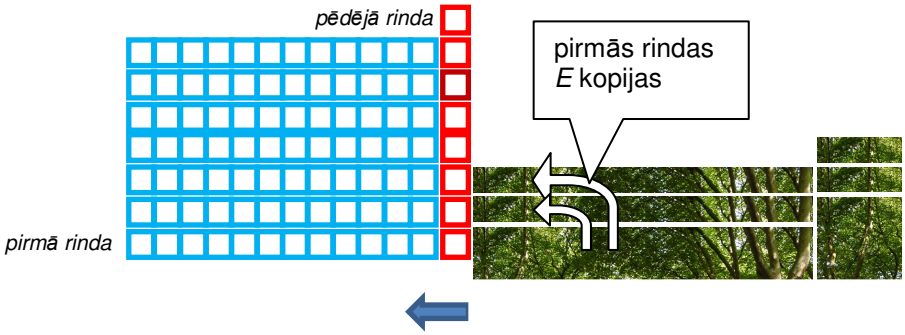
2. Horizontālais izpludināšanas filtrs M_H,
3. NH-CMF2 aproksimācijas filtri.

Vertikālais izpludināšanas filtrs

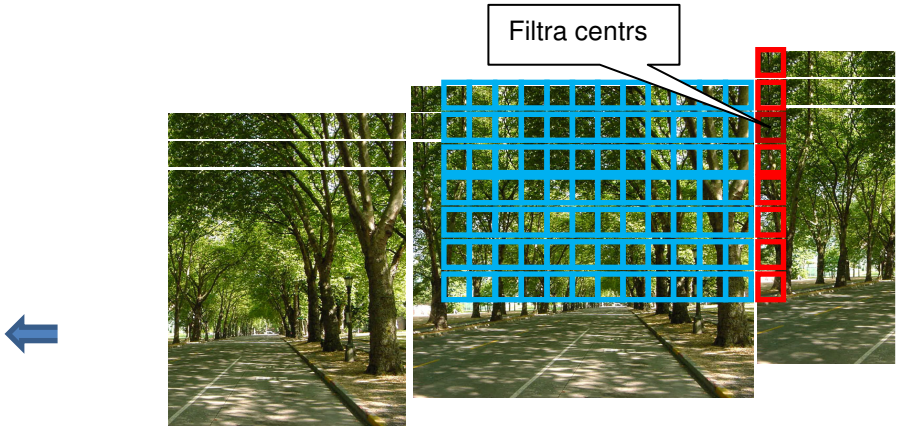
Filtra maskas izmēri ir 1 (horizontālā virzienā) uz L_V (vertikālā virzienā). Filtra atmiņas šūnu struktūra ir sekojoša (atzīmēti ir tikai savienojumi, kas veic attēla papildināšanu, bet neatzīmēti savienojumi starp dažādām rindām):



Viss iepriekš teiktais par nekauzālo filtru realizāciju šajā gadījumā arī ir spēkā, izņemot to, ka šim filtram minētās darbības izpildās veselām rindām, nevis atsevišķiem pikseļiem. Tā, ielādējot pirmo rindu, dati ir jākopē pirmajām E rindām:

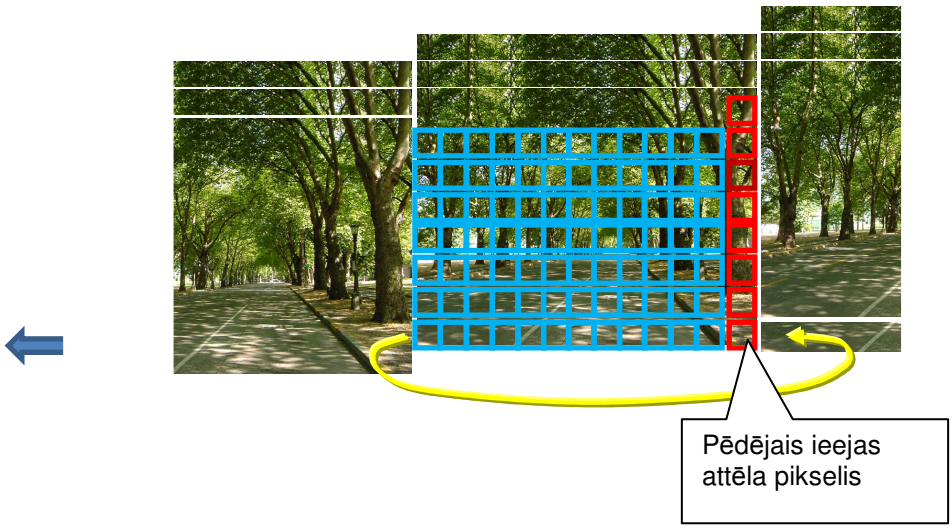


Tālāk, bīdot datus iekšā sagaida momentu, kad visa filtra atmiņa ir aizņemta ar datiem. Šajā momentā (pēc S rindu ignorēšanas filtra izejā) attēla pirmais pikselis nokļūst filtra centrā:

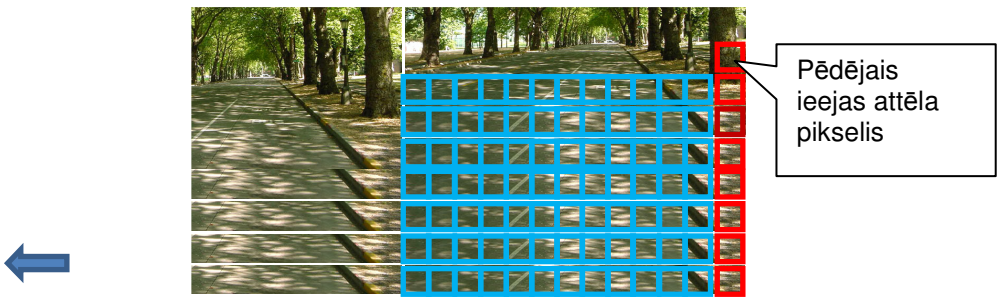


Šis moments ir (S+1) rindas sākums; šajā momentā jāsāk ģenerēt filtrētus datus filtra izejā, kā arī drīkst padot izejā nefiltrētus datus, kurus nolasa no filtra atmiņas centra. No šī brīža filtru simulē kā kauzālo.

Kad tiek iegūts pēdējais ieejas attēla pikselis ir jāveic pēdējas rindas kopēšanu:



Šo operāciju izpilda S rindām, līdz visi attēla dati netiek izbīdīti ārā (moments, kad pēdējais attēla pikselis nokļūst filtra centrā):



Šī situācija noslēdz viena kadra apstrādi vertikālajā filtrā, līdz ar to tas pārslēdzas uz sākotnēju stāvokli un sāk visu no jauna.

Turpmākie plāni:

- Horizontālā izsmērēšanas filtra implementācija projektā;
- NH-CMF2 aproksimācijas implementācija projektā.

Secinājumi

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

Sejas un plaukstas attēlu iegūšanas metožu izpētē turpmāk tiek plānots izveidot sākuma punkta noteikšanas algoritma izstrāde MATLAB vidē, plaukstas ģeometrisko izmēru metodes aprakstīšana un algoritma izstrāde un eksperimenti ar esošo datubāzi algoritma darbības pārbaudei, uzlabošanai.
Sejas atpazīšanas sistēmas izveidei tiek plānota MATLAB koda pārveidošana uz C++ kodu, lai programma var darboties patstāvīgi bez MATLAB, programmas ieviešana iegultā sistēmā, cilvēka attēla iegūšana no kameras un pārbaudes testi, kā arī TMDXEVM6678L attēla apstrādes programmas pielietošana sejas atpazīšanas algoritmu realizācijā.
Biometrisko datu iegūšanas algoritmu paralelizācijas un implementēšanas programmējamos loģiskos masīvos aktivitātes ietvaros nākotnē paredzēts izpētīt filtra maskas parametru ietekmi uz filtru, turpināt darbu pie dubultā leņķa samazināšanas metodes un novērst problēmas, ko radīja šķidrā kristāla displeja kontroliera izveide FPGA platformā.
Biometrisko datu kriptēšanas, glabāšanas un lietojama apakšsistēmas izveides aktivitātē turpmāk tiek plānots pilnveidot un papildināt izveidoto programmu, novēršot kļūdas, kas rodas darba procesā.
Tehnoloģijas demonstratora eksperimentālā maketa montēšanas apakšaktivitātē maketa pirmās versijas detalizēts izveides apraksts ir sniegts iepriekšējā progresa pārskatā. Turpmāk veiktie iekārtas papildinājumi un uzlabojumi tiks aprakstīti nākošajos progresa pārskatos.
Algoritmu implementēšana eksperimentālajā maketā notiks arī turpmāk, nākotnē paredzēts projektā implementēt horizontālo izsmērēšanas filtru un veikt NH-CMF2 aproksimāciju.

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. Sestajā progresa pārskata periodā notika sanāksmes, kurās tika pārrunāti dažādi jautājumi saistībā ar projektu, svarīgākie no tiem: LCD darbība biometrijas iekārtā; CORDIC realizācija FPGA; leņķa samazināšanās problēmas; sejas atpazīšanas algoritmu implementēšana iegultā sistēmā; demonstratora izveides gaita; plaukstas ģeometrijas parametru noteikšana, u.c. jautājumi.
Iepriekšējā laika periodā tika sagatavots zinātniskais raksts par projektā veiktajām aktivitātēm, kas saistītas ar biometrijas datu iegūšanu, apstrādi un šifrēšanu. Raksts „Biohashing and Fusion of Palmprint and Palm Blood Vein Biometric Data" prezentēts starptautiskā IEEE organizētā konferencē par plaukstas biometrisko parametru izmantošanu personu atpazīšanā (ICHB2011), Honkongā. Šajā pārskata periodā Projekta izpildītāji piedalījās Starptautiskajā izgudrojumu izstādē „MINOX 2012", kurā ieguva „Dienas Biznesa" speciālbalvu.
Turpinās darbs pie aktivitātes „Intelektuālā īpašuma tiesību aizsargāšana", līguma ar patentpilnvaroto SIA „PĒTERSONA PATENTS" ietvaros ir sagatavots un iesniegts starptautiska patenta pieteikums.
Ir noteikti nākamā perioda pētniecisko darbu uzdevumi un sasniedzamie rezultāti. Par projekta pētnieciskiem sasniegumiem tiek plānots informēt arī piedaloties ar referātiem un publikācijās konferencēs un semināros.