Proiect AC

Optical Flow

1 Introducere .

Proiectul propus are ca scop investigarea si implementarea in timp real al algoritmilor de flux optic Lucas-Kanade (LK) si Horn-Schunck (HS) pentru estimarea miscarii in secvente video.In urmatoarea lucrare ne propunem sa:

1. Prezentam teoretetic cei doi algoritmi.
2. Analizam literatura de specialitate privind implementarile hardware existente pe FPGA si platforme de simulare
3. Vedem care sunt compromisurile intre acuratete,latenta, consum de resurse si putere
4. Identificam metodele eficiente de mapare pe hardware(precizie numerica, structura de pipeline, abordari multi-scale)
5. Propunem o arhitectura HDl/FPGA adecvata pentru executia in timp real a fluxului optic.

Prin acest proiect urmarim realizarea unui sistem de estimare a miscarii intr-o secventa video ybde fluxul video este prelucrat cadru cu cadru, iar pentru fiecare pixel se calculeaza un vector de deplasare care descrie directia si amplitudinea miscarii intre doua cadre consecutive.

Mai departe ne dorim sa simulam conceptual un lant complet de procesare pentru flux optic, similar cu cel utilizat in aplicatii de viziune artificiala embedded(Ex de procese care folosesc fluxul optic: monitorizare video, navigatie pentru roboti sau astistenta la conducere) dar limitata la nivel de analiza si proiectare. Vom descrie etapele principale de prelucrare:

- Calculul gradientelor spatiale$I_x I_y$ si temporaleI_t ale intensitatii
- Aplicarea schemelor numerice ale algoritmilor: LK si HS
- Obtinerea unui camp de flux optic dens/semi-dens

Si vom analiza rezultatele incercam sa analizam corectitudinea lor si fezabilitatea hardware(latenta,paralelizare,precizie numerica).

Din punct de vedere algoritmic, proiectul isi propune sa clarifice conditiile in care metodele Lucas–Kanade si Horn–Schunck pot furniza o estimare stabila si coerenta a

miscarii (presupuneri, avantaje, limitari) si sa evidentieze diferentele dintre o abordare locala si una globala a fluxului optic.

Din punct de vedere arhitectural vom explica cum ar putea fi organizat un sistem capabil de procesare a imaginii video de mari dimensiuni(Full-HD/4K) in timp real supus la anumite constrangeri:

-De timp:Sa se finalizeze operatiile de preprocesare, calcul de grafient si actualizare a campului de misacre si generarea rezultatelor in intervalul dintre doua cadre succesive.

-De resurse:Sa se incadreze in limitele rezonabile ale unui FPGA tipic, sa foloseasca eficient memoria interna, blocurile aritmetice si logica disponibila.

-De acuratete:Sa mentina un nivel acceptabil de precizie numerica gestionand compromisurile dintre numarul de biti, stabilitatea algoritmilor si complexitatea hardware.

2.Prezentarea Problemei

Fluxul optic descrie modul in care intensitatea unui pixel dintr-un cadru se modifica in timp datorita miscarii obiectelor sau a camerei. Practic, pentru doua cadre consecutive ale unui videoclip, se presupune ca intensitatea unui punct ramane constanta (1) pe durata unei deplasari mici, iar aceasta proprietate permite formularea problemei sub forma unei ecuatii locale.

(1) $I(x, y, t) = I(x + u, y + v, t + \Delta t)$

Aceasta ecuatie este subdeterminată (fenomenul „aperture problem”), ceea ce face necesara utilizarea unor metode suplimentare pentru a obtine o solutie stabila.

In contextul procesarii video apar urmatorii factori:

1. **Sensibilitatea la zgomot**: gradientele de intensitate si derivata temporala trebuie calculate in conditii de variatii puternice ale imaginii.
2. **Dimensiunea datelor**: un singur cadru poate avea milioane de pixeli, iar pentru fiecare trebuie rezolvat sistemul de ecuatii al fluxului optic.
3. **Constrangerea de timp real**: pentru rezolutii mari (HD, Full HD, 4K), algoritmii trebuie adaptati pentru a oferi valori la 30–60 fps, unde sistemul dispune de doar 16–33 ms pentru procesarea unui cadru complet.

Lucas–Kanade trateaza problema la nivel local, presupunand ca intr-o mica fereastra miscarea este constanta. Se obtine astfel un sistem de ecuatii ce poate fi rezolvat prin metode de minimizare locala. ⇔ cu rezolvarea sistemului de 2x2 pentru fiecare pixel

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

In schimb, Horn–Schunck abordeaza problema global, introducand un termen de regularizare care impune netezime vectorilor de miscare in intreaga imagine.

$$E(u,v) = \iint (I_x u + I_y v + I_t)^2 + \alpha^2(|\nabla u|^2 + |\nabla v|^2)\, dx\, dy$$

Prin aceasta abordare, algoritmul HS produce rezultate uniforme si stabile, dar necesita iteratii multiple pentru convergenta, la un pas k trebuie sa stim termenii precedent si sa facem ecuatiile : $u^{(k+1)} = \bar{u}^{(k)} - \frac{I_x(I_x\bar{u}^{(k)} + I_y\bar{v}^{(k)} + I_t)}{\alpha^2 + I_x^2 + I_y^2}$ si $v^{(k+1)} = \bar{v}^{(k)} - \frac{I_y(I_x\bar{u}^{(k)} + I_y\bar{v}^{(k)} + I_t)}{\alpha^2 + I_x^2 + I_y^2}$ unde $\bar{u}$, $\bar{v}$sunt mediile vecinilor.

In cadrul implementarii hardware, ambele metode depind de operatii regulate pe vecinatati de pixeli (ferestre pentru LK, vecini directi pentru HS) si necesita calculul gradientilor in timp si spatiu. Aceste operatii pot fi mapate eficient pe un FPGA datorita naturii lor masiv paralele, insa apar provocari legate de:

- tipul reprezentarii numerice utilizate (virgula fixa sau flotanta);
- volumul mare de accesari in memorie pentru ferestre si vecinatati;
- necesitatea respectarii unui pipeline stabil pentru procesarea curentului video;
- sincronizarea corecta a etapelor de preprocesare, calcul al fluxului si generare a rezultatelor.

Atat LK, cat si HS necesita:

- acces rapid la vecinatati (ferestre 5×5 pentru LK, vecini directi pentru HS);
- calculul gradientelor spatiale si temporale cu filtre tip Sobel:

$$I_x = I * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, I_y = I * \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- sincronizarea corecta a etapelor de preprocesare;
- alegerea reprezentarii numerice (Q4.12in fixed-point).

Aceste operatii trebuie implementate intr-un pipeline stabil capabil sa sustina un flux video continuu, fara pierdere de cadre.

3.Fundamente Teoretice

Fluxul optic se bazeaza pe ideea ca intensitatea unui punct din imagine ramane aproximativ constanta pe o perioada scurta de timp, chiar daca obiectele sau camera se deplaseaza. Pe aceasta presupunere se construiesc toate metodele diferentiale utilizate pentru estimarea miscarii intre doua cadre consecutive

3.1. Ipoteza de constanta a intensitatii

Se considera ca pentru un pixel localizat in (x, y)la timpul t, intensitatea acestuia nu se schimba atunci cand se deplaseaza cu o mica valoare (u, v)catre un nou cadru. Formal:

$$I(x, y, t) \approx I\big(x + u_J y + v, t + 1\big)$$

Aceasta relatie este valabila doar pentru miscari mici si fara schimbari bruste de lumina sau reflexii.

Pentru a extrage informatia de miscare, relatia este linearizata, rezultand o ecuatie diferentiala care implica gradientele imaginii. Aceasta ecuatie contine trei marimi (I_x, I_y, I_t) dar doar doua necunoscute (u, v). Din aceasta cauza problema este „subdeterminată” si nu poate fi rezolvata fara constrangeri suplimentare.

3.2. Calculul gradientelor spatiale si temporale

Gradientele imaginii sunt necesare pentru orice metoda diferenciala de flux optic.

- **Gradientul spatial** (variatia intensitatii in directia x si y) este utilizat pentru a determina zonele in care exista textura suficienta pentru estimarea miscarii.
- **Gradientul temporal** (schimbarea intensitatii intre doua cadre) ofera informatia despre variatia in timp.

Aceste marimi sunt obtinute prin aplicarea unor filtre discrete (de exemplu filtre Sobel sau diferente finite). Gradientele trebuie sa fie stabile numeric si sa elimine zgomotul fara a distruge detaliile esentiale.

3.3. Problema „aperture” si imposibilitatea unei solutii directe

Ecuatia fluxului optic:

$$I_x u + I_y v + I_t = 0$$

nu are solutie unica pentru (u, v), deoarece un singur punct nu ofera suficiente informatii despre directia reala a miscarii.
De exemplu, intr-o zona cu o muchie verticala, miscarea in directia muchiei nu poate fi distinsa.

Din acest motiv, metodele diferentiale introduc **restrictii suplimentare**:

- Lucas–Kanade: presupune miscare constanta intr-o vecinatate mica.
- Horn–Schunck: impune netezime globala a campului de miscare.

3.4. Clasificarea metodelor diferentiale

Metodele diferentiale de flux optic sunt impartite in doua mari categorii:

1. **Metode locale** (ex. Lucas–Kanade):
    - analizeaza o mica fereastra in jurul fiecarui pixel;
    - rezolva un sistem de ecuatii prin least-squares;
    - functioneaza bine in zone cu gradient puternic.
2. **Metode globale** (ex. Horn–Schunck):
    - construiesc un camp de miscare pe intreaga imagine;
    - utilizeaza un termen de regularizare pentru netezime;
    - produc un flux optic dens si coerent.

3.5. Necesitatea preprocesarii si a stabilizarii numerice

Pentru ca fluxul optic sa fie calculat corect, sunt necesare cateva etape prealabile:

- **Filtrarea imaginii**: reducerea zgomotului inainte de calcularea gradientelor.
- **Normalizarea intensitatii**: evita amplificarea erorilor numerice in zone luminoase sau intunecate.
- **Multi-scale (piramide Gaussiene)**:
    - se estimeaza mai intai miscarea pe o imagine redusa (la scara mica),
    - apoi rezultatele se rafineaza pe niveluri superioare.
      Aceasta abordare imbunatateste stabilitatea si permite detectarea miscarilor rapide.

3.6. Fundamentul comun pentru implementarea hardware

Indiferent de algoritm, toate metodele diferentiale necesita:

- acces la vecinatati de pixeli (ferestre sau kernel-uri);
- operatii repetitive, uniforme, ideale pentru implementare paralela;
- un pipeline de calcul stabil (gradient → actualizare miscare → post-procesare);

- o reprezentare numerica adecvata (fixed-point este preferata pe FPGA).

Prin aceste aspecte, sectiunea teoretica pregateste fundalul necesar pentru a intelege cum sunt construite algoritmii Lucas–Kanade si Horn–Schunck si de ce pot fi mapati eficient in hardware.

4.Algoritmul Lucas–Kanade

Ideea principala a algoritmului este: in interiorul unei mici ferestre in jurul fiecarui pixel, miscarea este aproximata ca fiind constanta. Aceasta presupunere permite rezolvarea problemei fluxului optic prin analizarea unui set de ecuatii derivate din gradientele imaginii.

4.1Cum functioneaza

Folosim relatia : $I_x u + I_y v + I_t = 0$ pentru un pixel aflat intr-o vecinatate W dar aceasta nu este suficienta pentru determinarea vectorului (u, v) Dar, daca presupunem ca toti pixelii din fereastra au aceeasi deplasare, putem combina ecuatiile tuturor pixelilor si obtinem un sistem supradeterminat.

In fereastra W(tipic 3×3, 5×5 sau 7×7), gradientele spatiale si temporale genereaza urmatorul sistem:

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

Matricea:

$$A = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

este cunoscuta ca **matricea de structura** a ferestrei si determina cat de „informata” este zona pentru estimarea miscarii. Daca $A$are determinant mare, fereastra contine variatii in ambele directii (muchii 2D, colturi) si fluxul optic poate fi estimat stabil. Daca determinantul este aproape zero, fereastra contine doar textura intr-o singura directie si apare „aperture problem”.

Prin inversarea matricei A obtinem solutia:

$$(u, v) = -A^{-1} \cdot b$$

unde:

$$b = \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

4.2 conditii pentru o estimare corecta

Pentru ca Lucas–Kanade sa functioneze bine, este necesar ca:

- fereastra analizata sa contina **textura** (variatii puternice ale intensitatii);
- gradientele sa nu fie dominate de zgomot;
- variatiile de intensitate sa fie mici intre cadre;
- numarul de pixeli din fereastra sa fie suficient pentru a stabiliza sistemul.

In zone uniforme, fara textura, algoritmul nu poate stabili directia miscarii.

Pentru miscari mari sau scene cu rezolutii ridicate, versiunea clasica LK nu este suficienta. De aceea se foloseste metoda *pyramidal Lucas–Kanade*, care functioneaza astfel:

1. Imaginea este scalata in mai multe niveluri (piramida Gaussiana).
2. Fluxul optic se calculeaza intai pe nivelul cel mai mic (rezolutie redusa).
3. Vectorii calculati sunt propagati si rafinati pe nivelurile superioare.
4. Rezultatul final este un camp optic stabil si capabil sa capteze miscari rapide.

Abordarea multi-scale este foarte importanta in implementarile hardware, deoarece reduce cerintele de memorie si accelereaza convergenta.

4.3 Avantaje si limitari:

Avantaje:

i. functioneaza bine in zone cu textura;

ii. calculele sunt locale → usor de paralelizat pe FPGA;

iii. necesita doar rezolvarea unui sistem 2×2 pentru fiecare pixel.

Limitari:

i. esueaza in zone cu gradient slab;

ii. sensibil la zgomot;

iii. implica ferestre relativ mari, ceea ce creste consumul de memorie;

iv. inversarea matricei 2×2 pentru fiecare pixel poate deveni costisitoare in hardware.

4.4Plan pentru implementare hardware

Pentru implementarea pe FPGA, Lucas–Kanade necesita:

- **line buffers** pentru pastrarea ferestrelor 5×5 sau 7×7;
- module dedicate pentru calculul sumelor $\sum I_x^2, \sum I_y^2, \sum I_x I_y, \sum I_x I_t, \sum I_y I_t$;
- o unitate aritmetica capabila sa calculeze inversarea unei matrice 2×2;
- un pipeline stabil pentru procesarea fiecarui pixel la flux video.

LK este eficient in aplicatii unde avem nevoie de miscare locala precisa, insa devine dificil de scalat la rezolutii foarte mari din cauza ferestrelor de dimensiuni mari si a inversarii matricei pentru fiecare pixel.

5. Algoritmul Horn–Schunck

Algoritmul Horn–Schunck este o metoda globala de estimare a fluxului optic. Spre deosebire de Lucas–Kanade, care analizeaza doar informatia dintr-o fereastra locala, Horn–Schunck incearca sa calculeze un camp de miscare *dens si uniform* pentru toata imaginea. Ideea centrala este introducerea unui termen de netezime care obliga vectorii de miscare din zonele apropiate sa fie similari, eliminand variatiile bruste si instabilitatea.

Algoritmul porneste de la aceeasi ecuatie de constanta a intensitatii:

$$I_x u + I_y v + I_t = 0$$

dar adauga si o conditie suplimentara:

u si v trebuie să fie cât mai netede în întreaga imagine.

Pentru atingerea acestui obiectiv, Horn–Schunck minimizeaza o functie de energie care combina informatia datelor cu netezimea campului optic.

5.1 Cum functioneaza

Horn–Schunck rezolva fluxul optic printr-o schema iterativa. Pasii sunt:

1. Se calculeaza gradientele I_x, I_y, I_tpentru fiecare pixel.
2. Pentru fiecare pixel, se calculeaza media vecinilor pentru u si v

(media celor 4 sau 8 vecini).

3. Se actualizeaza u si v folosind formula de corectie:

$$u = \bar{u} - I_x \cdot \frac{I_x\bar{u} + I_y\bar{v} + I_t}{\alpha^2 + I_x^2 + I_y^2}$$
$$v = \bar{v} - I_y \cdot \frac{I_x\bar{u} + I_y\bar{v} + I_t}{\alpha^2 + I_x^2 + I_y^2}$$

4. Se repeta acest proces timp de un numar fix de iteratii (10–100 in functie de precizie).

Parametrul αcontroleaza cat de neteda este solutia finala:

- αmare → camp foarte uniform;
- αmic → mai sensibil la detalii, dar zgomotos.

5.2 Conditii pentru o estimare corecta

Pentru ca algoritmul Horn–Schunck sa returneze un camp optic stabil, trebuie indeplinite cateva conditii:

- gradientele spatiale sa fie calculate corect (filtrare impotriva zgomotului);
- variatiile dintre cadre sa fie moderate (metoda este sensibila la schimbari bruste);
- numarul de iteratii sa fie suficient pentru convergenta;
- parametrul αsa fie ales corespunzator (prea mic → zgomot, prea mare → supra-netezire).

HS nu depinde de textura unei ferestre ca LK, dar depinde de calitatea intregului camp de gradient.

5.3 Avantaje si limitari

Avantaje:

i. produce un camp optic **dens** (vector pentru fiecare pixel);
ii. nu depinde de inversarea matricii ca LK → calcule mai simple;
iii. foarte potrivit pentru implementare iterativa si paralelizata;
iv. rezultatele sunt netede si coerente pe intreaga imagine;

v. se poate combina usor cu abordari multi-scale.

Limitari:

i. necesita mai multe iteratii → timp de calcul mai mare;

ii. sensibiliate la iluminare, zgomot sau gradient slab;

iii. poate „intinde” miscarea in zone fara informatie (regularizare prea puternica);

iv. depinde de alegerea parametrului αsi de numarul de iteratii.

5.4 Plan pentru implementare hardware

Horn–Schunck este mult mai prietenos cu FPGA decat Lucas–Kanade datorita schemei iterative simple si accesului regulat la vecini. Pentru implementarea pe hardware sunt necesare:

- **line buffers** sau **frame buffers** pentru accesul la vecinii unui pixel;
- un modul care calculeaza media locala $(\bar{u}, \bar{v})$;
- unitati aritmetice pentru termenul de corectie $I_x\bar{u} + I_y\bar{v} + I_t$;
- o bucla de iteratie hardware (pipeline sau iterare pe linii);
- memorie interna pentru stocarea campurilor u si v la iteratia curenta.

Spre deosebire de LK, Horn–Schunck NU are nevoie de inversari de matrice sau de ferestre mari, ci doar de acces la vecinii directi.
Acest lucru il face mult mai scalabil pentru rezolutii mari (Full HD / 4K), deoarece operatiile se repeta uniform pentru fiecare pixel si se preteaza bine la paralelizare masiva.

5. Review al literaturii

În cadrul acestui capitol, ne propunem să analizăm stadiul actual al tehnologiei în domeniul estimării mișcării prin flux optic. Analiza se va concentra pe două lucrări de referință din bibliografie care abordează perspective complementare: optimizarea algoritmilor clasici (Horn-Schunck și Lucas-Kanade) pentru sisteme embedded de tip FPGA și tranziția către noile paradigme de procesare a imaginilor folosind senzori bazați pe evenimente (neuromorfici). Scopul este de a înțelege fundamentul teoretic, constrângerile fizice și soluțiile algoritmice moderne.

5.1 https://www.mdpi.com/1424-8220/22/13/5017

Lucrarea prezintă **prima implementare eficientă în timp real** pe FPGA a algoritmilor **Lucas–Kanade (LK)** și **Horn–Schunck (HS)** *în versiuni multi-scale*, capabilă să proceseze **video 4K @ 60 fps** cu **consum sub 6 W**.

In aceasta lucrarea sunt prezentate si enumerate mai multe modalitati de a genera acel vector field ce descrie miscarea unui pixel de la o imagine la alta , imagini ce fac parte din aceeasi secventa . Majoritatea se bazeaza pe suma diferenetelor patratice intr-o vecinatate mica ce se poate scala la o vecinatate mai mare , unele abordari au fost chiar si in domeniul frecventei folosind filtre Gabor si Transformata Fourier (nu foarte eficiente din cauza costului mare computational) , iar metodele mai recente se bazeaza in mare pe detectarea si maparea trasaturilor ca mai apoi sa se minimizeze anumite functii cu metode de tip gradient-based algorithms . In aceasta lucrare stintifica sunt prezentate doua metode de baza de la care au evoluat altele mai apoi , ci anume **Horn-Schunck** si **Lucas-Kanade** .

Pentru a obtine valori corecte ale fluxului optic folosind algoritmi de flux optic bazati pe gradient trebuie indeplinite anumite ipoteze . Prima este deplasarea mica a obiectului intre doua cadre consecutive care este de obicei indeplinita pentru obiectele care se misca lent . In alte cazuri anumite modificari trebuie facute – cresterea fps – urilor pentru device-ul de achizitie a datelor / imaginilor sau folosirea unei metode de tip multi-scale . A doua ipoteza este considerarea ecuatiei de luminozitate a pixelilor ca fiind o constanta pentru doua cadre consecutive de imagini . Ecuatia pentru doua cadre consecutive in momentele de timp t si t+1 care descrie luminozitatea pixelilor pentru o imagine I este urmatoarea :

$$I(x(t), y(t), t) = I(x(t+1), y(t+1), t+1) = const$$

$$\frac{dI(x(t), y(t), t)}{dt} = 0$$

Practic aceasta ecuatie impune faptul ca intensitatea pixel-ului la o anumita locatie (x,y) la timpul t este egala cu intensitatea pixel-ului la noua locatie (x+δx , y+δy) la momentul t+ δt , unde $\delta x \; si \; \delta y$ reprezinta deplasarile pe axe ale pixel-ului .

Cand ecuatia este extinsa in serie Taylor o aproximare se poate face , omitand derivatele de ordin superior de 1 , astfel conducand la ecuatia initiala in metodele de flux optic bazate pe gradient :

This equation can be written in a more compact form (Equation (4)), using the pixel offset defined as (u, v).

$$I_x u + I_y v + I_t = 0 \quad (4)$$

Mai multe metode au fost construite din acest punct , unele bazandu-se pe rezolvarea globala a imaginii , iar altele folosind metode locale – practic rezolvand ecuatia numai intr-o vecinatate aleasa .

Horn-Schunck :

Pentru aceasta metoda s-a mai propus o ipoteza adaugata la ecuatia de mai sus , ci anume ca fluxul optic ar trebui sa fie lin peste toata imaginea . Prin urmare , pentru fiecare pixel , fluxul calculat este similar intr-o vecinatate mica . Astfel s-a propus ecuatia de mai jos care are doua componente : prima este responsabila cu constanta de luminozitate a unui pixel (ca si in ecuatia 4) , iar a doua este responsabila cu regularizarea fluxul-ui pe toata imaginea (efectele regularizarii pot fi controlate prin constanta α) .

$$E = \iint [(I_x u + I_y v + I_t)^2 + \alpha^2(||\nabla u||^2 + ||\nabla v||^2)]dxdy \tag{5}$$

Algoritmul HS este unul global prin care minimul global al functiei poate fi cautat printr-o maniera iterativa folosind formulele :

$$u_{n+1} = \overline{u_n} - \frac{I_x(I_x\overline{u_n} + I_y\overline{v_n} + I_t)}{\alpha^2 + I_x^2 + I_y^2} \tag{6}$$

$$v_{n+1} = \overline{v_n} - \frac{I_y(I_x\overline{u_n} + I_y\overline{v_n} + I_t)}{\alpha^2 + I_x^2 + I_y^2} \tag{7}$$

where:
$\overline{u_n}, \overline{v_n}$—average velocity in the neighbourhood.

Lucas-Kanade :

O abordare diferita , in loc sa caute minimul global , metoda se uita intr-o mica vecinatate a unui pixel . Observatia este facuta pe baza faptului ca un pixel se muta o data cu vecinii lui . Practic pentru n pixeli numerotati $p_1, p_2, \ldots . p_n$ dintr-o vecinatare apropiata , ecuatia de intensitate a luminii poate fi scrisa sub forma matriciala :

$$\begin{pmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = - \begin{pmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_n) \end{pmatrix} \tag{8}$$

Pentru a simplifica notatiile s-a utilizat urmatoarea forma :

$$A = \begin{pmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{pmatrix}, \quad d = \begin{pmatrix} u \\ v \end{pmatrix}, \quad b = - \begin{pmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_n) \end{pmatrix} \tag{9}$$

Aceasta forma conduce la urmatoarea problema de optimizare , ci anume :

$$min \, \|Ad - b\|^2$$

O solutie a acestei ecuatii este data de urmatoarea forma :

$$\begin{pmatrix} u \\ v \end{pmatrix} = (A^T W A)^{-1} A^T W b$$

In aceasta metoda W reprezinta matricea de ponderi sau greutati . In contextul metodelor bazate pe gradient pentru fluxul optic , scopul principal al matricei W este de a acorda o importanta diferita (o pondere) pixelilor inclusi in acea fereastra . Pixelii apropiata de pixel-ul principal sunt considerati mai importanti in timp ce aceia mai departati sunt considerati mai putini importanti . Noi putem considera ca si optiune de implementare pentru matricea W o matrice unitate I sau pentru o acuratete buna putem folosi o Masca Gaussiana unde ponderile scad exponential pe masura ce pixeli se departeaza de pixelul principal .

Multi-scale method :

In multe aplicatii practice , algoritmii de calcul pentru fluxul optic nu sunt suficienti din cauza obiectelor ce se misca prea repede sau din cauza insuficientii ratei de achizitie a datelor . Astfel , este introdusa metoda de multi-scalare :

Mai intai se efectueaza constructia de baza a piramidei de imagini . Pentru fiecare dintre cele doua imagini (cadrul anterior si cadrul curent) se construieste o piramida de imagini.

Aceasta inseamna ca fiecare cadru este redimensionat (micsorat) de mai multe ori . De obicei , imaginea este micsorata de doua ori intre scarile adiacente (scalare) . Procesul se repeta de cateva ori (de obicei de 3 ori) , rezultand 4 scari in total .

Apoi intervine procesul iterativ de calcul . Calculul fluxul optic progreseaza de la cea mai mica rezolutie pana la rezolutia initiala (baza piramidei) . Fluxul optic este determinat pentru perechea de imagini aflate la cea mai mica scara (varful piramidei) . La aceasta scara mica , fluxul optic ofera o informatie bruta (grosiera) despre deplasarea generala a obiectului .

Apoi urmeaza pasul cel mai important (upscaling si warping) , unde fluxul rezultat este apoi marit la dimensiunea scarii imediat mai mari . Atat dimensiunile campului vectorial (matricea de flux) cat si valorile vectorilor de deplasare (u ,v) sunt , de obicei , dublate .

Warping ul inseamna deformarea cadrului anterior conform acestui flux optic upscalat , aceasta procedura reprezentand o compensare a miscarii si asigurarea ca , in scarile mai mari , deplasarea ramasa a pixelilor este mica , astfel incat ipoteza de deplasare sa nu fie incalcata .

Ca si ultim pas avem de rafinat fluxul , folosind cadrul anterior modificat si cadrul curent micsorat (la scara curenta , mai mare) , fluxul optic este calculat din nou . Intregul ciclu de calcul se repeta pana cand fluxul optic este calculat pentru perechea de imagini la dimensiunea lor initiala .

5.2 https://ieeexplore.ieee.org/document/9774999

In acest document , pe langa introducerea care este reprezentata de Fundamentele Fluxului Optic si Noile Tehnologii de Viziune despre care am discutat si in celalalt document , sunt prezentate urmatoarele :

Tranzitia catre senzori bazati pe evenimente :

Camerele traditionale cu cadre fixe nu sunt adecvate pentru a capta dinamica rapida a lumii reale . Senzorii de viziune bazati pe evenimente (neuromorfici) ofera o alternativa superiora .

- Caracteristici :

Acesti senzori produc un flux de evenimente asincron , fiecare eveniment fiind generate doar atunci cand intensitatea luminii se schimba la nivelul unui pixel . Ei ofera o rezolutie temporala extrem de inalta si un interval dinamic larg .

Aceasta natura asincrona face ca algoritmii clasici de flux optic sa nu poata folosi din plin avantajele acestor senzori , impunand dezvoltarea unor metode specifice care sa gestioneze fluxul de date bazate pe evenimente .

- Provocarea principala : Problema Aperturii

Una dintre cele mai mari dificultati in cadrul fluxului optic , indiferent de tipul de camera, este Problema Aperturii . Aceasta apare atunci cand calculul fluxului este efectuat doar pe o regiune locala (o apertura mica) a unui obiect in miscare . In astfel de cazuri , algoritmul nu poate determina directia reala a miscarii obiectului , ci returneaza doar o

estimare a miscarii care este normala (ortogonala) pe conturul local . Metodele timpurii de flux optic bazate pe evenimente nu au reusit sa rezolve eficient aceasta problema .

Solutia algoritmica : de la ARMS la fARMS

Pentru a rezolva Problema Aperturii într-un mod robust și adaptat senzorilor pe bază de evenimente, a fost propus algoritmul **ARMS (Aperture Robust Multi-Scale flow)**.

Principiul de baza al ARMS (Multi-scale Pooling) :

Pentru fiecare eveniment generat se calculeaza un flux local a carui directia este ortogonala pe conturul local . Se utilizeaza o serie de η ferestre spatiale (aperturi) de dimensiuni crescatoare , centrate pe evenimentul curent . Pentru determinarea fluxului real , se selecteaza acea fereastra spatiala pentru care magnitudinea medie a vectorilor de flux local este maxima . Teoretic, maximizarea magnitudinii medii a fluxului local este echivalentă cu minimizarea erorii față de **fluxul adevărat** al obiectului $(\boldsymbol{U})$. Estimarea finala reprezinta fluxul mediu calculat in acea fereastra optima .

Optimizare software : fARMS (faster ARMS)

Algoritmul ARMS original suferea de ineficiențe, în special din cauza calculelor repetitive și a dependenței sale de un "event frame" (cadru de evenimente), care nu se potrivea cu natura asincronă a datelor .

Cercetarea propune versiunea optimizată **fARMS**, care aduce următoarea îmbunătățire majoră:

- **Renunțarea la "Event Frame":** S-a abandonat utilizarea cadrului de evenimente în favoarea unui **Recent Flow event Buffer (RFB)**, un buffer circular care stochează ultimele N evenimente valide generate .
- **Reducerea Complexității:** Această modificare a permis o schimbare a modului de calcul: în loc să se itereze pe o regiune spațială mare, se iterează doar pe cele N evenimente din buffer .
    - **Complexitatea ARMS:** Era $O(w_m^2 \cdot \eta)$(dependentă pătratic de dimensiunea maximă a ferestrei w_m și, implicit, de rezoluția senzorului) .
    - **Complexitatea fARMS:** Este O(N · η) (dependentă doar de lungimea bufferului N și de numărul de ferestre .

Această optimizare a redus teoretic complexitatea de calcul cu până la **98.96%** (pe baza unei configurații de referință), permițând deja performanță în timp real pe platforme software obișnuite în majoritatea scenariilor .

6. Consideratii de Implementare FPGA

Implementarea algoritmilor de flux optic pe o arhitectură hardware reconfigurabilă (FPGA) impune o schimbare fundamentală de paradigmă față de execuția software secvențială: trecerea la o procesare masiv paralelă, orientată pe fluxuri de date (stream processing), capabilă să gestioneze constrângerile stricte de timp și resurse impuse de rezoluțiile video moderne (Full HD / 4K).

6.1. Arhitectura memoriei: Buffere de linie și acces paralel la vecinătăți

Principalul obstacol în procesarea video în timp real nu este puterea brută de calcul, ci **lățimea de bandă a memoriei** și **latența accesului la date**. Deoarece atât algoritmul Lucas–Kanade, cât și Horn–Schunck depind critic de informația spațială locală (gradientul într-un punct și media vecinilor), citirea permanentă din memoria externă (DRAM) pentru fiecare pixel din fereastra de procesare ar produce rapid o **gâtuire majoră** (bottleneck).

Soluția este implementarea unui sistem de **Line Buffers** (buffere de linie) folosind memoria internă a FPGA-ului (BRAM), care permite stocarea temporară a celor **N–1 linii video anterioare**. Această structură susține un mecanism de **fereastră glisantă** (Sliding Window), esențial pentru aplicarea filtrelor de convoluție — de exemplu operatorul Sobel pentru calculul gradientelor spațiale I_x, I_y.

Prin această arhitectură, toți pixelii unei ferestre (de exemplu 5×5pentru LK sau vecinătatea locală pentru HS) devin accesibili **simultan, într-un singur ciclu de ceas**, permițând calculul gradientelor spațiale și temporale fără a întrerupe fluxul video continuu.

6.2. Pipeline-ul de calcul și stabilizarea numerică

Pentru a garanta procesarea la 30–60 fps, arhitectura trebuie să fie complet **pipelined**, astfel încât fiecare etapă de calcul să se finalizeze într-un singur ciclu de ceas. Alegerea algoritmului influențează direct complexitatea acestui pipeline:

Provocarea Lucas–Kanade

Deși eficient local, LK necesită **inversarea unei matrici** 2×2(matricea $A^T A$) pentru fiecare pixel — o operație aritmetică costisitoare și dificil de stabilizat numeric în hardware, mai ales fără unități de virgulă mobilă performante.

Avantajul Horn–Schunck

HS este mult mai scalabil pe FPGA, deoarece **elimină inversarea matricială** și o înlocuiește cu o schemă iterativă bazată pe **medii locale**.

Iterațiile pot fi implementate prin **desfășurare spațială** (loop unrolling), instanțiind multiple unități de procesare în cascadă. Astfel, aproximările succesive ale vectorilor uși $v$sunt rafinate într-o singură trecere a datelor.

Aritmetica în virgulă fixă și precizia numerică

Datorită limitărilor de resurse ale FPGA-ului (număr restrâns de DSP-uri și celule logice), se renunță la virgulă mobilă în favoarea aritmeticii **fixed-point**.
Este necesară o analiză atentă a **formatului numeric** (ex. Q4.12), care trebuie:

- să ofere suficientă precizie pentru variațiile fine ale gradientelor;
- să prevină **overflow-ul** în operațiile de acumulare și ridicare la pătrat.

Integrarea într-o arhitectură multi-scale

Pentru a gestiona mișcările ample, arhitectura trebuie integrată într-o schemă **multi-scale piramidală**, procesând imaginea la rezoluții succesiv reduse — pas critic pentru stabilizarea fluxului optic în aplicații real-time.

7. Arhitectura Implementată: Pyramidal Horn–Schunck

Arhitectura propusă și implementată în cadrul acestui proiect vizează accelerarea hardware a algoritmului **Horn–Schunck** pentru estimarea fluxului optic dens, utilizând o abordare **piramidală pe două niveluri (multi-scale)**. Sistemul este optimizat pentru procesare de tip **streaming**, eliminând necesitatea stocării cadrelor video complete în memoria externă și permițând operarea în timp real.

Abordarea piramidală permite gestionarea deplasărilor mari ale obiectelor, care ar încălca ipoteza de variație mică a intensității, prin estimarea inițială a mișcării la rezoluții reduse și rafinarea progresivă a acesteia la rezoluția maximă.

7.1. Metodologia de Proiectare (PyMTL3)

Spre deosebire de fluxurile clasice de proiectare HDL, arhitectura a fost dezvoltată utilizând **PyMTL3 (Python-based Hardware Generation Framework)**. Această alegere a oferit următoarele avantaje majore:

- Modelarea algoritmului Horn–Schunck la nivel **bit-accurate**, într-un mediu de dezvoltare flexibil și productiv;
- Verificarea rapidă a modulelor matematice prin comparație directă cu modele de referință (**Golden Model**) implementate în Python;
- Generarea automată de cod **Verilog HDL** pentru sinteza și implementarea pe platforme FPGA, precum **Xilinx Artix-7**.

Această metodologie reduce semnificativ timpul de dezvoltare și riscul de erori, permițând iterații rapide asupra arhitecturii hardware.

7.2. Structura Piramidală pe Două Niveluri

Pentru a gestiona eficient mișcările de amplitudine mare, arhitectura implementează o structură **multi-scale** compusă din două ramuri de procesare paralele:

Nivelul Coarse (Grosier – 32 × 32)

Imaginea de intrare este redimensionată folosind modulul **Downsampler**, reducând rezoluția pentru a obține o estimare globală a mișcării. La acest nivel:

- Se calculează un câmp de mișcare brut (coarse flow);
- Se utilizează un parametru de regularizare ridicat,

$$\alpha = 20$$

pentru a asigura stabilitatea vectorilor de mișcare în prezența zgomotului și a informației incomplete.

Nivelul Fine (Fin – 64 × 64)

Câmpul de mișcare obținut la nivelul grosier este extins prin modulul **Upsampler** și utilizat drept **estimare inițială (Initial Guess)** pentru nivelul de rezoluție maximă. La acest nivel:

- Mișcarea este rafinată pentru a surprinde detaliile fine;
- Se utilizează un parametru de regularizare mai mic,

$$\alpha = 5$$

permițând captarea variațiilor locale ale mișcării.

Această strategie îmbunătățește convergența algoritmului și crește acuratețea estimării finale.

7.3. Descrierea Modulelor Hardware

Arhitectura este organizată ierarhic și include următoarele module hardware esențiale:

LineBuffer

Implementează o memorie circulară internă care stochează două linii video consecutive. Aceasta permite extragerea unei ferestre glisante de

3×3

pixeli, necesară calculului gradientelor, fără accesarea memoriei externe (DRAM).

GradientUnit

Calculează în paralel:

- derivatele spațiale I_x și I_y, utilizând operatori **Sobel**;
- derivata temporală I_t, pe baza diferenței dintre cadre consecutive.

Modulul este complet pipelinizat și asigură un **throughput de un pixel pe ciclu de ceas**.

HSCore (Motorul Matematic Horn–Schunck)

Reprezintă nucleul de calcul al sistemului și implementează schema iterativă Horn–Schunck. Actualizarea vectorilor de mișcare u și v se realizează conform relațiilor:

$$u^{(k+1)} = \bar{u}^{(k)} - \frac{I_x\left(I_x\bar{u}^{(k)} + I_y\bar{v}^{(k)} + I_t\right)}{\alpha^2 + I_x^2 + I_y^2}$$

$$v^{(k+1)} = \bar{v}^{(k)} - \frac{I_y\left(I_x\bar{u}^{(k)} + I_y\bar{v}^{(k)} + I_t\right)}{\alpha^2 + I_x^2 + I_y^2}$$

unde $\bar{u}$ și $\bar{v}$ reprezintă mediile locale ale vecinilor.

Downsampler și Upsampler

Module dedicate pentru:

- scalarea rezoluției imaginilor;
- proiecția corectă a vectorilor de mișcare între nivelurile piramidei multi-scale.

7.4. Reprezentarea Numerică și Gestiunea Resurselor

Pentru optimizarea utilizării resurselor hardware disponibile pe FPGA, arhitectura propusă utilizează aritmetică în virgulă fixă (fixed-point). Formatul numeric ales este Q4.12, care include un bit de semn, trei biți pentru partea întreagă și doisprezece biți pentru partea fracționară. Această configurație oferă un compromis adecvat între precizia necesară algoritmului Horn–Schunck și complexitatea hardware asociată implementării.

Alegerea reprezentării fixed-point permite reducerea consumului de resurse și asigură stabilitatea numerică a calculelor iterative, evitând costurile ridicate asociate utilizării aritmeticii în virgulă mobilă pe platforme FPGA.

În urma procesului de sinteză logică realizat cu utilitarul Yosys, au fost obținute următoarele rezultate privind utilizarea resurselor pentru întregul sistem. Arhitectura utilizează un total de 33.187 LUT-uri, incluzând toate tipurile LUT1–LUT6, majoritatea acestora fiind alocate modulelor aritmetice din nucleul HSCore. Numărul total de registre utilizate este de 1.848, acestea fiind dedicate în principal buffere-lor de linie și registrelor de pipeline necesare sincronizării etapelor de procesare. În această sinteză generică, nu au fost utilizate blocuri DSP dedicate, toate operațiile de înmulțire și împărțire fiind mapate pe logica programabilă pentru a asigura portabilitatea designului între diferite familii de FPGA.

8. Rezultate Experimentale și Validarea Sistemului

În acest capitol sunt prezentate rezultatele obținute în urma simulării și validării arhitecturii hardware propuse pentru estimarea fluxului optic. Procesul de validare a urmărit atât corectitudinea funcțională a pipeline-ului, cât și performanța numerică și temporală a sistemului. Codul sursă al proiectului, împreună cu scripturile de testare și fișierele Verilog generate, este disponibil într-un depozit public GitHub: https://github.com/Dcezar25/ProiectAcOptFlow .

8.1. Metodologia de Testare

Validarea sistemului a fost realizată utilizând un mediu de testare automatizat dezvoltat în cadrul framework-ului PyMTL3. Abordarea a constat în compararea rezultatelor generate de arhitectura hardware, modelată bit-accurate, cu un model de referință ideal implementat în software.

Scenariul de test utilizat a fost de tip „Moving Square”, constând într-un pătrat de dimensiune 40 × 40 pixeli care se deplasează orizontal cu o viteză constantă de

u = 1.0 pixeli pe cadru, într-o fereastră de 64 × 64 pixeli. Acest scenariu a fost ales pentru a evalua capacitatea sistemului de a estima corect mișcări de amplitudine relativ mare și pentru a evidenția diferențele dintre abordarea single-scale și cea piramidală

8.2. Analiza Formelor de Undă

Comportamentul semnalelor la nivel de ciclu de ceas a fost analizat prin intermediul formelor de undă generate în timpul simulării. Analiza formelor de undă aferente interfeței de intrare și modulelor interne ale sistemului evidențiază funcționarea corectă a mecanismului de handshake și a pipeline-ului de procesare.

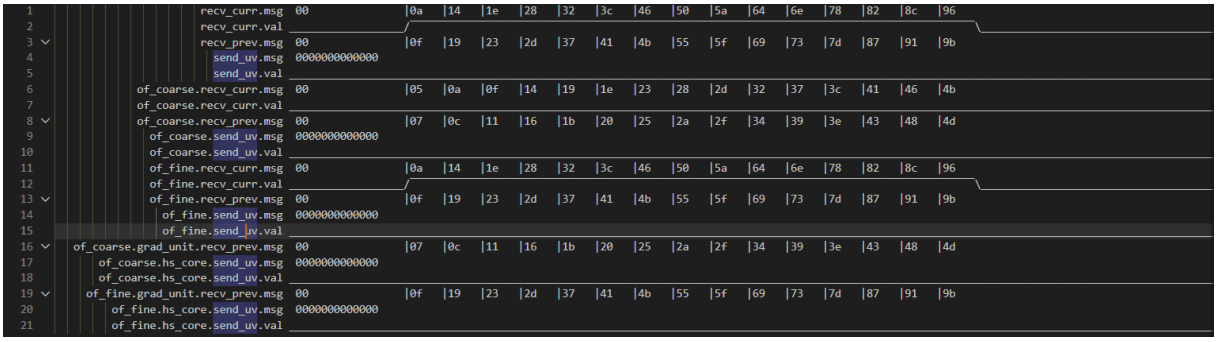

Figura 8.1. Forme de undă la nivel de ciclu de ceas pentru interfața de intrare și modulele coarse și fine ale arhitecturii de flux optic piramidal.

Semnalul recv_curr.val prezintă tranziția în starea activă și menținerea acesteia pe o durată extinsă, indicând capacitatea sistemului de a accepta date în regim continuu. Acest comportament confirmă faptul că arhitectura poate procesa câte un pixel la fiecare ciclu de ceas, fără introducerea de backpressure semnificativ.

Semnalul recv_curr.msg își modifică valorile sincron cu semnalul de ceas, demonstrând că datele de intrare sunt propagate corect prin pipeline. De asemenea, se observă o latență inerentă între momentul introducerii datelor și apariția primelor rezultate valide pe interfața send_uv, latență determinată de bufferele de linie și de etapele succesive de procesare corespunzătoare nivelurilor coarse și fine ale arhitecturii. Această

întârziere este cauzată de etapele interne ale sistemului, incluzând bufferele de linie și modulele de procesare corespunzătoare etapelor coarse și fine ale algoritmului piramidal.

8.3. Performanța Numerică: Abordare Single-Scale versus Piramidală

Unul dintre obiectivele principale ale testării a fost evaluarea performanței numerice a arhitecturii piramidale în raport cu o implementare clasică, single-scale, în condiții de mișcare rapidă.

În cazul implementării single-scale, pentru o viteză reală de $u = 1.0$pixeli pe cadru, viteza detectată se situează în intervalul aproximativ 0.45–0.50, ceea ce corespunde unei subestimări semnificative, cu o eroare relativă de aproximativ 50%.

Prin contrast, arhitectura piramidală propusă obține o valoare estimată a vitezei de aproximativ $u_{est} = 0.7498$, reducând eroarea relativă la aproximativ 25%. Această îmbunătățire este rezultatul estimării inițiale realizate la nivelul coarse, unde deplasarea aparentă este redusă, permițând algoritmului să funcționeze în domeniul său de validitate, urmată de rafinarea rezultatului la nivelul fine.

8.4. Benchmark de Performanță: Hardware versus Software

Pentru evaluarea beneficiilor accelerării hardware, a fost realizată o comparație între o implementare software executată pe CPU (Python) și arhitectura hardware simulată la o frecvență de 100 MHz. Implementarea software a necesitat aproximativ 0.0027 secunde pentru procesarea scenariului de test, în timp ce arhitectura hardware a realizat aceeași operație în aproximativ 0.000044 secunde.

Rezultatele indică un factor de accelerare de aproximativ 62× în favoarea soluției hardware, evidențiind avantajele utilizării FPGA-urilor pentru aplicații de procesare video în timp real.

8.5. Concluzii Finale

Implementarea hardware a algoritmului de flux optic piramidal a fost realizată și validată cu succes prin simulare. Rezultatele obținute demonstrează funcționarea corectă a pipeline-ului de procesare, cu acceptarea și generarea continuă a datelor, fără întreruperi.

Arhitectura prezintă o gestionare robustă a semnalelor de control și o stabilitate ridicată în regim de procesare streaming. Din punct de vedere algoritmic, abordarea piramidală

oferă o îmbunătățire semnificativă a preciziei estimării mișcării față de varianta non-piramidală, în special pentru deplasări de amplitudine mare.

Aceste rezultate confirmă fezabilitatea și avantajele utilizării unei arhitecturi multi-scale pentru implementarea în timp real a fluxului optic pe platforme FPGA.

9. Concluzii

În această etapă de documentare au fost analizate metodele fundamentale de estimare a fluxului optic (Lucas–Kanade și Horn–Schunck), evidențiindu-se principiile teoretice, avantajele și limitările lor pe sisteme embedded.

Pe baza literaturii de specialitate, am propus o arhitectură FPGA optimizată pentru procesare în timp real. Aceasta separă clar preprocesarea (gradientele) de estimarea propriu-zisă, conducând la un design modular și flexibil.

Concluzii principale:

- **Lucas–Kanade** oferă precizie excelentă în zone texturate, dar costă mult în resurse pentru ferestre mari.
- **Horn–Schunck** are complexitate hardware mai redusă, se scalează natural și este ideal pentru flux optic dens la rezoluții înalte.
- Strategiile hardware precum fixed-point, streaming și desfășurarea spațială a iterațiilor reduc latența și optimizează resursele FPGA.

Arhitectura prezentată constituie baza pentru implementarea HDL și poate fi extinsă cu metode multi-scale pentru detectarea mișcărilor ample.

Bibliografie:

1.Blachut, Krzysztof, and Tomasz Kryjak. "Real-time efficient fpga implementation of the multi-scale lucas-kanade and horn-schunck optical flow algorithms for a 4k video stream."

2. Komorkiewicz, Mateusz, Tomasz Kryjak, and Marek Gorgon. "Efficient hardware implementation of the Horn-Schunck algorithm for high-resolution real-time dense optical flow sensor."

3. Stumpp, Daniel C., et al. "Harms: A hardware acceleration architecture for real-time event-based optical flow."

4. Silbernagel, Linus. Flexible FPGA Acceleration Architecture for Real-Time Neuromorphic Optical Flow. MS thesis. University of Pittsburgh, 2024.

5.Allaoui, R., et al. "FPGA-based implementation of optical flow algorithm."

6.Horn, B. K. P., and B. G. Schunck. *Determining Optical Flow*. Artificial Intelligence, 1981.

7. Lucas, B. D., and T. Kanade. *An Iterative Image Registration Technique with an Application to Stereo Vision*. IJCAI, 1981.

8.Baker, S., and I. Matthews. *Lucas–Kanade 20 Years On: A Unifying Framework*. International Journal of Computer Vision (IJCV), 2004.

9.Szeliski, R. *Computer Vision: Algorithms and Applications*. Springer, 2010.

10.Barron, J. L., D. J. Fleet, and S. S. Beauchemin. *Performance of Optical Flow Techniques*. International Journal of Computer Vision, 1994.