

# 5. Corespondențe stereo și reconstrucția 3D a unei scenei

---

Detectorul de colțuri Harris  
Reconstrucția 3D prin triangulație

---

În acest laborator va fi prezentată o metodă de detecție a corespondențelor în imagini stereo precum și modul prin care aceasta poate fi utilizată pentru a reconstrui poziția 3D a punctelor vizualizate printr-o cameră stereo.

## 5.1 Baze teoretice

### 5.1.1 Detectorul de colțuri Harris

Cea mai comună definiție a unui colț într-o imagine a fost dată de Harris [?] în 1988 și se bazează pe matricea derivatelor de ordinul doi ( $\partial^2 x, \partial^2 y, \partial x \partial y$ ) a intensității imaginii  $I$ . Derivatele de ordinul doi ale unei imagini, la fiecare poziție  $(x, y)$ , pot fi considerate ca formând o nouă ”*imagine de derivate de ordinul doi*”, sau o nouă *imagine Hessian*  $H(x, y)$ . Terminologia prezentată este dată de matricea Hessian în jurul unui punct care, pentru cazul bidimensional, poate fi definită după cum urmează:

$$H(x, y) = \begin{bmatrix} \frac{\partial^2 I(x, y)}{\partial x^2} & \frac{\partial^2 I(x, y)}{\partial x \partial y} \\ \frac{\partial^2 I(x, y)}{\partial y \partial x} & \frac{\partial^2 I(x, y)}{\partial y^2} \end{bmatrix} \quad (5.1)$$

Colțurile astfel detectate pot reprezenta principalele caracteristici care permit determinarea corespondențelor între două imagini stereo. Un exemplu de astfel de corespondențe poate fi observat în Fig. 5.1. Colțurile identificate cu detectorul Harris sunt considerate drept caracteristici solide ce pot reprezenta corespondențe în imagini stereo.

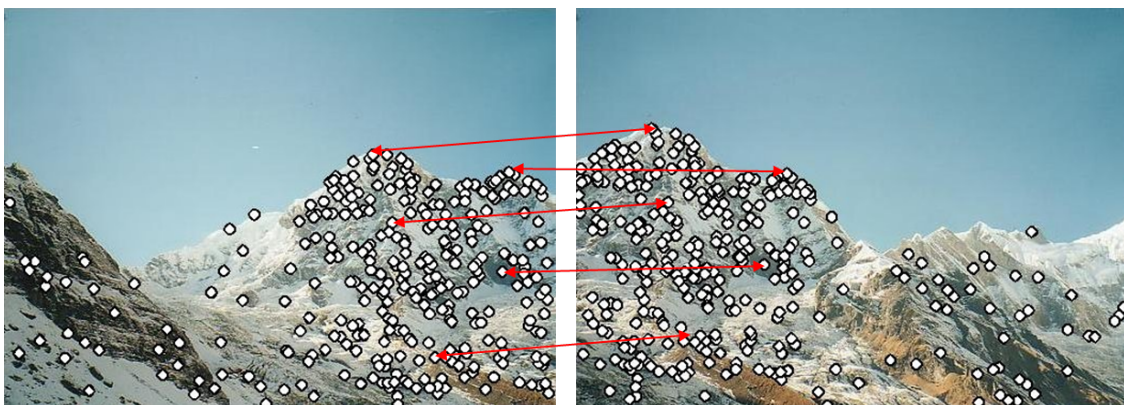


Fig. 5.1 Puncte corespondente între două imagini.

Detectorul Harris se bazează pe analiza funcției de autocorelație, ce estimează schimbarea locală a intensității pentru o poziție  $(x, y)$ , sau derivata intensității imaginii la poziția  $(x, y)$ :

$$E(\Delta x, \Delta y) = \sum_{x,y} w(x, y) \cdot [I(x + \Delta x, y + \Delta y) - I(x, y)]^2, \quad (5.2)$$

unde  $w$  reprezintă o mască de netezire de formă circulară. De obicei,  $w$  este o mască Gaussiană de forma:

$$w(x, y) = \frac{1}{2\pi\sigma} \exp^{-\frac{x^2+y^2}{2\sigma^2}} \quad (5.3)$$

unde  $\sigma$  reprezintă deviația standard sau scara de netezire.

În implementarea detectorului Harris se pot distinge două etape:

- prima, în care sunt calculate valorile funcțiilor de autocorelație pentru fiecare pixel din imagine;
- a doua, unde este determinat un maxim local al funcției de autocorelație într-o vecinătate definită de utilizator. Pixelii asociați acestor valori de maxim local sunt considerați ca fiind colțuri.

Variația intensității pixelilor din imagine poate fi calculată utilizând o mască ale cărei dimensiuni sunt alese de către utilizator. Variația intensității pixelilor survine în urma deplasării acestei măști pe distanțe mici de-a lungul diferitelor direcții. Considerând o astfel de abordare, pot exista trei cazuri distincte:

- dacă masca conține pixeli al căror nivel este apropiat de nivelul intensității imaginii, în urma deplasării măștii în orice direcție va rezulta o variație mică a acesteia;
- dacă masca conține porțiuni dintr-o imagine, o deplasare a măștii de-a lungul marginii va conduce la variații mici ale intensității. În schimb, o deplasare perpendiculară pe margine va conduce la obținerea unei variații mari a intensității pixelilor;
- dacă în interiorul măștii se află un colț atunci deplasarea măștii în orice direcție va conduce la obținerea unei variații semnificative a intensității. În acest caz, un colț poate fi determinat găsind valoarea minimă a variației intensității.

Variația intensității pixelilor (gradientii imaginii) pot fi reprezentați astfel:

$$I_x = I \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} = \frac{\partial I(x, y)}{\partial x}, \quad (5.4)$$

$$I_y = I \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T = \frac{\partial I(x, y)}{\partial y}. \quad (5.5)$$

Pentru variații mici, rezultă:

$$E(\Delta x, \Delta y) = \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}. \quad (5.6)$$

### 5.1.2 Reconstrucția 3D

Principiile de bază ale reconstrucției stereo sunt ilustrate grafic în Fig. 5.2. Punctele  $p_L$  și  $p_R$  reprezintă proiecția punctului 3D ( $P$ ) în planurile imaginilor celor două camere (stângă și dreaptă), fiecare senzor având propriul punct de vizualizare. Reconstrucția stereo a punctului tridimensional  $X$  este dată de proiecția razelor, ilustrate prin linie punctată în Fig. 5.2, ce pleacă din centrele optice  $c_L$  și  $c_R$  ale camerelor și trec prin punctele  $x_L$  și  $x_R$ .

*Matricea de proiecție* descrie configurația unei camere în lumea reală. Aceasta este rezul-

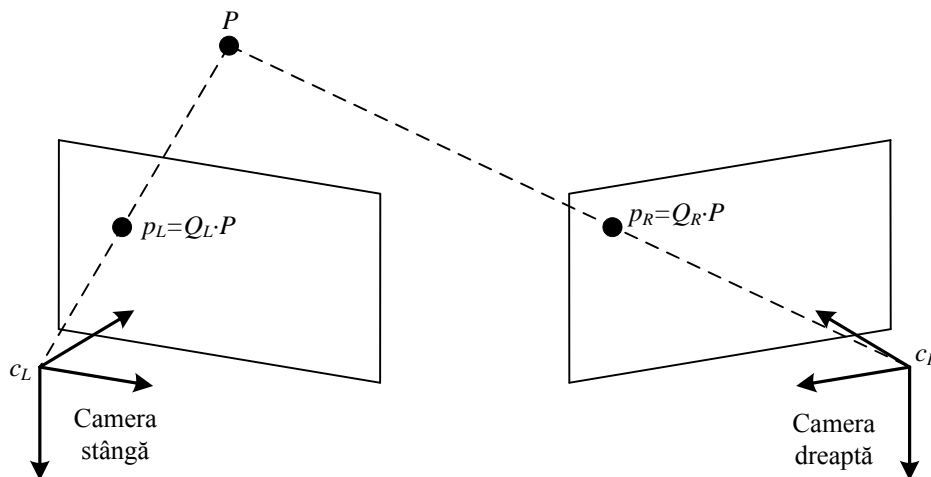


Fig. 5.2 Reconstrucția unui punct 3D ( $P$ ) din punctele vizualizate 2D  $p_L$  și  $p_R$ , achiziționate de la o cameră stereo cu matricele de proiecție  $Q_L$  și  $Q_R$  cunoscute.

tatul unui proces de *calibrare a camerei*. În cazul unei camere stereo sunt folosite două matrice de proiecție, una pentru senzorul stâng  $Q_L$  și una pentru cel drept  $Q_R$ . În funcție de semnificația lor, parametrii matricei de proiecție pot fi clasificați după cum urmează:

- *parametri intrinseci* ( $C_{int}$ ), ce descriu caracteristicile interne ale camerei, precum distanța focală, intersecția axei optice cu planul imaginii și aspectul pixelilor, etc.;
- *parametri extrinseci* ( $C_{ext}$ ), ce reprezintă o transformare omogenă între poziția și orientarea camerei în raport cu un sistem de coordonate de referință.

Atunci când ambii parametri, intrinseci și extrinseci, sunt cunoscuți, matricea de proiecție a unei camere poate fi determinată cu următoarea relație:

$$Q = C_{int} \cdot C_{ext}. \quad (5.7)$$

Acuratețea cu care se poate realiza reconstrucția unei scene 3D depinde de cantitatea de informație a priori cunoscută a parametrilor sistemului video. Astfel, dacă sunt cunoscuți atât parametrii intrinseci cât și cei extrinseci ai sistemului de camere stereo, este posibilă reconstrucția scenei fără ambiguități prin utilizarea *metodei triangulației* [? ? ?].

În aplicația curentă, principiul triangulației este aplicat imaginilor achiziționate de la sistemul stereo. Acest proces presupune utilizarea a două camere video, poziționate rigid la distanța  $b$  una față de cealaltă, ce achiziționează imagini care vizualizează aceeași caracteristică (aceiași punct  $P$ ) în interiorul lor.

Punctul 3D  $P$  din scenă este observat ca un punct 2D  $p_L$ , în planul stâng al imaginii, respectiv  $p_R$ , în planul drept al imaginii (v. Fig. 5.2), formând un sistem, în coordonate omogene, de forma [? ? ?]:

$$\begin{cases} p_L = (x_L, y_L, 1), \\ p_R = (x_R, y_R, 1). \end{cases} \quad (5.8)$$

De asemenea, *axa principală* intersectează planul imaginii în punctul de coordonate  $c_L(x_{cL}, y_{cL})$ , corespunzător imaginii achiziționate de la camera stângă, respectiv în punctul  $c_R(x_{cR}, y_{cR})$  pentru centrul imaginii achiziționate de la camera dreaptă. Se presupune că originea planului de coordonate coincide cu centrul lentilei camerei stângi.

Având la dispoziție informația legată de cele două puncte proiectate în planurile imaginilor, cât și distanța dintre centrele optice ale celor două camere, se pune problema determinării

coordonatelor 3D ale punctului  $P$ . Pentru aceasta se pot utiliza următoarele expresii:

$$X = x_L \cdot \frac{b}{d}, \quad (5.9)$$

$$Y = y_L \cdot \frac{b}{d}, \quad (5.10)$$

$$Z = f \cdot \frac{b}{d}, \quad (5.11)$$

unde  $X, Y, Z$  sunt coordonatele 3D ale punctului  $P$  reprezentat în coordonate omogene, iar  $d$  reprezintă *disparitatea*. În cazul de față, disparitatea reprezintă diferența dintre proiecția punctului  $P$  în imaginea dreaptă și proiecția sa în imaginea stângă, de-a lungul axei  $x$ . Cu cât  $d$  este mai mare, cu atât punctul este mai aproape de cameră. Disparitatea poate fi determină după cum urmează [? ]:

$$d = x_L - x_R. \quad (5.12)$$

Cu ajutorul relațiilor (5.9) ÷ (5.11) se poate trage concluzia că distanța  $Z$  este invers proporțională cu disparitatea, iar coordonatele  $X$  și  $Y$  ale sistemului de referință depind doar de valoarea variabilei  $d$  și de distanța dintre cele două camere.

## 5.2 Cerințe

1. Să se citească de pe HDD o pereche de imagini stereo;
2. Să se detecteze colțurile din imaginea stângă;
3. Să se determine corespondențele dintre imaginea stânga și cea dreaptă;
4. Să se determine coordonatele 3D ale corespondențelor determinate la punctul 3.

## 5.3 Codul sursă al aplicației

```

1 #include <stdlib.h>
2 #include <iostream>
3 #include "opencv2/opencv.hpp"
4 #include "opencv2/highgui/highgui.hpp"
5 #include "opencv2/imgproc/imgproc.hpp"
6
7 using namespace cv;
8
9 int main(int argc, char ** argv)
10 {
11     // Parametrii camerei stereo
12     float fLiniaDeBaza (0.012);
13     float fDistanțaFocala (524.0);
14
15     // Coordonate 3D de reconstruit
16     float X(0.0), Y(0.0), Z(0.0);
17
18     std::string strCaleImagineStanga, strCaleImagineDreapta;
19     std::string strCaleImagineStanga, strCaleImagineDreapta;
20
21     // Incarcarea imaginii de pe disk

```

```
22     if (argc != 3)
23     {
24         std::cout<<"Usage: aplicatie <cale-ImagineStanga> <cale-↵
                imaginereapta>"<<std::endl;
25         exit(0);
26     }
27     else
28     {
29         strCaleImagineStanga = argv[1];
30         strCaleImagineDreapta = argv[2];
31     }
32     // Imagini de intrare
33     Mat ImagineStanga = imread(strCaleImagineStanga.c_str(), ↵
                CV_LOAD_IMAGE_GRAYSCALE);
34     Mat ImagineDreapta = imread(strCaleImagineDreapta.c_str(), ↵
                CV_LOAD_IMAGE_GRAYSCALE);
35
36     // Corespondente stereo
37     std::vector<Point2f> CorespondenteStanga;
38     std::vector<Point2f> CorespondenteDreapta;
39     std::vector<uchar> nStatus;
40
41     int nNoMaximCorespondente (1);
42     std::vector<float> fFeatureErrors;
43
44     // Detectarea colturilor din imaginea stanga
45     goodFeaturesToTrack(ImagineStanga,
46                         CorespondenteStanga,
47                         nNoMaximCorespondente,
48                         0.01,
49                         5.0);
50
51     // Determinarea corespondentelor dintre imaginea stanga si cea ↵
                dreapta
52     calcOpticalFlowPyrLK(ImagineStanga,
53                         ImagineDreapta,
54                         CorespondenteStanga,
55                         CorespondenteDreapta,
56                         nStatus,
57                         fFeatureErrors);
58
59     std::cout << "No corespondente = " << CorespondenteStanga.size() ↵
                << std::endl << std::endl;
60
61     // Reconstructia 3D
62     for (int i = 0; i < CorespondenteStanga.size(); i++)
63     {
64         float fDisparitate = CorespondenteStanga[i].x - ↵
                CorespondenteDreapta[i].x;
65
66         X = CorespondenteStanga[i].x * (fLiniaDeBaza / fDisparitate);
67         Y = CorespondenteStanga[i].y * (fLiniaDeBaza / fDisparitate);
```

```

68     Z = fDistanțaFocala * (fLiniaDeBaza / fDisparitate);
69
70     std::cout << "Reconstructie 3D (X, Y, Z) punct " << i << ": " <<
        << X << ", " << Y << ", " << Z << std::endl;
71 }
72
73     return 0;
74 }

```

#### 5.4 Descrierea funcțiilor principale

```

45 void goodFeaturesToTrack(InputArray image, OutputArray corners, int ←
    maxCorners, double qualityLevel, double minDistance)

```

Calculează colțuri într-o imagine gri.

- `image`: imaginea de intrare;
- `corners`: colțuri detectate;
- `maxCorners`: numărul maxim de colțuri ce poate fi detectat (vor fi selectate cele mai puternice colțuri în cazul în care sunt detectate mai multe);
- `qualityLevel`: calitatea minimă a unui colț;
- `minDistance`: distanța Euclidiană minimă ce poate exista între două colțuri vecine.

```

52 void calcOpticalFlowPyrLK(InputArray prevImg, InputArray nextImg, ←
    InputArray prevPts, InputOutputArray nextPts, OutputArray status, ←
    OutputArray err)

```

Calculează corespondențele dintre două imagini utilizând metoda Lucas-Kanade.

- `prevImg`: imaginea de intrare stanga;
- `nextImg`: imaginea de intrare dreapta;
- `prevPts`: vectorul de puncte pentru care se caută corespondențe;
- `nextPts`: punctele corespondente din imaginea dreapta;
- `status`: vector de status (în cazul în care o corespondența nu a fost găsită, acel index va fi setat 0, altfel va fi 1);
- `err`: vector de eroare (indică gradul de precizie a detecției de corespondențe).