Sisteme de recunoastere a formelor – Laborator 1

RANSAC – potrivirea unei linii la un set de puncte

1. Obiective

Scopul acestei lucrari de laborator este de a familiariza studentii cu mediul Diblook si de a introduce o metoda simpla de recunoastere a unei forme: estimarea unei linii drepte dintr-un set de puncte dintr-un plan.

2. Fundamente teoretice

Random Sample Consensus (RANSAC) este o paradigma pentru potrivirea unui model la date experimentale, introdusa de Martin A. Fischler si Robert C. Bolles in 1981.

Dupa cum au declarat Fischler si Bolles [1] "Procedura RANSAC este opusul tehnicilor conventionale de filtrare: in loc sa folosim cat mai multe date posibile pentru a obtine o solutie initiala, si apoi sa eliminam punctele invalide, RANSAC foloseste un set initial de date cat mai mic posibil si apoi mareste acest set cu date valide atunci cand este posibil."

Algoritmul RANSAC este descris in cele ce urmeaza [2]:

Obiectiv: Potrivirea robusta a unui model la un set de date *S* care contine puncte zgomot.

Algoritm:

1. Selectie aleatoare a unui esantion de puncte *s* din multimea *S* si instantiere a modelului din acest esantion.

2. Determinarea multimii de date S_i care contine puncte situate la o distanta mai mica decat un prag *t* de model. Multimea S_i este multimea de consens a esantionului, si defineste punctele din *S* care satisfac modelul.

3. Daca dimensiunea lui S_i (numarul de puncte care satisfac modelul) este mai mare decat un prag *T*, algoritmul se termina. Optional, se poate re-estima modelul folosind toate punctele din S_i.

4. Daca dimensiunea lui S_i este mai mica decat *T,* se selecteaza un nou subset si se repeta pasii anteriori.

5. Dupa *N* incercari, se selecteaza multimea de consens S_i cu cele mai multe puncte, si (optional) modelul este re-estimat folosind toate punctele din aceasta multime.

2.1. RANSAC pentru linii

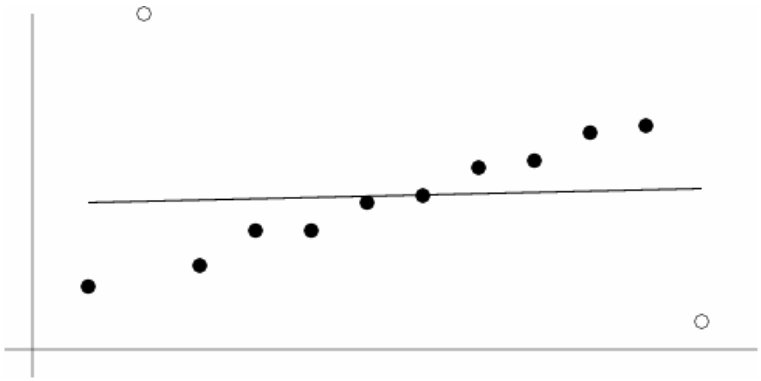

Figura 1-a

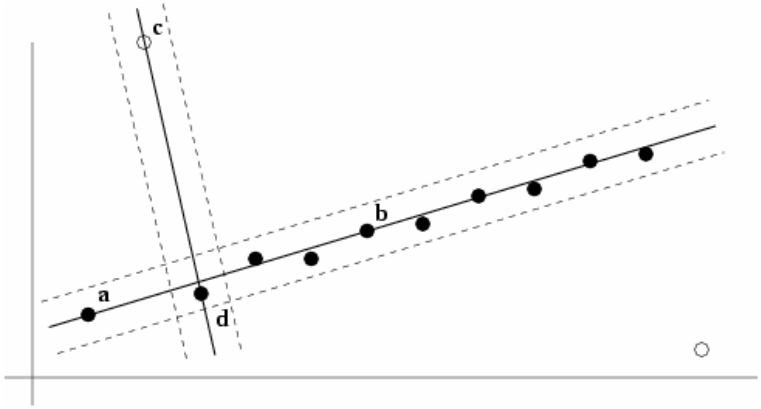

Figura 1-b

Problema, ilustrata in figura 1-a, este urmatoarea: fiind data o multime de puncte 2D, gasiti linia care minimizeaza suma distantelor perpendiculare (regresie ortogonala), in conditiile in care nici unul din punctele valide nu deviaza de la linie cu mai mult de *t* unitati. Exista de fapt doua probleme: potrivirea unei linii la date, si clasificarea datelor in puncte valide si puncte zgomot. Pragul *t* este selectat in functie de zgomotul propriu masuratorii.

Ideea este sa se selecteze doua puncte aleator, iar aceste puncte vor defini o linie. Multimea suport (consens) pentru aceasta linie este data de punctele care sunt la mai putin de o distanta prag de aceasta linie. Selectia aleatoare este repetata de mai multe ori, si linia care are cele mai multe puncte support este considerata potrivirea cea mai robusta. Intuitiv, daca unul din cele doua puncte alese pentru potrivirea liniei este parte a zgomotului, linia estimata nu va avea o multime suport foarte mare.

Mai mult, daca masuram calitatea unei linii prin marimea multimii support, avem avantajul de a favoriza cele mai bune linii de la inceput. Astfel, linia (a,b) din figura 1-b are o marime a suportului de 10, pe cand linia (a, d) are un suport de doar 4. Astfel, desi nici una din linii nu contine zgomot, va fi selectata linia (a,b).

Parametrii algoritmului

1. **Pragul de distanta:** Se alege pragul *t* astfel incat un punct este valid cu o probabilitate *a*. Acest calcul implica cunoasterea distributiei de probabilitate pentru distanta unui punct valid la model. In practica acest prag este ales empiric.

2. **Numarul de esantioane:** Numarul de esantioane *N* este ales pentru a asigura, cu o probabilitate *p*, ca cel putin unul din esantioane nu contine zgomot. De obicei *p* este ales ca 0.99. Daca *w* este probabilitatea ca orice punct selectat este valid, $\varepsilon = 1-w$ este probabilitatea ca el sa fie zgomot. Sunt necesare cel putin N selectii, incat $\left(1 - w^s\right)^N = 1 - p$, deci $N = \log(1 - p)/\log\left(1 - (1 - \varepsilon)^s\right)$.

3. **Marimea pragului pentru multimea consens:** O regula simpla este ca algoritmul sa se finalizeze in momentul in care dimensiunea multimii consens este similara cu numarul de puncte valide presupuse a fi in setul de date, dandu-se proportia dintre puncte valide si zgomot. De exemplu, pentru *n* puncte $T=(1- \varepsilon)n$. Pentru potrivirea liniei din Figura 1, o estimare conservatoare este $\varepsilon = 0.2$, deci $T = (1.0-0.2)\cdot 12 = 10$.

3. Fundamente matematice

Ecuatia unei linii care trece prin doua puncte distincte (x_1, y_1) si (x_2,y_2) este data de:

$$(y_1 - y_2)X + (x_2 - x_1)Y + x_1y_2 - x_2y_1 = 0$$

Distanta unui punct (x_0, y_0) la o dreapta definita de $aX+bY+c = 0$ este:

$$d = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

4. RANSAC pentru linii

Pasii pentru potrivirea unei linii la un set de puncte sunt urmatorii:

1. Initializarea parametrilor algoritmului:
   Pentru imaginea data (*line.bmp*) se vor seta:
   $t = 10$;
   $p = 0.99$;
   $w = 0.3$;
   $s = 2$;
   Avand acesti parametri se poate calcula *N* si *T*.
2. Repetati pentru *N* iteratii:
   a. Selectie aleatoare a doua puncte din setul de date.
   b. Calcularea parametrilor (*a*, *b*, *c*) care definesc o linie care trece prin cele doua puncte.
   c. Numararea punctelor valide pentru aceasta linie.

d. Daca numarul de puncte valide este mai mare sau egal cu *T*, terminare pas 2.
e. Se pastreaza linia cu un numar maxim de puncte valide.
f. (optional) potrivirea liniei la toate punctele valide folosind tehnica celor mai mici patrate (least squares).
g. Desenarea liniei.

5. Desenarea unei linii in DIBLook

```
void CDibView::OnDrawLine()
{
	INCEPUT_PRELUCRARI();

	BYTE * lpDst = (BYTE*)::FindDIBBits((LPSTR)lpD);
	BYTE * lpSrc = (BYTE*)::FindDIBBits((LPSTR)lpS);
	DWORD dwWidth  = ::DIBWidth((LPSTR)lpS);
	DWORD dwHeight = ::DIBHeight((LPSTR)lpS);
	DWORD w=WIDTHBYTES(dwWidth*8);


	CDC dc;
	dc.CreateCompatibleDC(0);
	CBitmap ddBitmap;
	HBITMAP hDDBitmap = CreateDIBitmap(::GetDC(0),
				&((LPBITMAPINFO)lpS)->bmiHeader, CBM_INIT, lpSrc,
				(LPBITMAPINFO)lpS, DIB_RGB_COLORS);

	ddBitmap.Attach(hDDBitmap);

	CBitmap* pTempBmp = dc.SelectObject(&ddBitmap);

	CPen pen(PS_SOLID, 1, RGB(255,0,0));
	CPen *pTempPen = dc.SelectObject(&pen);

	// drawing a line from point (x1,y1) to point (x2,y2)
	int x1=0;
	int y1=0;
	int x2=20;
	int y2=40;
	dc.MoveTo(x1,dwHeight-1-y1);
	dc.LineTo(x2,dwHeight-1-y2);

	dc.SelectObject(pTempPen);
	dc.SelectObject(pTempBmp);

	GetDIBits(dc.m_hDC, ddBitmap, 0, dwHeight, lpDst,
			(LPBITMAPINFO)lpD, DIB_RGB_COLORS);

	SFARSIT_PRELUCRARI("line");
}
```

6. Activitate practica

Folosind mediul Diblook, implementati algoritmul pentru potrivirea unei linii la un set de puncte. Folositi imaginea *line.bmp* pentru a incarca punctele de intrare. Imaginea este de 8 biti/pixel, iar punctele sunt desenate cu culoarea neagra (valoarea pixelului este zero). Desenati obiectul rezultat pe imagine.

7. Bibliografie

[1] Robert C. Bolles, Martin A. Fischler: *A RANSAC-Based Approach to Model Fitting and Its Application to Finding Cylinders in Range Data*, 1981

[2] Richard Hartley, Andrew Zisserman: *Multiple View Geometry in Computer Vision,* 2003