UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Transformarea de perspectiva inversa

Student: Urcan Denisa-Teodora, 30236
Indrumator: Tiberiu Marita
Data: 30.05.2020

Cuprins

- Introducere
  - Descrierea problemei
  - Contextul problemei
  - Motivatie
- Studiu bibliografic
- Metoda propusa
  - Descrierea metodei
  - Detalierea conceptelor
- Rezultate experimentale
- Concluzii
- Bibliografie

1. Introducere

1.1. Descriere problemei

Problema implementata in cadrul acestui proiect are la baza transformarea de perspectiva inversa. Avand o coala alba de hartie observata dintr-o perspectiva aleatoare, imaginea se transforma astfel incat coala sa devina un triunghi, eliminand totodata efectul de perspectiva. Primul pas in realizarea acestui studiu consta in testarea implementarii oferite de biblioteca OpenCv, urmand apoi o implementare proprie precum si sesizarea diferentelor dintre cele doua implementari. Problema descrisa anterior si concomitent rezolvarea sa consta in determinarea unor corespondente intre imaginea initiala, cu efect de perspectiva, si imaginea dorita, obtinuta conform eliminarii efectului de perspectiva.

1.2. Contextul problemei

Eliminarea efectului de perspectiva este o problema des intalnita in zilele noastre. Conform descrierii, aceasta presupune aducerea in fata a unei coli de hartie (un obiect), avand in vedere tot continutul acestuia (e.g scris). O prima problema pe care abordarea aceasta o rezolva este imprimarea unei poze. Presupunand ca avem o poza pe care dorim sa o printam, filtrarea sa in cadrul acestui program va rezulta o imagine similara cu una scanata. Acesta este doar un prim exemplu de utilizare. Transformarea inversa a perspectivei poate deveni baza rezolvarii unor multitudini de probleme contemporane, de la detectia automata a curbei pana la aplicatii care au ca si componente transformari video.

1.3. Motivatie

Constituind o problema contemporana, eliminarea efectului de perspectiva sau transformarea inversa poate deveni o baza solida a sistemelor inteligente, precum si a unei lucrari de licenta.

2. Studiu bibliografic

Din majoritatea documentelor studiate si mentionate ulterior, reiese principiul care sta la baza alinierii unei imagini, acesta constand in transformarea printr-o matrice numita si Homography(omografie). Omografia e caracterizata de o matrice de 3x3, dupa cum urmeaza:

$$H = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix}$$

Matricea se calculeaza corespunzator punctelor alese, dupa cum urmeaza: considerand ca si obiect o coala de hartie, care are 4 colturi, se vor alege ca si puncte corespondente cele 4 colturi ale obiectului (hartiei) precum si cele 4 colturi ale imaginii, rezultatul constand in eliminarea perspectivii pe intreaga imagine. Imaginea rezultata va consta in maparea intregii coli de hartie pe dimensiunea imaginii. Algoritmul de calculare a matricii de omografie presupune concomitent calcularea a 4 corespondente, corespunzatoare celor 8 puncte: (p1, p1'), (p2, p2'), (p3, p3'), (p4, p4'), unde pi, i = 0,..4 vor fi cele 4 colturi ale obiectului, iar pi', i = 0,...4 cele 4 colturi ale imaginii, crespunzatoare obiectului in imaginea finala. Cele 4 corespondente se mapeaza fiecare pe o matrice de 2x9, dupa cum urmeaza:

$$p_i = \begin{bmatrix} -x_i & -y_i & -1 & 0 & 0 & 0 & x_i x'_i & y_i x'_i & x'_i \\ 0 & 0 & 0 & -x_i & -y_i & -1 & x_i y'_i & y_i y'_i & y'_i \end{bmatrix}$$

Cele 4 corespondente se vor retine intr-o matrice, dupa cum urmeaza:

$$P = \begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1 x'_1 & y_1 x'_1 & x'_1 \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1 y'_1 & y_1 y'_1 & y'_1 \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2 x'_2 & y_2 x'_2 & x'_2 \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2 y'_2 & y_2 y'_2 & y'_2 \\ -x_3 & -y_3 & -1 & 0 & 0 & 0 & x_3 x'_3 & y_3 x'_3 & x'_3 \\ 0 & 0 & 0 & -x_3 & -y_3 & -1 & x_3 y'_3 & y_3 y'_3 & y'_3 \\ -x_4 & -y_4 & -1 & 0 & 0 & 0 & x_4 x'_4 & y_4 x'_4 & x'_4 \\ 0 & 0 & 0 & -x_4 & -y_4 & -1 & x_4 y'_4 & y_4 y'_4 & y'_4 \end{bmatrix}$$

Adaugand o constrangere pentru a evita solutiile in care toate elementele matricii H sunt 0, se va adauga constrangerea ca determinantul acesteia sa fie 1, |H| = 1. Matricea H va rezulta din relatia:

$$PH = 0$$

Operatia descrisa ulterior corespunde functiei de findHomography, definita astfel:

int cvFindHomography(const CvMat* src_points, const CvMat* dst_points, CvMat* homography, int method=0, double ransacReprojThreshold=3, CvMat* mask=0)

Dupa calcularea matricii omografe, aceasta trebuie aplicata pentru fiecare pixel din imagine, operatie care corespunde functiei de warpPerspective din biblioteca Opencv, definita dupa cum urmeaza:

void cvWarpPerspective(const CvArr* src, CvArr* dst, const CvMat* map_matrix, int flags=CV_INTER_LINEAR+CV_WARP_FILL_OUTLIERS, CvScalar fillval=cvScalarAll(0))

Detaliile legate de implementarea proprie se vor descrie in sectiunile care urmeaza.

3. Metoda propusa

3.1 Descrierea metodei

Metoda descrisa in cadrul paragrafului anterior este si cea aleasa pentru implementarea in C. Aditional, s-au implementat diverse metode ajutatoare, precum cea pentru preluarea colturilor intr-o anumita ordine. Acest algoritm, implementat pe baza functiei din biblioteca OpenCv goodFeaturesToTrack, descrisa astfel:

void cvGoodFeaturesToTrack(const CvArr* image, CvArr* eig_image, CvArr* temp_image, CvPoint2D32f* corners, int* corner_count, double quality_level, double min_distance, const CvArr* mask=NULL, int block_size=3, int use_harris=0, double k=0.04)

Aceasta metoda are la baza un algoritm scanline, presupunand preluarea colturilor prin trasarea unor linii orizontale. Deoarece ordinea colturilor preluata in baza acestui algoritm se diferentiaza dupa pozitia si respectiv perspectiva obiectului (a colii de hartie) in imagine, dupa preluarea colturilor acestea se vor sorta in 2 liste (top si bottom) conform pozitiilor coordonatelor sale fata de centrul de greutate al obiectului. Astfel se asigura ordinea clockwise pentru oricare dintre imaginile testate.
Preluarea colturilor e precedata de gasirea matricii de omografie, conform descrierii din paragraful anterior. De asemenea, pentru calculul matricii de omografie s-a folosit libraria svd, care ofera direct solutia sistemului descris, selectand pentru matricea H ultima coloana (ultimul vector singular din V). Deoarece acest lucru oferă o omografie DLT (transformare liniara directa)se minimizeaza eroarea algebrica. Aceasta eroare nu este semnificativa din punct de vedere geometric si, prin urmare, este posibil ca omografia calculata sa nu fie la fel de buna precum cea oferite de biblioteca OpenCv. Dupa aflarea matricii de omografie, este usor sa aplicam perspectiva asupra fiecarui pixel din imagine, dupa cum urmeaza:

$$\begin{bmatrix} x'/\lambda \\ y'/\lambda \\ \lambda \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

3.2. Detalierea conceptelor

In aceasta sectiune se vor explica in detaliu conceptele prezentate in paragrafele anterioare, prezentand in final un pseodocod pentru conceptele alese.
Prima metoda prezenta in cod, si anume cea de gasire si sortare a colturilor, se realizeaza folosind functia mentionata goodFeaturesToTrack. Pentru a usura munca, pozele testate vor fi grayscale, avand un fundal inchis (de regula, negru). Imaginea originala este binarizata folosind un prag automat pe baza aflarii intensitatii minime si maxime, dupa algoritmul prezentat in laborator. Deoarece fundalul nu este in totalitate negru, in urma algoritmului de binarizare automata au fost prezentate cateva zgomote de fundal. Pentru ca sa nu fie afectata gasirea colturilor, s-a aplicat un filtru median care a dus la eliminarea zgomotelor de fundal de tip sare si piper, algoritm de asemenea implementat in laborator. Urmatoarea problema care se pune la gasirea colturilor este ca avand scris in interiorul colii de hartie, functia din biblioteca OpenCv goodFeaturesToTrack va gasi colturile scrisului si nu colturile obiectului in anumite circumstante. Pentru eliminarea acestor cazuri defavorabile imaginea se prelucreaza cu operatii morfologice, si anume un numar egal de dilatari si eroziuni pentru a nu modifica aria obiectului. Astfel, scrisul se elimina si colturile vor fi prelucrate pe o imagine care nu contine scris. Dupa cum este mentionat anterior, colturile obiectului se vor sorta dupa pozitiile coordonatelor colturilor fata de centrul de greutate. Astfel, se vor crea cele 2 liste, top si bottom. Astfel, punctele care au coordonata y mai mica decat cea a centrului de masa se vor afla in lista de top, altfel in cea de bottom. Fiecare lista in parte va avea in ordine punctele cu coordonata x mai mica pe prima pozitie. Pseudocodul corespunzator descrierii este:

```
list<Point> corners = goodFeaturesToTrack
foreach corner in corners:
        massCenter += corner
massCenter *= 1/corners.size

foreach corner in corners:
        if corner.y  < massCenter.y
                push(top, corner)
        else
                push(bottom, corner)
```

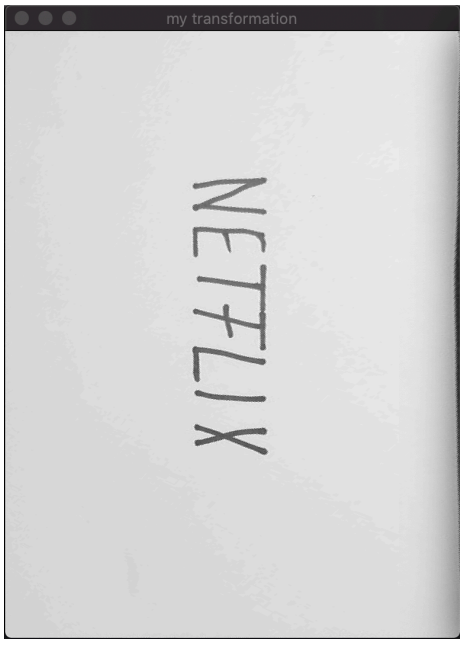
Dupa stabilirea ordinii colturilor, colturile imaginii se vor lua in aceeasi ordine. Urmatoarea functie relevanta descrierii este cea de gasire a matricii de omografie. Conform descrierii, dupa crearea celor 4 vectori pentru corespondente si implicit a matricii P, se foloseste libraria SVD care va oferi direct solutia sistemului. Pentru matricea H, se ia ultimul vector singular corespunzator ultimei coloane din matricea solutie (V). Ca si observatie, aceasta modalitate de prelucrare a sistemului va oferi matricea V transpusa, urmand in urma prelucrariilor sa se aduca la forma initiala (V) si sa se redimensioneze pentru a obtine matricea H de 3x3. Aplicarea perspectivei pe intreaga matrice se realizeaza parcurgand imaginea in totalitate, pixel cu pixel. Orice punct corespondent 2D se va gasi conform urmatorului pseudocod:

```
A = (j, i, 1).
tempMatrix = H * A
dst = x/scale, y/scale
```

Deoarece A e o matrice 1x3 iar H e o matrice de 3x3 (dupa redimensionarea vectorului V), rezultatul in tempMatrix va fi o matrice de 1x3, cu valorile (x, y, scale).

4. Rezultate experimentale

Pentru a sesiza diferentele dintre implementarea proprie si cea oferita de biblioteca OpenCv, o functie aditionala va marca cu o culoare diferita diferentele dintre pixelii celor doua imagini rezultate. Imaginile rezultate sunt destul de similare ca si aspect, se observa diferente in imaginea creata cu metoda proprie de homografie prin pixelarea imaginii (in imaginea corespunzatoare implementarii proprie, se observa ca scrisul este putin pixelat). Diferentele si rezultate apar in pozele de mai jos:



5. Concluzii

Dupa cum se mentioneaza in paragraful anterior, imaginea rezultata in urma implementarii propii apare putin mai pixelata decat imaginea obtinuta prin implementarea din OpenCv. Eroarea algebrica pare destul de nesemnificativa observand imaginile anterioare, dar la colorarea fiecarui pixel diferit intr-o anumita culoare am obtinut rezultatul urmator. De asemenea, pentru o mai buna sesizare a diferentelor am realizat in mod similar o colorare pentru pixelii care sunt egali in cele doua imagini. Cu albastru (turcoaz) se vor observa diferentele, iar cu galben s-au marcat asemanarile.

6. Bibliografie

https://www.learnopencv.com/image-alignment-feature-based-using-opencv-c-python/
https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html
https://docs.opencv.org/3.4/d9/dab/tutorial_homography.html
https://math.stackexchange.com/questions/494238/how-to-compute-homography-matrix-h-from-corresponding-points-2d-2d-planar-homog
http://www.cse.psu.edu/~rtc12/CSE486/lecture16.pdf
https://docs.opencv.org/master/d9/dab/tutorial_homography.html
https://www.learnopencv.com/homography-examples-using-opencv-python-c/
https://answers.opencv.org/question/174548/inverse-perspective-mapping-ipm-on-the-capture-from-camera-feed/
https://gist.github.com/anujonthemove/7b35b7c1e05f01dd11d74d94784c1e58
https://nikolasent.github.io/opencv/2017/05/07/Bird%27s-Eye-View-Transformation.html\
https://en.wikipedia.org/wiki/Homography_(computer_vision)
https://www.researchgate.net/publication/220960373_Inverse_perspective_transformation_for_video_surveillance