Morfarea imaginilor

Marc Rareș-Cristian

Rezumat

Am implementat un sistem de morfare al imaginilor folosind OpenCV și C++. Utilizăm triangularea de tip Delaunay, transformări afine și interpolare liniară pentru a crea tranziții între două imagini.

Cuprins

1 Introducere

Morfarea imaginilor este o tehnică care creează o tranziție între două imagini prin transformarea graduală a geometriei și aspectului imaginii sursă în imaginea destinație.

1.1 Funcționalități principale

- Selecția punctelor din cele 2 imagini
- Triangularea și vizualizarea ei
- Previzualizarea rezultatului
- Exportarea rezultatului în format GIF

2 Arhitectura

2.1 Componente

- **ImageMorpher**: Componenta care implementează morfarea
- **UIController**: Gestionează interfața
- **Main**: Punctul de intrare

3 Teorie matematică

3.1 Interpolarea liniară

Pentru un parametru de morfare $\alpha \in [0, 1]$, pozițiile punctelor intermediare sunt calculate folosind interpolarea liniară:

$$P_{intermediar}(t) = (1 - \alpha) \cdot P_{sursă} + \alpha \cdot P_{destinație} \tag{1}$$

3.2 Triangularea Delaunay

Folosim triangularea Delaunay pentru a crea un mesh care:

- Maximizează unghiul minim în fiecare triunghi
- Evită triunghiurile lungi și subțiri, care pot cauza artefacte

3.3 Transformarea afină

Fiecare triunghi este "warp-uit" folosind o matrice de transformare afină, adică prezervă liniile și paralelismul:

4 Algoritmi

Algorithm 1 Morfarea imaginilor

Intrare: Imaginea sursă I_s, imaginea destinație I_t, parametrul de morfare α
Ieșire: Imaginea morfată I_m
Încarcă și redimensionează imaginile
Calculează punctele intermediare folosind interpolarea liniară
Generează triangularea Delaunay
Inițializează imaginile de ieșire W_s și W_t ca matrice de zero
for triunghi T_i în triangulare **do**
    Ia vârfurile triunghiului sursă T_s^i
    Ia vârfurile triunghiului țintă T_t^i
    Ia vârfurile triunghiului intermediar T_m^i
    Fa warping $I_s(T_s^i) \to W_s(T_m^i)$ folosind transformarea afină
    Fa warping $I_t(T_t^i) \to W_t(T_m^i)$ folosind transformarea afină
end for
$I_m = (1 - \alpha) \cdot W_s + \alpha \cdot W_t$
returnează I_m

Algorithm 2 Warping la triunghi

Intrare: Imaginea sursă I, vârfurile triunghiului V_{src}, V_{dst}
Ieșire: Triunghiul după warp în imaginea destinație
Calculează bounding boxes pentru triunghiurile sursă și destinație
Calculează matricea de transformare afină M de la V_{src} la V_{dst}
Aplică warping afin în regiunea triunghiului sursă
Copiază pixelii după warp în destinație folosind masca (masca de la bounding box spune care pixeli fac parte din triunghi)

5 Implementare

5.1 Pipeline-ul de procesare

5.1.1 Inițializarea

```
bool ImageMorpher::initialize(const std::string& sourcePath,
                              const std::string& targetPath) {
    sourceImg = cv::imread(sourcePath);
    targetImg = cv::imread(targetPath);

    if (sourceImg.cols > MAX_IMAGE_WIDTH) {
        float ratio = static_cast<float>(MAX_IMAGE_WIDTH) / sourceImg.
            cols;
        cv::resize(sourceImg, sourceImg, cv::Size(), ratio, ratio);
    }

    if (sourceImg.size() != targetImg.size()) {
        cv::resize(targetImg, targetImg, sourceImg.size());
    }

    return true;
}
```

5.1.2 Gestionarea punctelor

Menținem perechi de puncte corespunzătoare între imaginile sursă și destinație:

```
void ImageMorpher::addCorrespondingPoints(const cv::Point2f&
    sourcePoint,
                                          const cv::Point2f& targetPoint
                                              ) {
    sourcePoints.push_back(sourcePoint);
    targetPoints.push_back(targetPoint);
    triangles.clear();
}
```

5.2 Triangularea

```
bool ImageMorpher::calculateTriangulation() {
    if (sourcePoints.size() < 3) return false;

    cv::Rect rect(0, 0, sourceImg.cols, sourceImg.rows);
    std::vector<cv::Point2f> points = sourcePoints;

    points.push_back(cv::Point2f(0, 0));
    points.push_back(cv::Point2f(rect.width - 1, 0));
    points.push_back(cv::Point2f(rect.width - 1, rect.height - 1));
    points.push_back(cv::Point2f(0, rect.height - 1));

    cv::Subdiv2D subdiv(rect);
    for (const auto& point : points) {
        subdiv.insert(point);
    }

    std::vector<cv::Vec6f> triangleList;
    subdiv.getTriangleList(triangleList);

    return true;
}
```

5.3 Keybind-uri

Tastă	Funcție
S	Salvează punctele în fișier
L	Încarcă punctele din fișier
Z	Scoate ultima pereche de puncte
T	Afișează triangularea
C	Șterge toate punctele
A	Preview
G	Generează GIF-ul
X	Șterge fișierele generate local

5.4 Îmbunătățiri Viitoare

- Funcții de interpolare non-liniare (sigmoid)
- Detecția automată a trăsăturilor
- Calcule pe GPU

6 Referințe

1. https://sauravmittal.github.io/computational-photography/image-morphing/
2. https://en.wikipedia.org/wiki/Delaunay_triangulation
3. https://docs.opencv.org/4.x/df/d5b/group__imgproc__subdiv2d.html
4. https://ffmpeg.org/