Instantaneous Deformations of Camera and Video Images.

Prof. dr. em. Jan Paredaens
University of Antwerp, Belgium
www.JanParedaens.be
e-mail: jan.paredaens@uantwerpen.be

Abstract

We show an app, called CaViDe, written in Processing [1,4], that enables the user to deform instantaneously the images of a camera or a video. Images of cameras or videos are normally enclosed in rectangles. CaViDe enables the user to deform this rectangle by adding a corner, deleting a corner or moving a corner. In this way we obtain a polygon. The content of the rectangle, hence the image itself, is also deformed such that it fits into the obtained polygon. Remark that, when the image in the rectangle is changing during playing the video, the deformed image is also changing without any delay.

In https://player.vimeo.com/video/845059906?h=415e7001c2 one can see an application of CaViDe.

1. Introduction

Software for deformation of images of photos is well known. However the deformation of images of cameras and videos is more complex, especially when this deformation is realised without any delay. Images of cameras or videos are normally enclosed in a rectangle. CaViDe, the app that we discuss, enables the user to deform this rectangle. In this way we obtain a polygon. The content of the rectangle, hence the image itself, is also deformed such that it fits into the obtained polygon. Remark that, when the image in the rectangle is changing during playing the video, the deformed image is also changing without any delay.

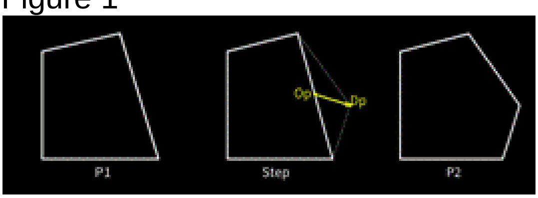
In Section 2 we describe how the rectangle that contains the image can be deformed into a polygon. In Section 3 we give necessary conditions for the deformation of the image contained in the polygon. In Sections 4-6 we handle CaVide. We describe the deformation of a polygon, the deformation of the content of a convex polygon and the deformation of the content of a non-convex polygon [2,3].

In https://player.vimeo.com/video/845059906?h=415e7001c2 one can see an application of CaViDe.

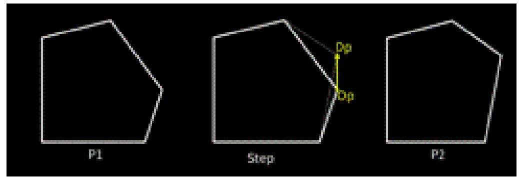
2. The Deformation of a Polygon

First we define a position (i,j) as a point on the screen whose horizontal coordinate is i and whose vertical coordinate is j [2,3]
We start the deformation with a rectangle. This rectangle is deformed by a series of steps, each deforms the polygon P1 into a new polygon P2. We also suppose that the number of corners of a polygon is bigger than three. There are three kinds of steps, S1, S2 and S3, each characterized by two positions on the screen: the original position Op and the destination position Dp. The vector <Op,Dp> defines the deformation direction :

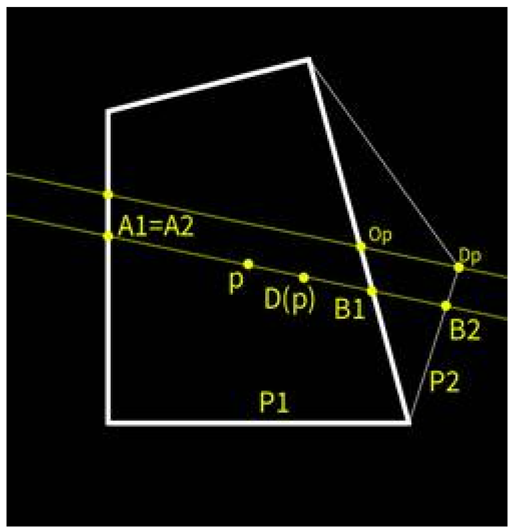
S1. Adding a new corner between two consecutive corners of P1, obtaining P2, Cfr. Figure 1 ;
Figure 1



S2. Changing the position of a corner of P1, obtaining P2, Cfr. Figure 2.
Figure 2



S3. Deleting a corner of P1, obtaining P2. Here the destination position Dp is defined by
|Op,A| / |Op,B| = |Dp,A| / |Dp,B|,
where |p,q| denotes the distance between position p and position q,
Cfr. Figure 3.
Figure 3



3. Conditions for the Deformation of the Content of a Polygon

The set of positions contained in P1 (resp.P2) is called C1 (resp. C2). C1

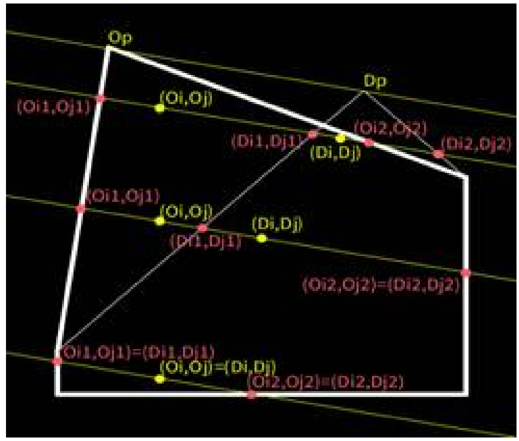
has to be deformed into C2. This deformation is called D. So D is a function from C1 to C2. D must be such that the deformation of the image in P1 into the image in P2 is elegant and natural.
Clearly, D also has to satisfy a number of obvious and natural conditions.

Condition 1 : If P1 = P2 then D has to be the identity;
Condition 2 : D(Op) = Dp; Condition 3 : No information can be lost, ie. for every position p of C1 D(p) has to be a position of C2;
Condition 4 : For every position p2 of C2 there is a position p1 of C1 with D(p1) = p2;
Condition 5 : All the positions of C1 have to be deformed in the same direction, ie. the vectors <p,D(p)> must be in the same direction for every position p of C1;
Condition 6 : If position p is on a side of P1 then D(p) has to be on a corresponding side of P2;
Condition 7 : If p1 and p2 are close to each other then D(p1) and D(p2) have to be close too;
Condition 8 : (Cfr. Figure 4) Consider a line that is parallel to the deformation direction and cuts P1 in positions A1 and B1 and P2 in positions A2 and B2. If position p belongs to the line segment (A1,B1) then D(p) belongs to the line segment (A2,B2) and |A1,p|/|A2,D(p)|=|A1,B1|/|A2,B2|.

For convex polygons Conditions 1-8 can be satisfied as will be proved by CaViDe in Section 5.

For non-convex polygons all the conditions cannot be satisfied together. It will be discussed in Section

Figure 4



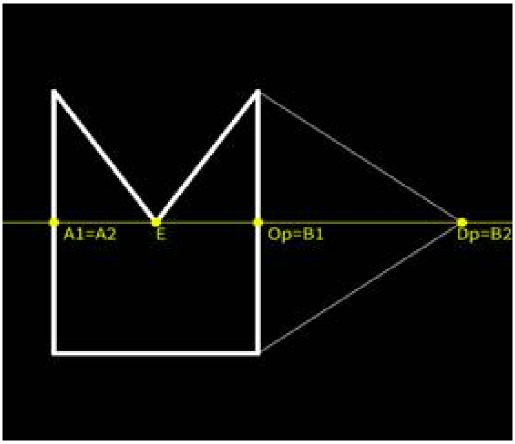
4. CaViDe: The Deformation of a Polygon

Recall Figures 1-3. We now show how the steps S1, S2 and S3 are implemented in CaViDe.

S1. The mouse is set of a side of P1, position Op, resulting in a red dot, the mouse is left pushed and is moved to a new position, position Dp. When the mouse is released, Dp is the position of the new corner obtaining P2.
S2. The mouse is set to a corner of P1, position Op, resulting in a green dot, the mouse is left pushed and it is moved to a new position Dp. When the mouse is released, Dp is the new position of the corner obtaining P2;
S3. The mouse is set to a corner of P1, position Op, resulting in a green dot, the mouse is right pushed. When to mouse is released the corner vanishes, obtaining P2.

5. CaViDe: The Deformation of the Content of a Convex Polygon

Figure 5



In this section we discuss how a deformation step for the content of a convex polygon is implemented in CaViDe. This implementation is mainly based on Condition 8. It is independent of the kind of the step (S1, S2 or S3) but only depends on the polygon P1, the polygon P2 and the deformation direction.

We illustrate this implementation in Figure 5 for step S2. Steps S1 and S3 are analogous, mutatis mutandis

We show 3 different pixel positions (Oi,Oj) of C1. For each of them we have a line parallel to the deformation direction. This line cuts the polygon P1 at position (Oi1,Oj1) and (Oi2,Oj2) and the polygon P2 in (Di1,Dj1) and (Di2,Dj2).

By condition 8 we have

|(Oi1,Oj1),(Oi,Oj)|/|(Oi1,Oj1),(Oi2,Oj2)| = | (Di1,Dj1),(Di,Dj)| / |(Di1,Dj1),(Di2,Dj2)|

where |(Oi1,Oj1),(Oi,Oj)| denotes the distance between (Oi1,Oj1) and (Oi,Oj).

so

Oi = Oi1 + (Oi2-Oi1) * (Di-Di1)/(Di2-Di1) (1)

Oj = Oj1 + (Oj2-Oj1) * (Dj-Dj1)/(Dj2-Dj1) (2)

Let P be the polygon between two steps.

There is a function that gives for every position of P the corresponding position in the rectangle captured by the camera or video.

In order to represent the function above we use a two dimensional array called 'conversie' of positions. Let (i,j) be a position of P. conversie[i][j] gives the corresponding position in the rectangle captured by the camera or video.

Initially conversie[i][j] = (i,j).

During a step conversie is updated as follows:

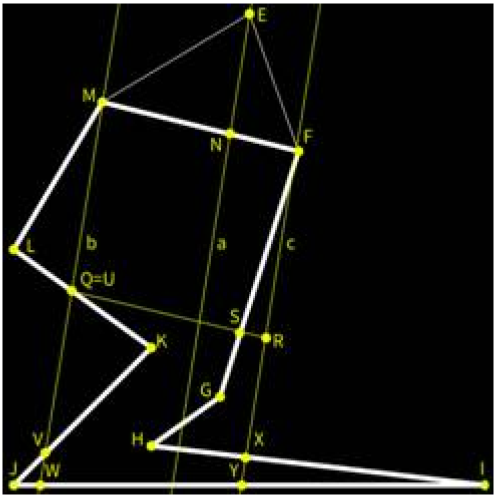
conversie[Di][Dj] := conversie[Oi][Oj] where Di, Dj, Oi, Oj satisfy (1), (2).

So, between two steps, the picture captured can change and the picture shown also changes, but the array 'conversie' does not change. During the step the value of 'conversie' changes.

In the implementation of CaViDe the actual polygon is brown between two steps and the actual content changes accordingly to the picture captured. During a step the actual content is frozen and the polygon is red.

6. CaViDe : The

Deformation of the Content of a Non- convex Polygon
Figure 6



For non-convex polygons the conditions 1-8 cannot in general be satisfied. Indeed in Figure 6 D(E)=E, by Condition 6 and D(E)=B1 by Condition 8, which is a contradiction.

We propose to weaken Condition 8 in this way: Condition 8a: There exists two convex polygons P3 and P4 (with content C3 and C4 respectively) such that

-. C3 is a subset of C1 and C4 is a subset of C2;
-. D|C3 obeys Conditions 1-8;
-. D|(C1-C3) is the identity. Clearly the deformation of the content of convex polygons, that we discussed in Section 5, also satisfies Conditions 1-7,8a.

We now give an implementation that satisfies Conditions 1-7,8a for non- convex polygons, Cfr Figure 7. We only discuss Step S1. Steps S2 and S3 are analogous.

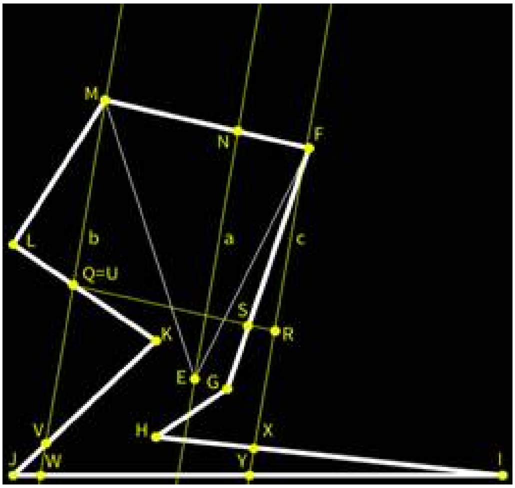
Consider the non-convex Polygon P1 = (F,G,H,I,J,K,L,M) that is deformed to polygon P2 = (F,G,H,I,J,K,L,M,E) by a step of kind S1.
We define consecutively:

-. line a that contains Op and Dp (here N and E);
-. line b that contains M and is parallel to a;
-. line c that contains F and is parallel to a;
-. T is the set of all corner points of P1 and all intersections of P1 with b or c, that are between b and c or on b or c.

Here T = {G,H,K,X,Y,Q,V,W}.
-. Q the closest point of T to line segment (M,F), with the

Figure 7 exception of the points of the line segment (M,F);
-. line segment (Q,R) that is parallel to side (M,F);
-. if line segments (F,G) and (Q,R) have an intersection then S is that intersection, else S=R;
-. if line segments (M,L) and (Q,R) have an intersection then U is that intersection, else U=Q;
-. P3 = (F,S,U,M) and P4 = (F,S,U,M,E);

This construction is almost always possible and satisfies Conditions 1-7,8a.
But, in one case, there is a problem, Cfr. Figure 8.
Figure 8



Here again we consider the non-convex Polygon P1 = (F, G , H , I , J , K , L , M) that is deformed to polygon P2 = (F,G,H,I,J,K,L,M,E) by a step of kind S1. But the result P4 = (F,S,U,M,E) is not a polygon anymore. Such a deformation is excluded.

This problem could be avoided by replacing the line segment (Q,R) by a more complex series of line segments between Q and R.

6. References

[1]Processing, https://processing.org.
[2] M. Dunajski, Geometry, A Very Short Introduction, 2022. [3] R. Gelca, I. Onisor, C. Shine, GeometricTransformations, 2022.
[4] C. Reas, B. Fry, Processing, A Programming Handbook for Visual Designers and Artists, 2014