B Spline curve evolution and a Tangential Evolution Term: Implementation Notes

V. Srikrishnan

February 7, 2010

1 Introduction

This library is based on the work[**?**]. Briefly, this library provides an implementation of curve evolution equation using B-Splines. This software collection makes heavy use of the OpenCV[**?**] library. Before describing the library itself, a brief overview of parametric active contours is provided. This will provide the background for understanding the code as well as introducing the various symbols we have used in the code. The referenes are kept to a minimum because this is not a paper. This does not mean that the we do not acknowledge the earlier research works! For any queries or bug-reports, please email me at v.srikrishnan@gmail.com .

2 Curve Evolution Equations and Active Contours

Active contours are a well-known formulation for segmentation and tracking tasks. Contours are planar curves; an energy functional is defined on them. The general form of the energy functional is as

$$E[C] = E_{ext}[C] + E_{int}[C] \tag{1}$$

where $C(p) = [x(p), y(p)]$ is the curve parameterised by p. The Euler-Lagrange equations for the function leads to PDEs which are solved numerically to obtain the solution which hopefully will segment the object of interest. The curve evolution can be written as

$$C_t(p,t) = \alpha(p,t)\mathbf{T}(p,t) + \beta(p,t)\mathbf{N}(p,t) \tag{2}$$

Here **N** and **T** stand for the local normal and tangent vectors respectively. This is illustrated in figure **??**. The important point to note is that

- normal force component $\beta(p,t)$ has a role in changing the shape of the curve.
- tangent term $\alpha(p,t)$ re-parameterises the curve.

For parametric implementation of evolution equations, the curve points bunch and space out along the curve. This causes poor segmentation and problem of loop formation in case of B-Spline implementation. For further details, see [**?**]. In this work, the authors propose a tangential term which re-parameterises the curve at every iteration to remove the above mentioned problem. *This does not happen with level set implementation of curve evolution equations.*

2.1 BSplines and evolution equation

BSplines have been used extensively for implementation. There are numerous advantages of using B-Splines for implementation. B-Spline curve are defined as:

$$X(p) = \Sigma_{i=1}^{M} B(p,n)C(p) \tag{3}$$

The curve $X(p)$ is generated as the linear combination of the basis functions $B(p,n)$ with the weights being $C(p)$. The weights are also called control points. Notice that if the discretisation is fixed and the basis calculated, then different curves can be generated by different control point sets.

We now have to implement 2 within the B-Spline framework. Therefore, we have to essentially update the control points $C(p)$ at each time step. This is done in a least squares sense. Let the curve be discretised into N points. Therefore, the force moving the curve is known at these N points. Usually, N is much larger than the number of control points M; this is not a restrictive assumption but is the reason that B-Splines are so popular. Let the change in control points from time t to $t+1$ be denoted by ∂C because of the external force **F** moving the curve by ∂X. The change ∂C can be written following 3 as

$$\partial X = \mathbf{F} = \Sigma_{i=1}^{M} B(p,n)\partial C(p) \tag{4}$$

Now, we have N equations and $M < N$ equations which is an overdetermined system. This is solved using least squares.

3 Basic Datastructures

Description of basic datastructures. From previous section, it is clear that the basis function once calculated can be used to generate different curves sharing the same discretisation and order. The first major datastructure is

```
    struct Cv_Basis
{
    int N_b, N, K;
    CvMat *Deriv1;
    }
```