A Subdivision Framework for Partition of Unity Parametrics

Amirhessam Moltaji
University of Calgary *

Adam Runions
Max Planck Institute for Plant Breeding Research†

Faramarz F. Samavati
University of Calgary ‡

ABSTRACT

Partition of Unity Parametrics (PUPs) are a generalization of NURBS that allow us to use arbitrary basis functions for modeling parametric curves and surfaces. One interesting problem is finding subdivision schemes for this recently developed and flexible class of parametrics. In this paper, we introduce a systematic approach for determining uniform subdivision of PUPs curves and tensor-product surfaces. Our approach formulates PUPs subdivision as a least squares problem, which enables us to find exact subdivision filters for refinable basis functions and optimal approximate schemes for irrefinable ones. To illustrate this approach, we provide sample subdivision schemes with different properties, which are further demonstrated by presenting various examples.

Index Terms: I.3.5 [Computing Methodologies]: Computational Geometry and Object Modeling

1 INTRODUCTION

For almost half a century, parametric curves and surfaces, most notably NURBS, have been an important paradigm in the computational modeling of freeform curves and surfaces. They provide smooth parametrics with local control. Although NURBS offer a number of important benefits for geometric modeling, they impose two significant constraints. First, NURBS can only support restricted control net topologies, which forces modelers to use multiple NURBS patches. Second, NURBS offer limited control over the properties of the parametrics they define. For example, the character of the contribution of each control point to the resulting curve or surface cannot be modified (only the relative contribution of control points). Furthermore, it is not possible to increase smoothness without changing the local support of control points [21]. These limitations stem from the use of B-Splines as the underlying basis functions.

Partition of Unity Parametrics (PUPs) were developed to address the limitations of NURBS commented on above. PUPs generalize NURBS by allowing arbitrary basis functions without enforcing any topological restriction [21]. Here, the weighted B-Spline functions of NURBS are replaced by arbitrary weight functions. This permits modelers to control the characteristics of curves and surfaces by changing the underlying basis functions.

PUPs retain important properties of NURBS such as affine invariance and local support. In addition, the flexibility in choosing basis functions enables PUPs to support a variety of features such as interpolation [21] and C^∞ continuity in tandem with compact support [22]. It is shown in [21] that various curves can be generated by fixing control points and just changing the underlying basis functions. Further applications of PUPs such as font modeling, cursive writing and sketch-based deformation are also explored in [21, 22].

Given the flexibility of PUPs, an interesting and important problem is to determine subdivision schemes for this class of parametrics. This allows us to take advantage of both PUPs and subdivision methods in tandem.

In this paper, we derive and introduce subdivision schemes for PUPs curves and tensor-product surfaces. We reproduce the key classes of curves studied in [21, 22] by means of subdivision schemes. We present a systematic approach for finding such subdivision schemes by formulating refinability as a least squares problem. The least squares solution allows us to identify refinable functions based on its residual. In addition, it produces refinement coefficients, which can be used in stationary subdivision schemes. Moreover, it provides the best possible approximate schemes for irrefinable functions. As an immediate result of our method, we consider polynomials and a class of C^∞ functions with interpolation and derive subdivision schemes for them.

The remainder of the paper is organized as follows. First, we briefly discuss the related works and recent advances in PUPs and subdivision. Second, we introduce PUPs and our notations. Next, we define PUPs subdivision based on the refinement of PUPs weight functions. Then, we propose a method for deriving PUPs subdivision schemes by means of least squares. Afterwards, we introduce the class of functions we have utilized to derive example subdivision schemes. Finally, we conclude the paper with a discussion of our results and key directions for future work.

2 RELATED WORKS

Subdivision has become a common technique for shape modeling. These methods, including B-Spline and NURBS subdivision schemes, have been widely studied (see [4, 11, 17, 28] for a comprehensive review). As PUPs are a superset of NURBS, a group of our related works consists of algorithms that extend common NURBS subdivision schemes.

The Lane-Riesenfeld algorithm, as the most studied subdivision algorithm, encapsulates B-Spline subdivision into a refinement and smoothing phase. The method is limited, however, as it can only model uniform subdivision of B-Splines. Attempting to address this issue, Cashman et al. [5, 7, 8] have proposed new algorithms that support non-uniform refinement for B-Splines of arbitrary degree. Furthermore, they extend their algorithm to meshes with extra-ordinary points in [6]. In [9], Cashman et al. extend the Lane-Riesenfeld algorithm by utilizing the repeated application of local smoothing operators to create successively smoother curves. These algorithms improve NURBS subdivision methods, but are still restricted to weighted B-Spline basis functions. In another attempt to extending B-Spline subdivision schemes Schaefer et al. [26] replace arithmetic mean of these subdivision schemes with non-linear average functions (e.g. the geometric mean). They succeed in deriving subdivision schemes for Gaussians, spiral and circular arcs. However, because they are using non-linear average functions, the resulting schemes are not affine-invariant.

Least Squares is the main tool that we employ to derive subdivision schemes. This technique has been previously used for reverse subdivision [2, 23–25]. We also use the idea of refining basis functions. FThis idea was first proposed by Micchelli and Prautzsch, who used refinement of basis functions for the systematic study of stationary subdivision schemes [18]. Although pioneering, their analysis is limited to non-negative refinement coefficients that sum to one (row-wise) in a subdivision matrix. PUPs refinement coefficients are free of such restrictions.

*e-mail: amoltaji@ucalgary.ca
†e-mail: runions@mpipz.mpg.de
‡e-mail: samavati@ucalgary.ca

Finally, our main framework is based on PUPs, a generalization of NURBS [21]. The PUPs framework extends NURBS by supporting arbitrary basis functions. The authors have demonstrated applicability of PUPs in several aspects of geometric modeling by exploring different types of basis functions. Recently, they developed a new basis function that provides interpolation and C^∞ continuity with compact support [22]. In addition, PUPs have been used for texture synthesis [3] and to derive rendering kernels with arbitrary accuracy order [1].

3 Background

In this section, we introduce our notation and provide the definition of PUPs curves.

Given a set of ordered control points $\{P_1,\dots,P_n\}$ with their corresponding weight functions $\{w_1(u),\dots,w_n(u)\}$, a PUPs curve $Q(u)$ is defined as

$$Q(u)=\sum_{i=1}^{n}\frac{w_i(u)}{\sum_{j=1}^{n}w_j(u)}P_i \quad \text{for} \quad u_l<u<u_r\,, \tag{1}$$

where u_l and u_r denote the domain bounds. In this definition, each weight function $w_i(u)$ is normalized through division by $\sum_j w_j(u)$ to ensure affine-invariance. We assume $\sum_j w_j(u)\neq 0$ for all u (to guard against indeterminant forms).

Although any set of weight functions can be used, often PUPs are constructed from shifted versions of a given function [1, 21, 22]. Particularly, given a weight function $w(u)$ and a scalar value d as shift, $w_i(u)$ is defined as $w(u-id)$ and the corresponding *uniform* PUPs curve is

$$Q(u)=\sum_{i=1}^{n}\frac{w(u-id)}{\sum_{j=1}^{n}w(u-jd)}P_i\,. \tag{2}$$

Choosing a uniform B-Spline function (over a set of consecutive integer knots) as $w(u)$ and setting shift to be one ($d=1$), Eq. 2 defines a uniform B-Spline curve (note that the denominator reduces to 1 because of B-Spline's sum-to-one property). Additionally, by choosing weighted B-Spline basis functions Eq. 1 defines a NURBS curve. Hence, PUPs are consistent with NURBS and B-Splines, and the PUPs framework thus extends them.

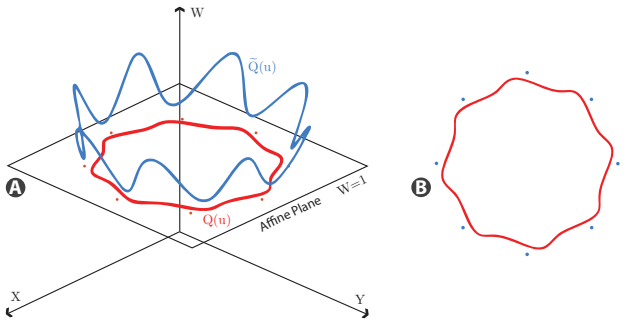

Figure 1: (A) a curve in the Grassmann space before normalization (in blue). After normalization the blue curve is projected to affine space and the red PUPs curve is produced. (B) The resulting 2D PUPs curve.

It is also important to note that any PUPs curve can be written as a rational curve. Let $\mathbf{p}$ be the vector of control points, $\mathbf{1}$ be the vector of ones, and $\mathbf{W}(u)$ be the vector of weight functions. We define $Q(u)$ as

$$Q(u)=\frac{\tilde{Q}(u)}{G(u)}\,, \tag{3}$$

where

$$\begin{aligned}\big[\tilde{Q}(u),G(u)\big]&=\begin{bmatrix}w_1(u) & \cdots & w_n(u)\end{bmatrix}\begin{bmatrix}P_1 & 1\\ \vdots & \vdots\\ P_n & 1\end{bmatrix}\\ &=\mathbf{W}(u)\ [\mathbf{p},\mathbf{1}]\ .\end{aligned} \tag{4}$$

In this definition, $Q(u)$ is decomposed to $\tilde{Q}(u)$ (a curve in the Grassmann space) which division by $G(u)$ projects it to the affine plane [14] (see Fig. 1). This decomposition allows us to work on the curve independent from normalization. We will later employ this definition of PUPs to derive its subdivision in the next section.

4 PUPs Subdivision

Our goal is to build subdivision schemes for uniform PUPs. This class of PUPs are general enough to produce different types of curves while constraining the problem such that we can solve it eloquently. Moreover, we focus on binary subdivision, although the generalization to n-ary subdivision is possible and straightforward. Our approach for subdividing uniform PUPs is inspired by B-Spline subdivision [28]. In this section, we formulate PUPs subdivision and introduce a framework for utilizing these schemes.

4.1 Deriving PUPs Subdivision

Ideally, given a set of control points, a subdivision scheme increases the number of control points without changing the curve defined by these points. In binary subdivision, the number of control points is doubled through subdivision. Hence, to subdivide a PUPs curve $Q(u)$, we attempt to find another PUPs curve $Q^*(u)$ with twice number of control points, that nevertheless represents the same curve. Then $Q^*(u)$ is a subdivided form of $Q(u)$ if and only if

$$Q(u)=Q^*(u)\,,\quad \text{for all}\ \ u\in[u_l\,,\,u_r]\ . \tag{5}$$

Considering the rational form of PUPs, equality of $\tilde{Q}(u)$ and $\tilde{Q}^*(u)$ (the counterparts of $Q(u)$ and $Q^*(u)$ in the Grassmann space, see Fig. 1) suffices for satisfying Eq. 5.

In other words, if two curves are equal before projection, they will be equal after projection as well. Note that the converse is not necessarily true as different curves in the Grassmann space can produce the same image in affine plane (see Fig. 2). Hence, we

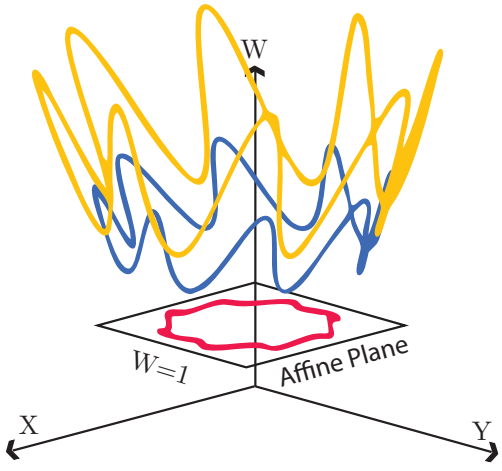

Figure 2: The red curve in affine space results from projecting either the yellow or blue curve.

attempt to subdivide PUPs curves in the Grassmann space. Note

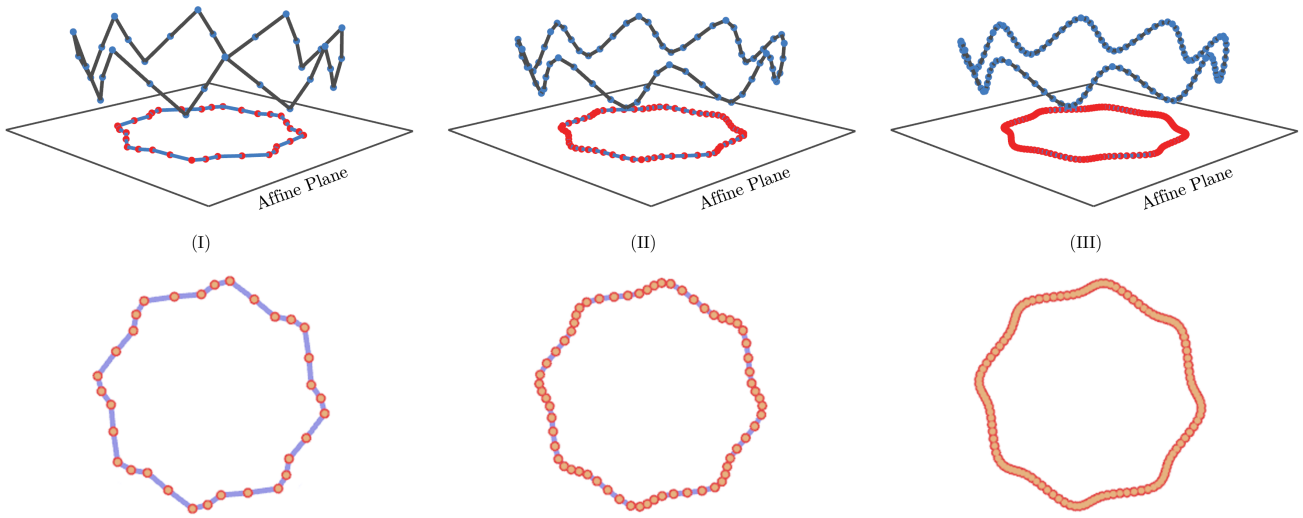

Figure 3: PUPs subdivision process. The first row illustrates subdivision in the Grassmann space. The blue points represent $\mathbf{p}^i$ (which converge to $\tilde{Q}(u)$) and the red points are $\hat{\mathbf{p}}^i$ (which converge to $Q(u)$). The second row illustrates $\hat{\mathbf{p}}^i$ in 2D (i.e. the affine plane).

that similar approaches have been used to subdivide NURBS when the control points have non-uniform weights [12, 19]. Therefore, we need to satisfy

$$\tilde{Q}(u) = \tilde{Q}^*(u)\,, \tag{6}$$

which by Eq. 4 can be written as

$$\mathbf{W}(u)\,\mathbf{p} = \mathbf{W}^*(u)\,\mathbf{p}^*\,, \tag{7}$$

where $\mathbf{W}(u)$ and $\mathbf{p}$ are the weight functions and control points of $Q(u)$, and $\mathbf{W}^*(u)$ and $\mathbf{p}^*$ are those of $Q^*(u)$.

Assuming $Q(u)$ is a uniform PUPs curve, $\mathbf{W}(u)$ consists of translates of $w(u)$:

$$\mathbf{W}(u) = [\ldots,\, w(u+d),\, w(u),\, w(u-d),\, \ldots]\,. \tag{8}$$

Taking inspiration from the derivation of B-Spline subdivision in [28], we define $\mathbf{W}^*(u)$ by uniformly shifting $w(2u)$ (the dilated version of $w(u)$):

$$\mathbf{W}^*(u) = [\ldots,\, w(2u+d),\, w(2u),\, w(2u-d),\, \ldots]\,. \tag{9}$$

We can then subdivide PUPs similar to B-Splines, provided we have a refinement equation for $w(u)$ relating it to its dilates. Finding such an equation for a given $w(u)$ is the main problem in constructing PUPs subdivision schemes. As an exact solution is not always possible, it is important to characterize when Eq. 7 can be solved exactly, and what the best approximation is when it cannot. In the next section, we address this problem and will explain different possible cases in detail. Here, we consider only refinement of $w(u)$ (with respect to d) with the form

$$w(u) = \sum_{-l}^{r} \alpha_i\, w(2u - id)\,, \tag{10}$$

where $\alpha_{-l}, \ldots, \alpha_r$ are scalar coefficients, and l and r indicate the left and right bandwidth respectively. Then, using Eq. 10, we can rewrite each weight function of $\mathbf{W}(u)$ in terms of $\mathbf{W}^*(u)$ by utilizing a refinement matrix R

$$\mathbf{W}(u) = \mathbf{W}^*(u)\,R\,, \tag{11}$$

where each column of R contains $\alpha_{-l}, \ldots, \alpha_r$:

$$R = \begin{bmatrix} & \vdots & \vdots & \\ & \alpha_{-l} & 0 & \\ & \alpha_{-l+1} & 0 & \\ & \alpha_{-l+2} & \alpha_{-l} & \\ \ldots & \vdots & \vdots & \ldots \\ & \alpha_r & \alpha_{r-2} & \\ & 0 & \alpha_{r-1} & \\ & 0 & \alpha_r & \\ & \vdots & \vdots & \end{bmatrix}\,. \tag{12}$$

Note that R is a banded matrix (due to the assumed local support of w(u)) and successive columns of R are identical up to a shift by two rows.

By substituting $\mathbf{W}^*(u)\,R$ for $\mathbf{W}(u)$ in Eq. 7, we derive

$$\mathbf{W}^*(u)\,R\,\mathbf{p} = \mathbf{W}^*(u)\,\mathbf{p}^* \tag{13}$$

and because $\mathbf{W}^*(u)$ consists of non-zero functions, we have

$$\mathbf{p}^* = R\,\mathbf{p}\,. \tag{14}$$

Therefore, the new control points result from multiplying the refinement matrix by the old control points.

4.2 Subdivision Process

Using the matrix R we can subdivide a PUPs curve to the desired level of refinement. Let $\mathbf{p}^0$ denote the initial control points and $\mathbf{p}^i$ denote the control points after i subdivision steps. We define

$$\mathbf{p}^i = R\,\mathbf{p}^{i-1}\,, \tag{15}$$

and by repeated application, we obtain

$$\begin{aligned} \tilde{Q}(u) &= \mathbf{W}(u)\,\mathbf{p}^0 \\ &= \mathbf{W}(2u)\,\mathbf{p}^1 \\ &\vdots \\ &= \mathbf{W}(2^i u)\,\mathbf{p}^i\,. \end{aligned} \tag{16}$$

Assuming $w(u)$ is a continuous function in $\mathscr{L}^2$ space with compact support, the support of $w(2^i u)$ (and its translates in $\mathbf{W}(2^i u)$) converges to zero by successive subdivision [10]. Consequently, the sequence of $\mathbf{p}^i$ converges to $\tilde{Q}(u)$.

Note that $\mathbf{p}^i$ resides in the Grassmann space and hence, each point in $\mathbf{p}^i$ has an associated weight (i.e. homogeneous coordinate). By dividing each point by its weight, we project $\mathbf{p}^i$ to the affine plane and yield $\hat{\mathbf{p}}^i$. As the sequence of $\mathbf{p}^0, \ldots, \mathbf{p}^i$ converges to $\tilde{Q}(u)$, the sequence of $\hat{\mathbf{p}}^0, \ldots, \hat{\mathbf{p}}^i$ converges to $Q(u)$ (see Fig. 3). Moreover, after projecting the points, if we want to subdivide $\hat{\mathbf{p}}^i$ again, we first lift $\hat{\mathbf{p}}^i$ into the Grassmann space and retrieve $\mathbf{p}^i$ (as subdivision is performed in Grassmann space). Then, by subdividing $\mathbf{p}^i$, we produce $\mathbf{p}^{i+1}$ and by projecting $\mathbf{p}^{i+1}$ to affine plane we get $\hat{\mathbf{p}}^{i+1}$. Fig. 4 illustrates this process.

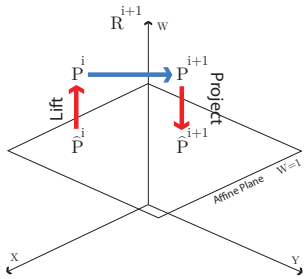

Figure 4: Overview of the PUPs subdivision process. First, multiply the points' coordinates by their homogeneous component to lift them into Grassmann space. Second, subdivide the points. Third, project the points to affine space by dividing their coordinates by their homogeneous component.

5 Finding Refinement Coefficients

We are interested in the solution of Eq. 10. As explained in the previous section, we can subdivide a uniform PUPs, if $\alpha_{-l}, \ldots, \alpha_r$ exist such that Eq. 10 is satisfied. Finding such coefficients is difficult as $w(u)$ can be any function. On the other hand, not all functions are necessarily refinable (i.e. satisfying Eq. 10) and hence, we need a method for identifying these functions.

5.1 Least Squares Based Refinement

Given a uniform PUPs defined by $w(u)$ and d, its identical weight functions are uniformly translated by d. After one subdivision step, the weight functions are dilated by a factor of two and consequently, the corresponding new weight funtions are defined by translating $w(2u)$ with $\frac{d}{2}$. Based on the value of d, each old weight function shares its support with one or more dilated weight function (see Fig. 5 for an example configuration).

The goal of the refinement equation is to represent $w(u)$ in terms of a linear combination of the dilated functions. As $w(u)$ is a function with compact support, only a few dilated functions contribute in the refinement equation. Hence, for any sample parameter $\bar{u}$ in the support $w(u)$, the value of the function satisfies

$$w(\bar{u}) = \begin{bmatrix} w(2\bar{u}+ld) & \ldots & w(2\bar{u}-rd) \end{bmatrix} \begin{bmatrix} \alpha_{-l} \\ \vdots \\ \alpha_r \end{bmatrix}, \qquad (17)$$

where l and r are defined as the largest integers such that $w(2u+ld)$ and $w(2u-rd)$ *completely* reside in the support of $w(u)$.

In general, solving the refinement equation is difficult as $w(u)$ can be any non-linear function. However, if we evaluate Eq. 17 at

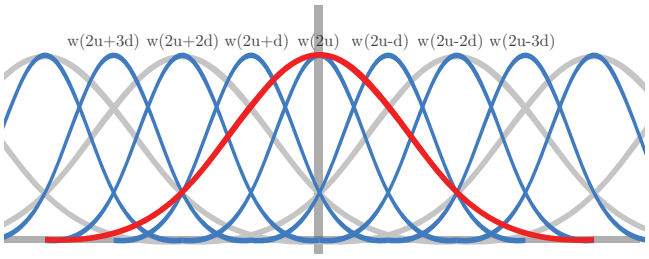

Figure 5: An example function arranged with its dilated forms. The red function represents $w(u)$ and the blue functions are dilated copies of $w(u)$. The gray functions are uniform translates of $w(u)$. Because of refinement, the red function results from a linear combination of the blue functions. Both l and r are 2 in this example since $w(2u+2d)$ and $w(2u-2d)$ completely reside in the red function's support. Note that the support of $w(u)$ only partially covers the support of $w(2u+3d)$ and $w(2u-3d)$.

set of samples, we can form a linear system, which can be used to determine the relation between the original and the dilated weight-functions. Let $[\bar{u}_1 \ldots \bar{u}_s]$ be a *dense* set of samples that are uniformly distributed in the support of $w(u)$. By means of this sampling set, we evaluate $w(u)$ and discretize the function (see. Fig. 6). We assume the sampling set is dense enough that the discretization error becomes relatively small. By evaluating Eq. 17 for each sample,

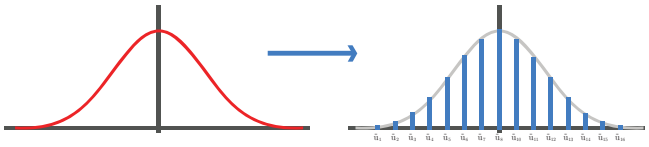

Figure 6: Function Sampling: the red function is discretized by means of 16 samples. Normally, more number of samples are used to minimize the discretization error.

we derive

$$\begin{bmatrix} w(\bar{u}_1) \\ \vdots \\ w(\bar{u}_s) \end{bmatrix} = \begin{bmatrix} w(2\bar{u}_1+ld) & \ldots & w(2\bar{u}_1-rd) \\ \vdots & \ddots & \vdots \\ w(2\bar{u}_s+ld) & \ldots & w(2\bar{u}_s-rd) \end{bmatrix} \begin{bmatrix} \alpha_{-l} \\ \vdots \\ \alpha_r \end{bmatrix}, \qquad (18)$$

which we denote by

$$\bar{\mathbf{w}} = M\,\mathbf{c}\,. \qquad (19)$$

Note that each row of M corresponds to an evaluation of Eq. 10 for one sample, and each column corresponds to the discretization of one dilated weight function.

Provided that the number of samples is larger than the number of coefficients, we will obtain an over-determined linear system. This over-determined system is solved using the pseudo-inverse of M:

$$\mathbf{c} = M^{\dagger}\bar{\mathbf{w}}\,. \qquad (20)$$

As Eq. 18 is over-determined, it might not be possible to solve it exactly. In other words, Eq. 18 is a least squares problem and the solution provided by the pseudo-inverse, minimizes norm-2 of the residual vector

$$\|\mathbf{r}\|_2 = \|\bar{\mathbf{w}} - M\,\mathbf{c}\|_2\,, \qquad (21)$$

which is the least squares error. For a dense enough sample set, if the error is zero, we have an exact solution for the refinement equation and thus, $w(u)$ is refinable. Many functions such as all

polynomials, triangular (hat) functions and B-Splines are refinable (see [27] for more details) and our approach based on least squares produces exact subdivision schemes for theses weight functions. For other functions, the error may not be zero, but by means of the pseudo-inverse, we find the best possible coefficients (in the least-squares sense). This implies that the resulting subdivided curve will deviate from the initial PUPs curve, but it will be the closest possible curve. For practical applications, the difference will be negligible if the least squares error is close to zero.

5.2 Quality Criterion

As an exact solution may not exist, we need a standard criterion to measure the quality of a subdivision scheme and compare different schemes. The least squares error is a candidate since the amount of deviation correlates with it. However, directly using the least squares error has two problems. First, it depends on the number of samples (i.e. the number of M rows) and second, it depends on the magnitude of $w(u)$. For example, while the same subdivision scheme is produced by starting with either $w(u)$ or $2w(u)$, the value of $\|\mathbf{r}\|_2$ is doubled for $2w(u)$. Hence, the least squares error must be normalized to use it as a quality criterion.

To normalize the least squares error, we use the Min-Max normalization method [16] since the resulting equation is also optimized by the least squares solution. The minimum value of $\|\mathbf{r}\|_2$ (obtained by pseudo-inverse) is 0 and the maximum value is $\|\bar{\mathbf{w}}\|_2$ (when all the coefficients are zero). By means of Min-Max normalization method, we define the refinement quality as

$$E = \frac{\|\mathbf{r}\|_2}{\|\bar{\mathbf{w}}\|_2} \,. \tag{22}$$

Note that the value of E is always between 0 and 1, where 0 indicates the best quality and an exact refinement. In addition, since the same number of samples is used for both $\|\bar{\mathbf{w}}\|_2$ and $\|\mathbf{r}\|_2$, the value of E is independent from the number of samples. Fig. 7 shows how different values of E relate to the quality of the resulting refinement scheme.

5.3 The Limit Function of Subdivision

Let $\hat{\alpha}_{-l}, \ldots, \hat{\alpha}_r$ be the scalar coefficients resulting from Eq. 20 and let $\psi(u)$ be a function that satisfies

$$w(u) = \psi(u) + \sum_{-l}^{r} \hat{\alpha}_i \, w(2u - id) \,. \tag{23}$$

When the least squares error is zero, $\psi(u)$ is the zero function and when the error is not zero, $\psi(u)$ is a continuous non-zero function with compact support in $\mathscr{L}^2$ (because of the properties of $w(u)$). We call $\psi(u)$ the *residual function*. It is important to note that $\psi(u)$ has at least the same smoothness as $w(u)$. In addition, since the solution is obtained from least squares, $\psi(u)$ is orthogonal to $\sum_{-l}^{r} \hat{\alpha}_i \, w(2u - id)$ [20] and its norm is less than norm of $w(u)$.

We can recursively expand Eq. 23 through repeated substitution of $w(2u - id)$ for the same equation to derive a general equation. Letting k be a positive integer that indicates the level of recursion, we can rewrite Eq. 23 as

$$w(u) = L(u,k) + R(u,k) \,, \tag{24}$$

where $L(u,k)$ and $R(u,k)$ are defined as

$$L(u,k) = \sum_{i=-(2k-1)l}^{(2k-1)r} P_{k,i}(\hat{\alpha}) w(2^k u - id) \,, \tag{25}$$

$$R(u,k) = \psi(u) + \sum_{j=1}^{k-1} \sum_{i=-(2j-1)l}^{(2j-1)r} P_{j,i}(\hat{\alpha}) \psi(2^j u - id) \,. \tag{26}$$

$P_{k,i}(\hat{\alpha})$ is a polynomial of degree k of the coefficients. More specifically, $P_{k,i}(\hat{\alpha})$ is the sum of all k-multiples of the coefficients such that the indexes of $\alpha_{i_1} \ldots \alpha_{i_k}$ satisfy

$$i_1 2^0 + \ldots + i_k 2^{k-1} = i \,. \tag{27}$$

For example, if l and r are both 2, $P_{2,0}$ is $(\alpha_0^2 + \alpha_2\alpha_{-1} + \alpha_{-2}\alpha_1)$ as $(0,0),(2,-1),(-2,1)$ satisfy $i_1 + 2i_2 = 0$.

Assuming $w(u)$ is C^k continuous, the residual function is also at least C^k continuous and consequently, both $L(u,k)$ and $R(u,k)$ are C^k continuous. In addition, $L(u,k)$ and $R(u,k)$ are orthogonal to each other for any k. By having these properties as well as compact support of $w(u)$ and $\psi(u)$, both $L(u,k)$ and $R(u,k)$ pointwise converge to a function at infinity. Let $L(u)$ be defined as

$$L(u) = \lim_{k \to \infty} L(u,k) \,, \tag{28}$$

which we call the *limit function of subdivision*. $L(u)$ is the actual weight function that is produced as the final result of refinement. In other words, if we consecutively subdivide a set of points by means of $\hat{\alpha}_{-l}, \ldots, \hat{\alpha}_r$, the points converge to a uniform PUPs that is defined by $L(u)$ as the weight function and d as the amount shift. Hence, the resulting final curve of a PUPs subdivision scheme will follow the properties of its limit function (including smoothness). Fig. 7 shows some example weight functions with their corresponding limit functions. Note that when the error is zero $L(u)$ is equal to $w(u)$ and when the error is not zero $L(u)$ is the closest refinable function to $w(u)$ with respect to Eq. 23.

6 Deriving Example PUPs Subdivision Schemes

To produce example PUPs subdivision schemes, we have employed different types of weight functions. In this section, we introduce each weight function, their properties and the specific adjustment we have made to derive subdivision schemes from them.

6.1 B-Spline Basis Functions

B-Spline subdivision schemes are common in computer graphics applications. As a proof of concept and as a way to test consistency of our method, we use B-Spline basis functions as $w(u)$. We aim to yield exactly the known B-Spline subdivision schemes by using least squares.

Let us consider the uniform quadratic B-Spline basis function, which is defined as

$$B(u) = \begin{cases} \frac{1}{2}u^2 & \text{if } 0 \le u \le 1 \\ \frac{1}{2}(-2u^2 + 6u - 3) & \text{if } 1 \le u \le 2 \\ \frac{1}{2}(u^2 - 6u + 9) & \text{if } 2 \le u \le 3 \end{cases} . \tag{29}$$

The active support of $B(u)$ and $B(2u)$ are $(0,3)$ and $(0,1.5)$ respectively. By employing the refinement equation with $d = 1$ we derive

$$B(u) = \alpha_0 B(2u) + \alpha_1 B(2u-1) + \alpha_2 B(2u-2) + \alpha_3 B(2u-3) \,, \tag{30}$$

as $l = 0$ and $r = 3$ according to $B(u)$ support and shift. We can now evaluate the functions and form a least squares system. By sampling the functions at [0.5, 1, 1.5, 2, 2.5] we get

$$\begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0.5 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} 0.125 \\ 0.5 \\ 0.75 \\ 0.5 \\ 0.125 \end{bmatrix} \,, \tag{31}$$

whose least squares solution is [0.25, 0.75, 0.75, 0.25] which is the filter used in Chaikin subdivision (see Fig. 8). Note that while only 5 samples are used, increasing the number of samples yields the same result.

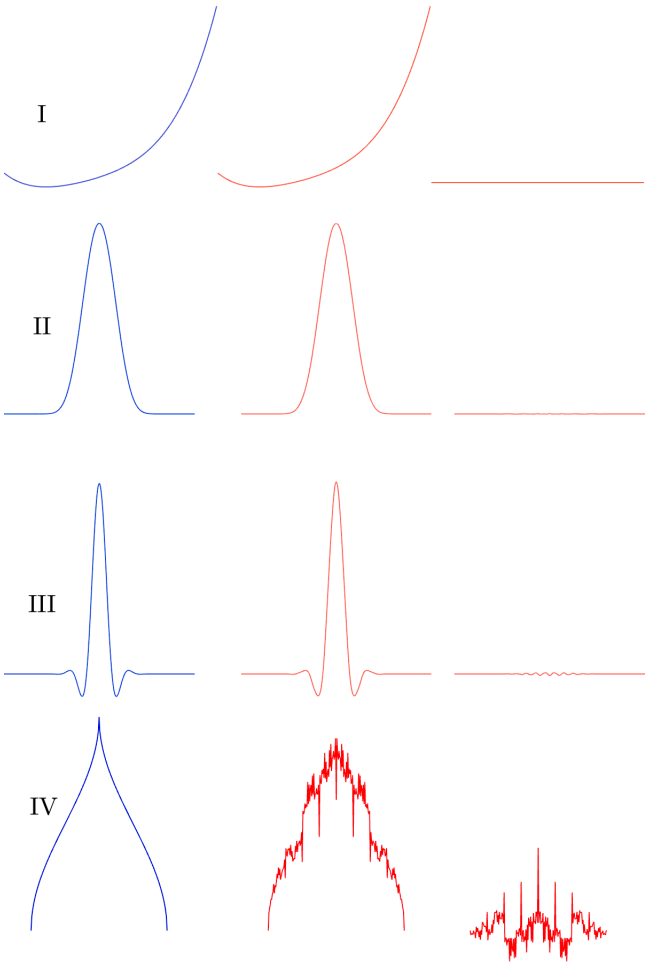

Figure 7: Four weight functions with their corresponding limit functions are shown. From left to right: weight function, limit function, and total residual. Row I illustrates a degree 4 polynomial within $[-1,1]$ with $5.53812E-8$ normalized error. Row II illustrates a C^∞ CINPACT function with 0.000787722 normalized error. Row III illustrates a C^∞ interpolating CINPACT function with 0.00921726 normalized error. Row IV illustrates a C^0 sharp function with 0.0868658 normalized error.

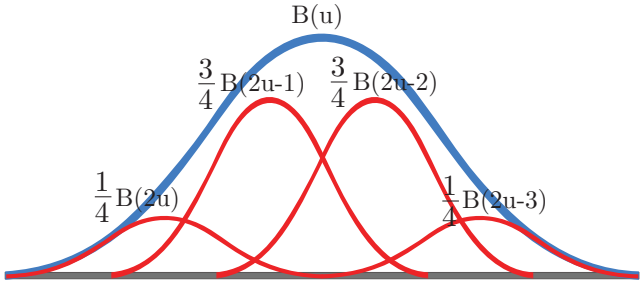

Figure 8: Refinement of a uniform quadratic B-Spline basis function: the blue function represents $B(u)$ and the red functions represent dilated copies of $B(u)$. At each parameter u, the sum of red function values equals the value of $B(u)$.

6.2 Polynomials

Polynomials are the foundation of splines and are widely used in mathematics and engineering. These functions are refinable and PUPs allows us to employ them and derive corresponding subdivision schemes. In general, a polynomial of degree κ is defined as

$$p(u) = a_0 u^0 + a_1 u^1 + \ldots + a_\kappa u^\kappa \,, \tag{32}$$

where $a_0, \ldots, a_\kappa$ are the polynomial coefficients. Although polynomials are refinable, they are not defined in $\mathscr{L}^2$ space. Hence, while polynomials are not bounded to a compact support, we assume $p(u)$ is defined in a bounded domain $[\mu_l\ ,\ \mu_r]$ for the sake of sampling. We also assume a shift of $d=1$. Row I in Fig. 7 shows $1+u+u^2+u^3+u^4$ with corresponding refinement coefficients $[0.0386556,\ -0.123698,\ 0.236816,\ -0.116536,\ 0.0272624]$. As the function lacks symmetry, the resulting refinement scheme is asymmetric (see Table 1 for more results).

6.3 CINPACT

Runions and Samavati introduced a class of C^∞ continuous weight functions in [22] to generate CINPACT splines. Unlike Gaussian functions, CINPACT weight-functions have C^∞ continuity with compact support. These two properties make the function interesting for many applications in geometric modeling, thus motivating us to derive corresponding subdivision schemes. In other words, an exact refinement scheme for CINPACT results in a C^∞ refinement scheme and an approximate one produces curves that are very close to a C^∞ curve.

The C^∞ continuous function they propose is a bump function of the form

$$w(u) = \begin{cases} e^{\frac{-\sigma u^2}{c^2-u^2}} & \text{if } -c < u < c \\ 0 & \text{otherwise} \end{cases} \,, \tag{33}$$

where c adjusts the active support and σ is a parameter. Row II in Fig. 7 illustrates an example bump function with its refinement. In [22], the authors have suggested a list of values for σ and c such the bump function resembles B-Splines when $d=1$. By utilizing those number, we have developed a list of refinement schemes (available in Table 3). Moreover, we also consider CINPACT functions with fixed support and to calculate the refinement coefficients with minimum possible error, we employed an exhaustive search to find the optimal σ for a set of candidates c. The last five rows of Table 3 present the resulting filters.

6.4 CINPACT with Interpolation

Along with the bump function, a class of C^∞ continuous interpolating functions is introduced in [22]. This function is defined by multiplying the bump function by the normalized-sinc function:

$$w(u) = \frac{sin(\pi u)}{\pi u} e^{\frac{-\sigma u^2}{c^2-u^2}} \,. \tag{34}$$

This function provides an interpolating curve when $d=1$ because the sinc function is 1 at $u=0$ and 0 for any other integer u. Hence, the interpolation occurs at any integer u because only one weight function is non-zero at those points.

We utilize this function to produce interpolating subdivision filters. To this end, we have employed hard constraints in our least squares system to ensure interpolation. Let us consider the refinement equation of an example function when $c=2$:

$$\begin{aligned} w(u) =& \alpha_{-2} w(2u+2) + \alpha_{-1} w(2u+1) + \alpha_0 w(2u) \\ &+ \alpha_1 w(2u-1) + \alpha_2 w(2u-2) \end{aligned} \,. \tag{35}$$

To ensure interpolation, the refinement equation must evaluate to 0 at $u=-2,-1,1,2$ (since they are interpolation sites and $w(-1) = w(1) = 0$). At $u=-2,2$ the condition is satisfied. At $u=1$, all the

new functions are zero except $w(2u-2)$. Hence to satisfy interpolation, α_2 must be zero. The same is also true for α_{-2}.

In general, for any interpolating CINPACT, the refinement coefficients with even indexes must be zero. We add these hard conditions to our least squares system before solving it (see [15] for detailed algorithms). Note that by using hard constraints, the residual is no longer necessarily orthogonal to the space of solutions, which is the price of having interpolation. In Table 2, we have provided a list of refinement schemes for different interpolating CINPACT functions.

7 Results

In this section, we provide some example curves that have been produced using the subdivision schemes developed in the previous section. All the provided examples are PUPs curves defined on a circular domain and their corresponding refinement matrices are circulant. Our goal is to reproduce the PUPs curves that are shown in [21, 22] by means of subdivision schemes. In each example figure, the red curve shows the original PUPs curve and its blue points represent the corresponding control points. The result of four consecutive subdivision is illustrated afterwards.

The curves in Fig. 10 and Fig. 11 illustrate the result of applying degree 6 and degree 5 polynomial subdivision filters. Fig. 12 shows the difference between cubic B-Spline, degree 6 ,and degree 10 polynomial subdivisions. While the characteristic of cubic B-Spline subdivision is fixed, we can control the characteristics of polynomial subdivisions by changing polynomial coefficients and degree. As it is illustrated in Fig. 12, degree 6 polynomial produces curves with less energy while both cubic B-Spline and degree 6 polynomial have the same number of refinement coefficients.

In Fig. 14 and Fig. 13 we respectively illustrate the result of applying CINPACT subdivision filters for $c = 2.5, \sigma = 6.28$ and $c = 3.684, \sigma = 17.27$. As it is shown in Table 3, the amount of normalized error is non-zero for these two functions. However, because the error is small, the difference is not visible. During our experiments with different functions, we found that a normalized error less than 0.01 does not seem to affect the visual quality of results. Nonetheless, as it is shown in Fig. 7, for many functions the amount of error is more significant and stationary refinement is insufficient to approximate the original curve well.

Finally, Fig. 15 and Fig. 9 show the result of using interpolating CINPACT subdivision filters. All the control points are interpolated in each step. Note that while the same initial control points are used, the bunny shape is different from Fig. 13 because a different weight function has been employed. Fig. 16 and Fig. 17 depict the difference between an interpolating CINPACT scheme and a 4-point and two 6-point interpolating subdivision schemes. The 4-point subdivision scheme (Dyn–Levin subdivision) is a common interpolating scheme [11], which is C^1 continuous and the two 6-point schemes are C^1 continuous and C^2 continuous respectively. As it is shown, the interpolating CINPACT generates a fairer curve in comparison to these schemes.

8 Conclusion and Future Works

In this paper, we have presented a framework to systematically construct PUPs subdivision schemes. We customize and build our schemes based on a given weight function. By choosing appropriate weight functions, these schemes guarantee special properties such as arbitrary smoothness and interpolation.

At present, our current schemes are all based on uniform refinement. As a future work, we are interested in developing non-uniform subdivision for PUPs as well. By using non-uniform subdivision, we can precisely control the geometric distribution of points and create curves where their spacing is uniform. Additionally, we are interested in extending our method to non-stationary subdivision, where subdivision rules may differ during successive subdivision iterations. Such freedom should enable us to reduce the amount of deviation by regulating refinement coefficients in each iteration. Moreover, we have not addressed the problem of boundary conditions, as arises when open curves are considered. We believe that by adjusting the proposed least squares method, it is possible to derive special filters for boundaries as well.

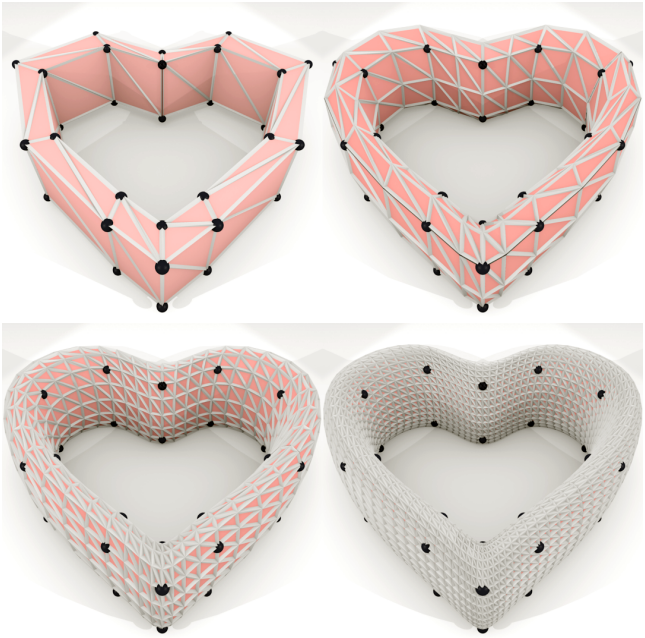

Figure 9: Subdivision of a tensor product mesh by interpolating CINPACT filters ($c = 5$, $\sigma = 4.79$). The black spheres represent the initial points that are interpolated through subdivision. The result of subdivision is smooth and shape-preserving.

Furthermore, it is important that we analyze the smoothness of our subdivision schemes in presence of least squares error. One approach is to prove that the limit function converges *uniformly*, which is a stronger condition and depends on the weight functions. Another approach is to analyze eigen-values of the local refinement matrices [18, 28]. However, existing techniques for analyzing such matrices are not directly applicable to PUPs, because PUPs refinement matrices do not necessarily sum to one (row-wise).

Another key future work is finding subdivision schemes for PUPs surfaces. Finding such a scheme is difficult as PUPs surfaces support control nets with arbitrary connectivity. Nonetheless this would greatly increase the capabilities of PUPs based subdivision by supporting freeform surface modeling.

References

[1] B. Akram, U. Alim, and F. F.Samavati. CINAPACT-splines: A family of infinitely smooth, accurate and compactly supported splines. In *Proceedings of the International Symposium on Visual Computing*, pages 819–829, 2015.

[2] R. H. Bartels and F. F. Samavati. Multiresolutions numerically from subdivisions. *Computers & Graphics*, 35(2):185–197, 2011.

[3] J. Caron and D. Mould. Texture synthesis using label assignment over a graph. *Computers & Graphics*, 39:24–36, 2014.

[4] T. J. Cashman. Beyond Catmull-Clark: A survey of advances in subdivision surface methods. *Computer Graphics Forum*, 31(1):42–61, 2012.

Polynomial	Support	Normalized Error E	Refinement Coefficients
$64-48u^2+12u^4-u^6$	$[-2\ ,\ 2]$	0.00784382	0.218601, 0.487201, 0.593167, 0.487201, 0.218601
$-1024+1280u^2-640u^4+160u^6-20u^8+u^{10}$	$[-2\ ,\ 2]$	0.00911911	0.109218, 0.520687, 0.747163, 0.520687, 0.109218
$-125u^2+75u^3-15u^4+u^5$	$[0\ ,\ 5]$	0.00799114	0.353395, 0.397012, 0.436764, 0.418329, 0.226328, 0.170216
$65536-16384u^2+1536u^4-64u^6+u^8$	$[-4\ ,\ 4]$	0.000986008	0.0989794, 0.163489, 0.251154, 0.314675, 0.343853, 0.314675, 0.251154, 0.163489, 0.0989794

Table 1: Subdivision schemes for example polynomial functions using 100000 samples.

Radius of Support c	σ	Normalized Error E	Refinement Coefficients
5	4.79	0.00353701	0.0240126, 0, -0.129882, 0, 0.606154, 0.99909,0.606154, 0, -0.129882, 0, 0.0240126
6	7.11	0.0030319	0.027396, 0, -0.130614, 0, 0.605189,1.00036, 0.605189, 0, -0.130614, 0, 0.027396
7	7.77	0.00119872	-0.00654163, 0, 0.0402722, 0, -0.145579, 0, 0.611639, 0.999841, 0.611639, 0, -0.145579, 0, 0.0402722, 0, -0.00654163
8	10.11	0.00104171	-0.00811822, 0, 0.0421704, 0, -0.146444, 0, 0.611717, 1.00012, 0.611717, 0, -0.146444, 0, 0.0421704, 0, -0.00811822
9	10.16	0.000448086	0.00240267, 0, -0.0148157, 0, 0.0543849, 0 , -0.158699, 0, 0.616902, 0.999918, 0.616902, 0, -0.158699, 0, 0.0543849, 0, -0.0148157, 0, 0.00240267
10	12.94	0.000384	0.00265512, 0, -0.014863, 0, 0.0536281, 0, -0.157451, 0, 0.616265, 1.00002, 0.616265, 0, -0.157451, 0, 0.0536281, 0, -0.014863, 0, 0.00265512

Table 2: Subdivision schemes for interpolating CINPACT functions using 100000 samples. For each c, the value of σ is chosen to minimize the normalized error.

Radius of Support c	σ	Normalized Error E	Refinement Coefficients
3.684	17.27	0.00234953	0.000624709, 0.0012315, 0.12346, 0.501413, 0.746809, 0.501413, 0.12346, 0.0012315, 0.000624709
5.158	28.48	0.000640194	-3.74974e-005, 6.12085e-005, 0.00978099, 0.122482, 0.4293, 0.674698, 0.53492, 0.201893, 0.0265691, 0.000328433, -6.72036e-006
5.574	28	0.000100758	-6.19362e-006, 2.09293e-005, 0.000824541, 0.0309125, 0.187042, 0.469027, 0.624356, 0.469027, 0.187042, 0.0309125, 0.000824541, 2.09293e-005, -6.19362e-006
5.56	23.83	3.9943e-005	1.54441e-005, -1.29547e-005, 0.0114439, 0.100182, 0.316355, 0.536602, 0.552305, 0.34697, 0.119866, 0.016157, 9.46916e-005, 2.35412e-005
5.626	21.31	1.76028e-005	5.52684e-006, -6.62029e-005, 0.0049863, 0.0611218, 0.224332, 0.438916, 0.541409, 0.438916, 0.224332, 0.0611218, 0.0049863, -6.62029e-005, 5.52684e-006
7.587	39.84	4.19507e-006	1.2717e-007, -3.27226e-007, 1.32139e-005, 0.00150855, 0.0216163, 0.113543, 0.309393, 0.505588, 0.525418, 0.348744, 0.140911, 0.0305733, 0.00265239, 3.89061e-005, -3.05938e-007, 1.95538e-007
5.919	21.05	9.05729e-006	2.34058e-005, -4.50717e-005, 0.00963688, 0.0758585, 0.234371, 0.424405, 0.511503, 0.424405, 0.234371, 0.0758585, 0.00963688, -4.50717e-005, 2.34058e-005
7.365	33.46	1.7112e-006	-1.65303e-007, 1.26294e-006, 0.000105757, 0.00499937, 0.0431399, 0.163356, 0.356599, 0.5025, 0.476659, 0.30249, 0.12126, 0.0266301, 0.00224167, 1.92762e-005, -3.74178e-007, 3.39172e-007
7.579	31.94	6.05072e-007	-4.36887e-007, 1.91219e-006, 0.00011323, 0.00475156, 0.0384021, 0.142051, 0.313574, 0.461845, 0.474848, 0.341854, 0.165946, 0.0492528, 0.00710831, 0.000251175, 1.87592e-006, -6.29036e-007
2.5	6.28	0.00717758	0.0232679, 0.339142, 0.638865, 0.638865, 0.339142, 0.0232679
3	7.27	0.001974	0.00561628, 0.206365, 0.482727, 0.610716, 0.482727, 0.206365, 0.00561628
3.5	7.2	0.000826769	0.00214406, 0.147246, 0.350305, 0.500264, 0.500264, 0.350305, 0.147246, 0.00214406
4	8.67	0.000403903	-0.00103534, 0.0756906, 0.255293, 0.419513, 0.500894, 0.419513, 0.255293, 0.0756906, -0.00103534
4.5	14.78	0.000121679	-0.000332284, 0.00972561, 0.114834, 0.338744, 0.537012, 0.537012, 0.338744, 0.114834, 0.00972561, -0.000332284

Table 3: Subdivision schemes for CINPACT functions using 100000 samples.

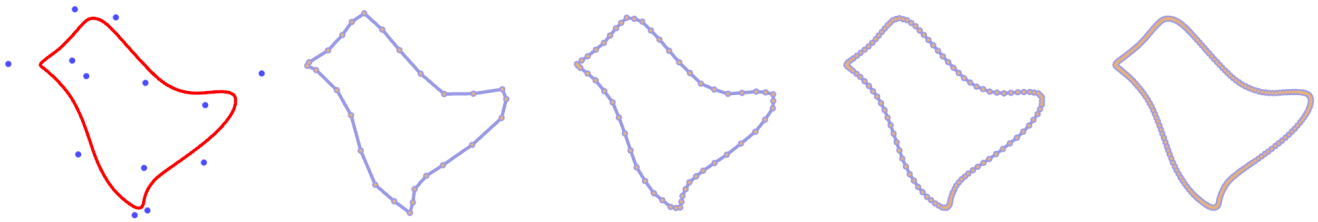

Figure 10: An example of subdivision using degree 6 polynomial filters.

Figure 11: An example of subdivision using degree 5 polynomial filters. The initial control points are obtained from [13].

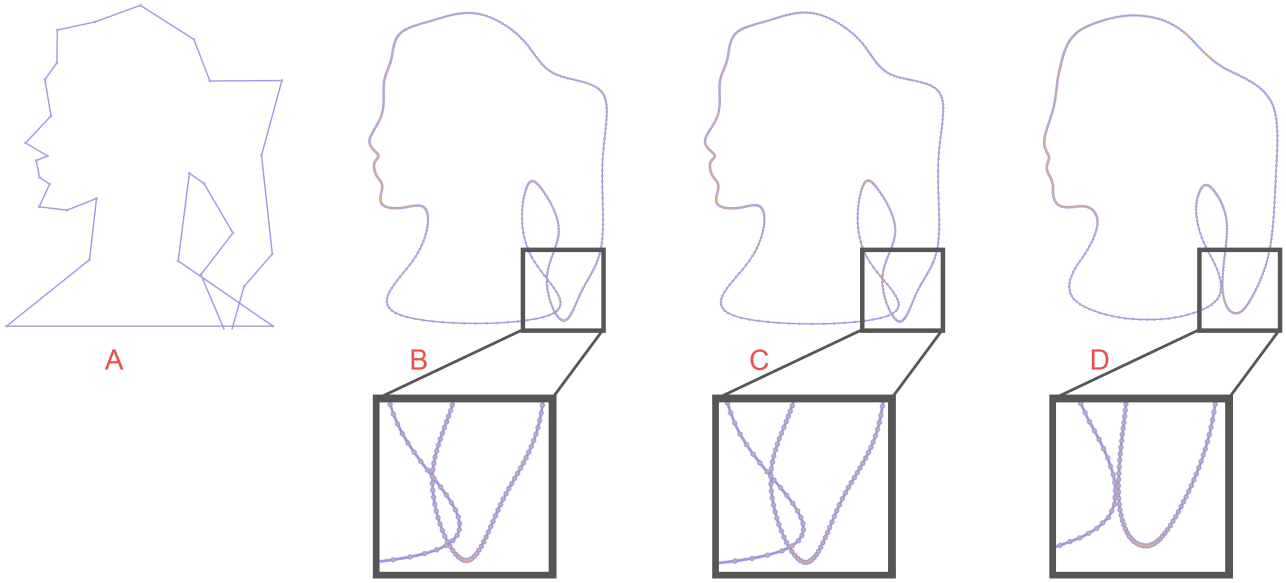

Figure 12: Comparison of cubic B-Spline subdivision and subdivision of polynomials. (A) The initial control net obtained from [13]. (B) The result of applying cubic B-Spline subdivision for four steps. (C) The result of applying degree 10 polynomial subdivision (row 3 of Table 1) for four steps and (D) the result of applying degree 6 subdivision (row 2 of Table 1). Wide range of shapes are generated by changing polynomial coefficients and degree without increasing or decreasing bandwidth. Comparison of (B) and (D) shows that the polynomial subdivision produces curves with less energy than cubic B-Spline subdivision, while they have the same bandwidth.

[5] T. J. Cashman, N. A. Dodgson, and M. A. Sabin. Non-uniform B-spline subdivision using refine and smooth. In *Proceedings of the 12th IMA International Conference on Mathematics of Surfaces XII*, pages 121–137. Springer-Verlag, 2007.

[6] T. J. Cashman, U. H. Augsdörfer, N. A. Dodgson, and M. A. Sabin. NURBS with extraordinary points: High-degree, non-uniform, rational subdivision schemes. *ACM Trans. Graph.*, 28(3):1–9, 2009.

[7] T. J. Cashman, N. A. Dodgson, and M. A. Sabin. Selective knot insertion for symmetric, non-uniform refine and smooth B-spline subdivision. *Comput. Aided Geom. Des.*, 26(4):472–479, 2009.

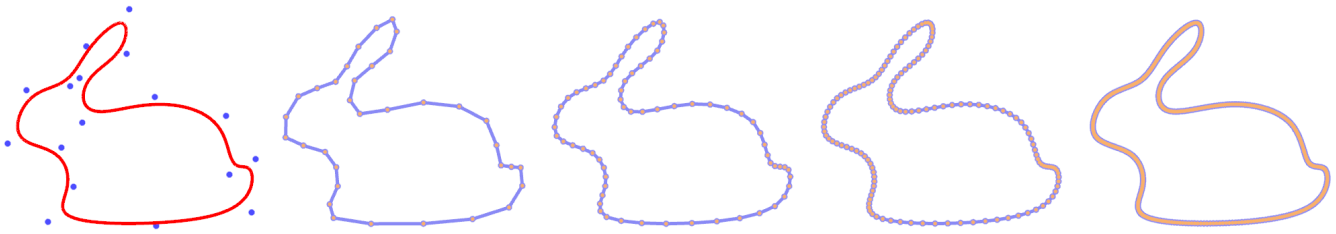

Figure 13: An example of subdivision using CINPACT filters ($c = 3.684$, $\sigma = 17.27$).

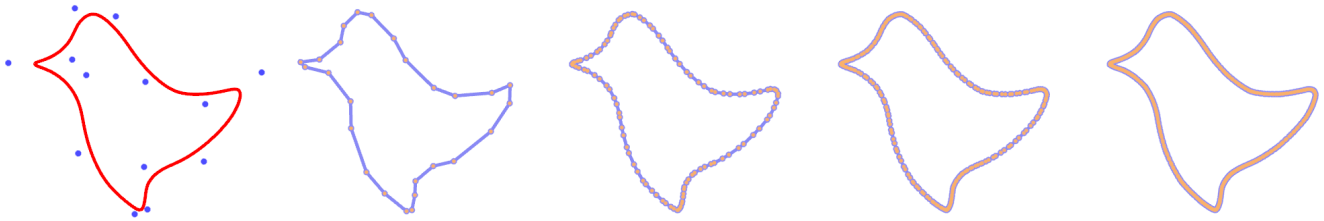

Figure 14: An example of subdivision using CINPACT filters ($c = 2.5$, $\sigma = 6.28$)

Figure 15: An example of subdivision using interpolating CINPACT filters ($c = 7$, $\sigma = 7.77$).

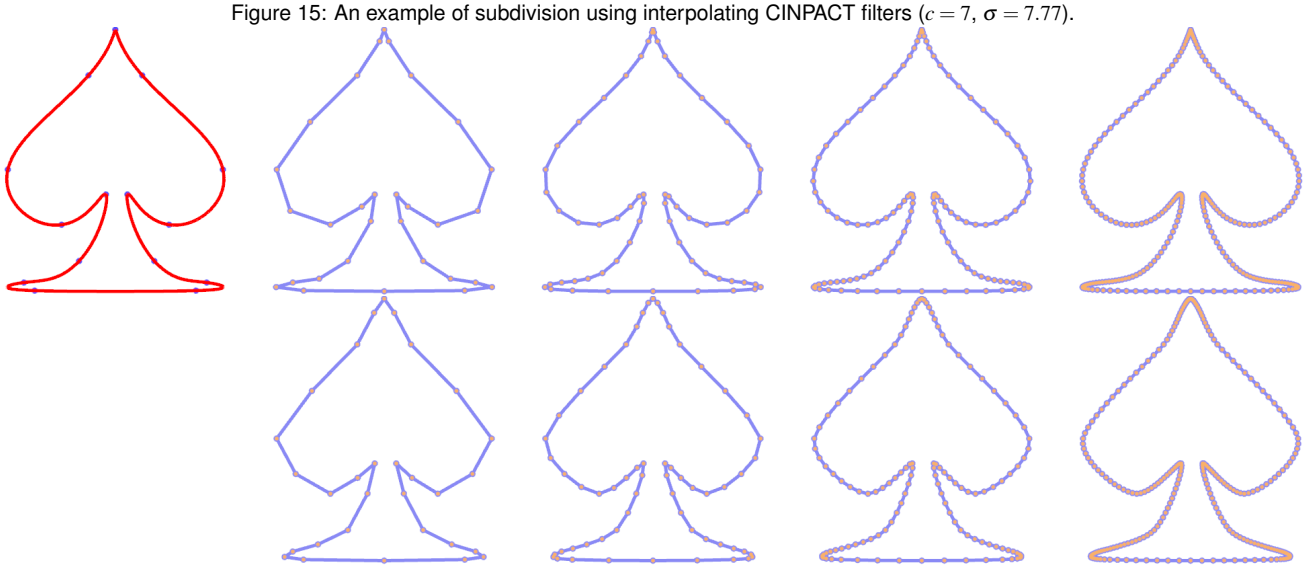

Figure 16: An example subdivision using interpolating CINPACT filters ($c = 5$, $\sigma = 4.79$) compared to the Dyn-Levin 4-point scheme (bottom row).

[8] T. J. Cashman, N. A. Dodgson, and M. A. Sabin. A symmetric, non-uniform, refine and smooth subdivision algorithm for general degree B-splines. *Comput. Aided Geom. Des.*, 26(1): 94–104, 2009.

[9] T. J. Cashman, K. Hormann, and U. Reif. Generalized Lane-Riesenfeld algorithms. *Computer Aided Geometric Design*, 30(4):398 – 409, 2013.

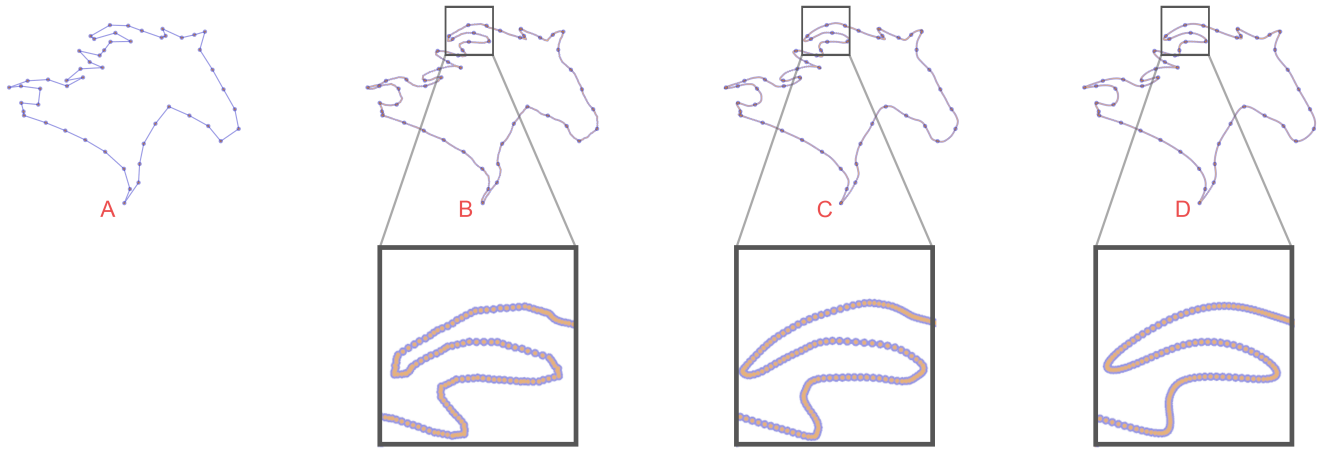

Figure 17: Comparison of CINPACT and other interpolating subdivision schemes after four subdivision steps: the initial control (obtained from [13]) is shown in (A). (B) The result of applying the C^1 continuous 6-point subdivision schemes defined by $0.1, 0, -0.25, 0, 0.65, 1, 0.65, 0, -0.25, 0, 0.1$. (C) The result of applying the C^2 continuous 6-point subdivision schemes defined by $0.05, 0, -0.2125, 0, 0.6625, 1, 0.6625, 0, -0.2125, 0, 0.05$. (D) The result of applying interpolating CINPACT scheme ($c = 5, \sigma = 4.79$). All the filters have the same bandwidth.

[10] C. Chui and Q. Jiang. *Applied Mathematics: Data Compression, Spectral Methods, Fourier Analysis, Wavelets, and Applications*. Mathematics Textbooks for Science and Engineering. Atlantis Press, 2013.

[11] N. Dyn and D. Levin. Subdivision schemes in geometric modelling. *Acta Numerica*, 11:73–144, 2002.

[12] G. E. Farin. *NURB Curves and Surfaces : From Projective Geometry to Practical Use*. A.K. Peters, 1995.

[13] Freepik.com. Free graphics resources, 2016. URL http://www.freepik.com. Online; Accessed: 2016-03-03.

[14] R. Goldman. *Pyramid Algorithms : A Dynamic Programming Approach to Curves and Surfaces for Geometric Modeling*. The Morgan Kaufmann series in computer graphics and geometric modeling. Morgan Kaufmann, 2003.

[15] G. H. Golub and C. F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, 1996.

[16] J. Han, M. Kamber, and J. Pei. *Data Mining, Southeast Asia Edition: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2006.

[17] M. Marinov, N. Dyn, and D. Levin. Geometrically controlled 4-point interpolatory schemes. In N. Dodgson, M. Floater, and M. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 301–315. Springer Berlin Heidelberg, 2005.

[18] C. A. Micchelli and H. Prautzsch. Uniform refinement of curves. *Linear Algebra and its Applications*, 114115(0):841 – 870, 1989.

[19] L. Piegl and W. Tiller. *The NURBS Book (2Nd Ed.)*. Springer-Verlag New York, Inc., 1997.

[20] M. Powell. *Approximation Theory and Methods*. Cambridge University Press, 1981.

[21] A. Runions and F. F.Samavati. Partition of unity parametrics: A framework for meta-modeling. *Vis. Comput.*, 27(6-8):495–505, 2011.

[22] A. Runions and F. Samavati. Cinpact-splines: A class of C^∞ curves with compact support. In *Curves and Surfaces*, volume 9213 of *Lecture Notes in Computer Science*, pages 384–398. Springer International Publishing, 2015.

[23] J. Sadeghi and F. F. Samavati. Smooth reverse subdivision. *Computers & Graphics*, 33(3):217–225, 2009.

[24] J. Sadeghi and F. F. Samavati. Smooth reverse loop and catmull-clark subdivision. *Graphical Models*, 73(5):202–217, 2011.

[25] F. F. Samavati and R. H. Bartels. Multiresolution curve and surface representation: Reversing subdivision rules by least-squares data fitting. *Computer Graphics Forum*, 18(2):97–119, 1999.

[26] S. Schaefer, E. Vouga, and R. Goldman. Nonlinear subdivision through nonlinear averaging. *Computer Aided Geometric Design*, 25(3):162 – 180, 2008.

[27] G. Strang and D.-X. Zhou. The limits of refinable functions. *Transactions of the American Mathematical Society*, 353(5): 1971–1984, 2001.

[28] D. Zorin, P. Schrder, T. DeRose, L. Kobbelt, A. Levin, and W. Sweldens. Subdivision for modeling and animations. In *ACM SIGGRAPH Courses(2000)*, pages 1–194. 2000.