Subdivision and Multiresolution for PUPs

Amirhessam Moltaji[a], Adam Runions[b], Faramarz F. Samavati[a]

[a] *University of Calgary*
[b] *Max Planck Institute for Plant Breeding Research*

Abstract

Partition of Unity Parametrics (PUPs) is a generalization of NURBS that permits the use of arbitrary basis functions to model parametric curves and surfaces. An interesting problem for PUPs is the identification of subdivision, reverse subdivision, and multiresolution schemes for this recently developed and flexible class of parametric curves and surfaces.

In this paper, we introduce a systematic approach to derive uniform subdivision schemes for PUPs curves and tensor-product surfaces. Our approach formulates PUPs subdivision as a least squares problem. This allows us to find exact subdivision filters for refinable basis functions and optimal approximate schemes for irrefinable ones. Additionally, we derive PUPs multiresolution masks based on their subdivision filters. We formulate the problem as a constrained least squares optimization, such that the resulting multiresolution schemes are banded and optimal in terms of minimizing multiresolution reconstruction error.

Finally, to illustrate our methods, we provide sample subdivision and multiresolution schemes with different properties. These include specific examples targeted towards applications of PUPs multiresolution schemes for compression, feature transfer, and macroscopic editing.

Keywords: Subdivision, Multiresolution, Partition of Unity Parametrics, Wavelets

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. 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 often forces modelers to use multiple NURBS patches. Second, NURBS offer limited control over the properties of the parametrics they define. For example, the characteristics 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 [1]. These limitations stem from the use of B-Splines as the underlying basis functions.

Partition of Unity Parametrics (PUPs) was developed to address the limitations of NURBS commented on above. PUPs generalize NURBS by allowing arbitrary basis functions without enforcing any topological restriction [1]. 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 [1] and C^∞ continuity in tandem with compact support [2]. Furthermore, it has been shown that various curves can be generated by fixing control points and simply changing the underlying basis functions [1]. Additional applications of PUPs, such as font modeling, cursive writing, texture synthesis and sketch-based deformation, have also been explored [1, 2, 3].

Given the flexibility of PUPs, an interesting and important problem is to determine subdivision and multiresolution schemes for this class of parametrics. This allows us to take advantage of PUPs flexibility for subdivision and multiresolution whose applications include but are not limited to hierarchical modeling, level-of-detail visualization, feature transfer, and compression.

In this paper, we outline the PUPs subdivision method developed in [4] and propose a method suitable for extending these schemes to multiresolution systems for PUPs curves and tensor-product surfaces. First, we outline a systematic approach to find PUPs subdivision schemes by formulating subdivision as a least squares problem (as originally proposed in [4]). The least squares solution allows us to identify refinable weight functions based on its residual, and also to produce refinement coefficients, which can be used in stationary subdivision schemes. Notably, for irrefinalble functions, our method provides the best possible approximate subdivision schemes. In this extended version of [4], we present a method to calculate PUPs multiresolution schemes from a given subdivision filter. The problem is formulated as a constrained least squares system, which results from considering fundamental conditions of multiresolution, and optimization terms that aim to minimize the multiresolution reconstruction error.

The remainder of the paper is organized as follows. First, we briefly discuss the related works and recent advances in PUPs, subdivision, and multiresolution. Second, we introduce

PUPs. Next, we define PUPs subdivision based on the refinement of PUPs weight functions. Then, we propose a method to derive PUPs subdivision schemes by means of least squares. Afterwards, we introduce the class of functions we have utilized to present example subdivision and multiresolution schemes. Finally, we conclude the paper with a discussion of our results and key directions for future work.

2. Related Work

Our framework is based on PUPs, a generalization of NURBS [1]. The PUPs framework extends NURBS by supporting arbitrary basis functions. Previous works have demonstrated the applicability of PUPs to several aspects of geometric modeling by exploring different types of basis functions. Recently, a new basis function was developed that provides interpolation and C^∞ continuity with compact support [2]. In addition, PUPs have been used for texture synthesis [3] and to derive rendering kernels with arbitrary accuracy order [5]. In contrast, this work focuses on the derivation of PUPs subdivision and multiresolution method, and related work on these two topics which are described in the following subsections.

2.1. Subdivision

Subdivision has become a common technique for shape modeling. These methods, including B-Spline and NURBS subdivision schemes, have been widely studied (see [6, 7, 8, 9] for comprehensive reviews). 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 a well-known 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. [10, 11, 12] 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 [13]. In [14], 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. [15] replace the arithmetic mean typically employed in 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 were not affine-invariant.

In [4], we use the idea of refining basis functions to derive subdivision schemes. This idea was pioneered by Micchelli and Prautzsch, who used refinement of basis functions for the systematic study of stationary subdivision schemes [16]. 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.

2.2. Multiresolution

Multiresolution representation of curves and surfaces is conventionally constructed using the theory of wavelets [17]. In this function-based approach, the scaling functions and their complementary basis functions are used to hierarchically decompose a high resolution model to a low resolution approximation and the missing details. Samavati and Bartels [18, 19] investigated an alternative discrete approach for constructing multiresolution by reversing subdivision rules. In this wavelet free approach, the required multiresolution filters (subdivision, reverse subdivision and their complementary operations) are found directly from some simple matrix computations. Samavati and Bartels showed that the underlying wavelet functions of their reverse subdivision approach are more compact than the ones created through function based approaches [18, 19]. For more general construction, Bartels and Samavati [20] take advantage of SVD decomposition to find all of the required operations. Lifting [21, 22] also appears to be useful in the discrete-based approach. Using lifting, one starts with initial multiresolution filters and alters them (using a lifting matrix) in order to derive higher quality filters.

Constructing PUPs wavelets is not trivial. This becomes more challenging problem when the PUPs basis functions are not always refinable. Therefore, our construction is a wavelet free approach inspired by the method of Samavati and Bartels [18]. We have a novel extension of their work, which enables us to construct PUPs multiresolution such that all of the operations are banded and optimal in terms of minimizing the reconstruction error. In our approach, we consider a constrained least squares system for achieving both goals. To further reduce the residual error, we also provide a systematic method for increasing bandwidth of multiresolution filters. Finally, our new extension allows for the derivation of smooth reverse subdivision filters, whose applications are explored in [23, 24].

3. Partition of Unity Parametrics

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 corresponding weight functions $\{w_1(u), \dots, w_n(u)\}$, a PUPs curve $\mathfrak{Q}(u)$ is defined as

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

where u_l and u_r denote the domain bounds. Each weight function $w_i(u)$ is normalized via division by $\sum_j w_j(u)$ to ensure affine-invariance. To guard against indeterminant forms we assume $\sum_j w_j(u) \neq 0$ for all u.

Although any set of weight functions can be used, PUPs are often constructed from shifted versions of a given function $w(u)$ [1, 2, 5]. Thus, given a scalar value d as shift, $w_i(u)$ is defined as $w(u - id)$ making the corresponding *uniform* PUPs curve

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

As noted in [4], for appropriately chosen $w(u)$, Eq. 2 defines a uniform B-Spline or NURBS curve. Hence, these are special cases of PUPs curves.

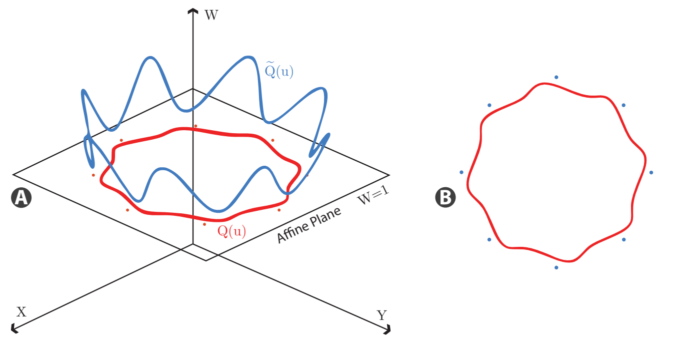


Figure 1: (A) A curve in Grassmann space before normalization (in blue). After normalization the blue curve is projected to affine space and the red PUPs curve is produced. (B) The resulting 2D PUPs curve.

Furthermore, 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, then:

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

where

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

In this definition, $\mathfrak{Q}(u)$ is decomposed to $\tilde{\mathfrak{Q}}(u)$ (a curve in the Grassmann space) which division by $G(u)$ projects to the affine plane [25] (Fig. 1). This decomposition allows us to work on the curve independent from normalization. We employ this definition of PUPs to derive its subdivision in the next section.

4. PUPs Subdivision

In [4], we presented a subdivision scheme for uniform PUPs. This class of PUPs can reproduce arbitary curves while simplifying the problem such that we can solve it eloquently. Moreover, we focused on binary subdivision (although extension to n-ary subdivision is straightforward). Here, we outline the formulation of PUPs subdivision presented in [4].

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 $\mathfrak{Q}(u)$, we attempt to find another PUPs curve $\mathfrak{Q}^*(u)$ with twice the number of control points, that nevertheless represents the same curve:

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

Considering the rational form of PUPs, equality of $\tilde{\mathfrak{Q}}(u)$ and $\tilde{\mathfrak{Q}}^*(u)$ (the counterparts of $\mathfrak{Q}(u)$ and $\mathfrak{Q}^*(u)$ in the Grassmann space, see Fig. 1) suffices for satisfying Eq. 5. Hence, we attempt to subdivide PUPs curves in the Grassmann space. Note that similar approaches have been used to subdivide NURBS when the control points have non-uniform weights [26, 27]. Therefore, we need to satisfy

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

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 $\mathfrak{Q}(u)$, and $\mathbf{W}^*(u)$ and $\mathbf{p}^*$ are those of $\mathfrak{Q}^*(u)$.

Assuming $\mathfrak{Q}(u)$ is a uniform PUPs curve, $\mathbf{W}(u)$ consists of translates of $w(u)$. Furthermore, as we are considering binary subdivision, we define $\mathbf{W}^*(u)$ by uniformly shifting $w(2u)$ (the dilated version of $w(u)$). We can then subdivide $\mathfrak{Q}(u)$, provided we have a refinement equation for $w(u)$ relating it to its dilates. Here, we consider only refinement of $w(u)$ (with respect to d) with the form

$$w(u) = \sum_{-l}^{r} \alpha_i \, w(2u - id) \; , \tag{8}$$

where $\alpha_{-l}, \ldots, \alpha_r$ are scalar coefficients, and l and r indicate the left and right bandwidth respectively. Then, using Eq. 8, we can rewrite each weight function of $\mathbf{W}(u)$ in terms of $\mathbf{W}^*(u)$ by utilizing a refinement matrix R

$$\mathbf{W}(u) = \mathbf{W}^*(u) \, R \; , \tag{9}$$

where each column of R contains $\alpha_{-l}, \ldots, \alpha_r$, and successive columns are identical up to a shift by two rows (note that R is banded, due to the assumed local support of w(u)). It is not always possible to solve Eq. 9 exactly. In the next section, we address this problem and identify optimal masks approximately solving Eq. 9 when exact refinement is impossible.

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

$$\mathbf{W}^*(u) \, R \, \mathbf{p} = \mathbf{W}^*(u) \; \mathbf{p}^* \tag{10}$$

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

$$\mathbf{p}^* = R \, \mathbf{p} \; . \tag{11}$$

Therefore, the new control points result from multiplying the refinement matrix by the old control points. This process can be repeated to further refine the PUPs curve, and after i subdivision steps we obtain:

$$\mathbf{p}^i = R \, \mathbf{p}^{i-1} \; , \tag{12}$$

where p^0 are the original control points. Additionally, it follows that

$$\tilde{\mathfrak{Q}}(u) = \mathbf{W}(2^i u) \; \mathbf{p}^i \; . \tag{13}$$

When $w(u)$ is a continuous function in $\mathscr{L}^2$ space with compact support, the support of $w(2^i u)$ tends to zero as i increases [28]. Consequently, the sequence $\mathbf{p}^i$ converges to $\tilde{\mathfrak{Q}}(u)$.

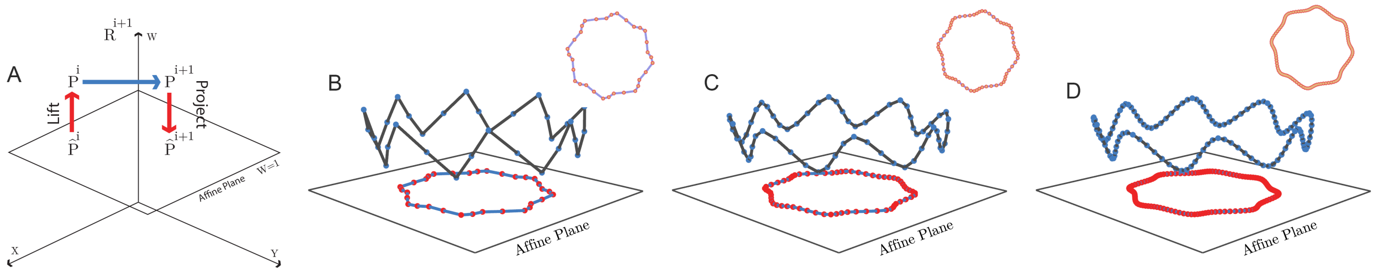


Figure 2: Overview of the PUPs subdivision process (A) illustrated by a simple example (B-D). (A) First, multiply the points' coordinates by their homogeneous component to lift them into Grassmann space. Second, subdivide the points. Third, project the points to affine space by dividing their coordinates by their homogeneous component. (B-D) Three iterations of PUPs subdivision. Subdivision is performed in Grassmann space, and projected to the affine plane (shown from above in insets). Blue points represent $\mathbf{p}^i$ (which converge to $\tilde{\mathfrak{Q}}(u)$) and the red points are $\hat{\mathbf{p}}^i$ (which converge to $\mathfrak{Q}(u)$).

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 $\tilde{\mathfrak{Q}}(u)$, the sequence of $\hat{\mathbf{p}}^0, \ldots, \hat{\mathbf{p}}^i$ converges to $\mathfrak{Q}(u)$ (see Fig. 2). 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 $\hat{\mathbf{p}}^{i+1}$.

4.2. Finding Refinement Coefficients

As explained in the previous section, we can subdivide a uniform PUPs curve if $\alpha_{-l}, \ldots, \alpha_r$ exist such that Eq. 8 is satisfied. Finding such coefficients is difficult as $w(u)$ can be any function. Furthermore, not all functions are refinable (i.e. satisfying Eq. 8), hence we need a method for identifying these functions.

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. 3 for an example configuration). For convenience, we assume $w(u)$ support is a multiple of d which is the case for all example schemes in this article. In case this condition is not satisfied, one possible tactic is to extend $w(u)$ from both ends with zero intervals such that its extended support becomes a multiple of d.

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

$$w(\bar{u}) = \begin{bmatrix} w(2\bar{u}+ld) & \ldots & w(2\bar{u}-rd) \end{bmatrix} \begin{bmatrix} \alpha_{-l} \\ \vdots \\ \alpha_r \end{bmatrix}, \qquad (14)$$

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 $w(u)$.

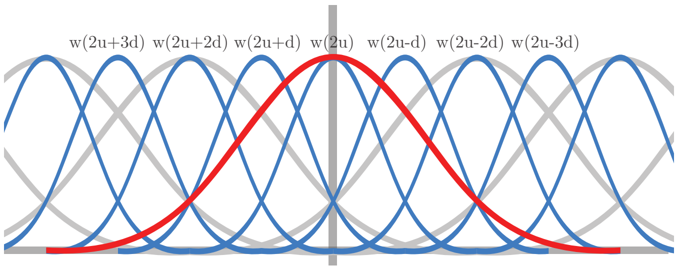


Figure 3: An example function arranged with its dilated forms. The red function represents $w(u)$ and the blue functions are dilated copies of $w(u)$. The gray functions are uniform translates of $w(u)$. Because of refinement, the red function results from a linear combination of the blue functions. 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 $w(2u-3d)$.

In general, solving the refinement equation is difficult as $w(u)$ can be any non-linear function. However, if we evaluate Eq. 14 at a 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 distributed in the support of $w(u)$. By means of this sampling set, we evaluate $w(u)$ and discretize the function (see. Fig. 4). We assume the sampling set is dense enough that the discretization error becomes relatively small. By evaluating

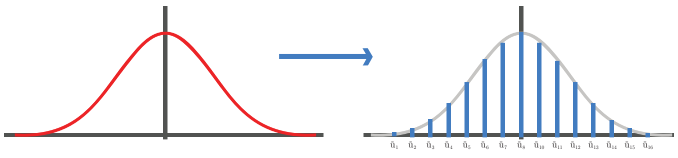


Figure 4: Function Sampling: the red function is discretized by means of 16 samples. Normally, more number of samples are used to minimize the discretization error.

Eq. 14 for each sample, we derive

$$\begin{bmatrix} w(\bar{u}_1) \\ \vdots \\ w(\bar{u}_s) \end{bmatrix} = \begin{bmatrix} w(2\bar{u}_1 + ld) & \dots & w(2\bar{u}_1 - rd) \\ \vdots & \ddots & \vdots \\ w(2\bar{u}_s + ld) & \dots & w(2\bar{u}_s - rd) \end{bmatrix} \begin{bmatrix} \alpha_{-l} \\ \vdots \\ \alpha_r \end{bmatrix}, \quad (15)$$

which we denote by

$$\bar{\mathbf{w}} = M\,\mathbf{c}\,. \quad (16)$$

Note that each row of M corresponds to an evaluation of Eq. 8 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:

$$\mathbf{c} = M^{\dagger}\bar{\mathbf{w}}\,. \quad (17)$$

As Eq. 15 is over-determined, it might not be possible to solve it exactly. In other words, Eq. 15 is a least squares problem and the solution provided by the pseudo-inverse, minimizes norm-2 of the residual vector

$$||\varepsilon||_2 = ||\bar{\mathbf{w}} - M\,\mathbf{c}||_2\,, \quad (18)$$

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 [29] 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.

To ascertain the quality of derived masks we normalize the residual error $||\varepsilon||_2$ by $||\bar{\mathbf{w}}||_2$, yeilding:

$$E = \frac{||\varepsilon||_2}{||\bar{\mathbf{w}}||_2}\,. \quad (19)$$

This provides an error measure which is statistically independent of the number of samples used to discritize w, as well as the magnitude of w. 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 $||\varepsilon||_2$, the value of E is independent from the number of samples.

4.3. Example PUPs Subdivision Schemes

In [4] we considered several classes of weight function and derived corresponding PUPs subdivision schemes. These included the rederivation of B-Spline subdivision filters, which served to demonstrate the correctness of our proposed method. We additionally, considered polynomial basis functions, as well as CINPACT approximating and interpolating functions [2]. Here, we summarize the resulting subdivision schemes. These schemes are then used as the basis for the multiresolution filters we derive in subsequent sections.

Polynomials. We denote a polynomial of degree κ as

$$\mathfrak{p}(u) = a_0u^0 + a_1u^1 + \ldots + a_\kappa u^\kappa\,, \quad (20)$$

where $a_0,\ \ldots,\ a_\kappa$ are the polynomial coefficients. All polynomials are refinable, but do not have compact support. Consequently, we assume $\mathfrak{p}(u)$ is defined in a bounded domain $[\mu_l\,,\,\mu_r]$ for the sake of sampling. We also assume a shift of $d = 1$.

CINPACT (approximating). Introduced by Runions and Samavati [2] as the basis for CINPACT splines, these weight-functions have C^∞ continuity with compact support (two important properties for geometric modeling). The C^∞ continuous function they propose are bump functions of the form

$$w(u) = \begin{cases} e^{\frac{-\sigma u^2}{c^2-u^2}} & \text{if } -c < u < c \\ 0 & \text{otherwise} \end{cases}, \quad (21)$$

where c adjusts the active support and σ is a parameter. Refinement coefficients for several CINPACT weight functions are provided in supplementary materials.

CINPACT (interpolating). Runions and Samavati [2] also propose a class of C^∞ continuous interpolating functions with compact-support. This function is defined by multiplying the bump function w_B from Eq. 21 by the normalized-sinc function:

$$w(u) = \frac{sin(\pi u)}{\pi u} w_B(u)\,. \quad (22)$$

This function creates interpolating curves when $d = 1$, as the sinc function is 1 at $u = 0$ and 0 for other integers (hence, only one weight-function is active at integer parameter values - forcing interpolation). In this case, to obtain interpolating subdivision filters hard constraints are introduced into Eq. 16 (see [4] for details). We have provided a list of refinement schemes for different interpolating CINPACT functions in supplementary materials. As an important example, subdivision filter (refinement coefficients) of interpolating CINPACT with $c = 5$ and $\sigma = 4.79$ is $[0.0240126, 0, -0.129882, 0, 0.606154, 0.99909, 0.606154, 0, -0.129882, 0, 0.0240126]$. This subdivision has been used for creating several examples in Section 6.

5. PUPs Multiresolution

Multiresolution techniques provide a means to change the resolution of curves and surfaces. In previous sections, we outlined the PUPs subdivision method presented in [4]. These subdivision schemes allow us to increase resolution by introducing additional control points. Thus, to obtain a multiresolution framework, we must now define complementary operations that enable us to decrease resolution, reversing the results of subdivision. This requires a method for deriving a corresponding reverse subdivision scheme for a given PUPs subdivision filter.

As explained in Section 2, there exist two main approaches for deriving multiresolution filters. In this paper, we employ the discrete approach (reverse subdivision) considering the following reasons. First, it is not clear how to directly deriving wavelet functions for a general scaling function resulting from PUPs. These functions sometimes are not even refinable. However, in the reverse subdivision method we can generalize some simple matrix computations for constructing multiresolution filters. Interestingly, as shown in [18, 19], the wavelet functions that resulted from the reverse subdivision filters are optimal in a least square sense. Notice that for construction multiresolution representation, we do not directly need wavelet functions and multiresolution filters are sufficient. Second, the flexibility and simplicity of the numerical approach enables us to search for banded and optimal filters as we explain in the following sections.

5.1. *Definitions*

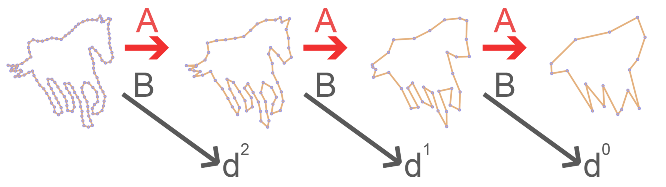


Figure 5: Repetitive decomposition of a curve: the image illustrates the results of decomposing a curve, where the A and B masks are used successively and d^i denotes the details at iteration i. The initial curve was obtained from [30].

In this section we introduce the mathematical preliminaries and definitions we use to derive PUPs multiresolution schemes based on the definitions and notations employed in [18].

Let $\mathbf{f}$ be a vector of control points. Using reverse subdivision, we find a vector of control points $\mathbf{c}$ (with lower resolution) via application of a reverse subdivision mask A (see Fig. 5):

$$\mathbf{c} = A\mathbf{f}\,. \tag{23}$$

For a $\mathbf{c}$ obtained from Eq. 23, an important question is whether or not we can reconstruct $\mathbf{f}$ by applying a subdivision mask. In general, it is not possible to exactly reconstruct $\mathbf{f}$ from $\mathbf{c}$ [20, 18] (regardless of the specified A) as some information may be lost when the resolution is decreased. Therefore, subdividing $\mathbf{c}$ typically only generates an approximation of $\mathbf{f}$, here denoted as $\tilde{\mathbf{f}}$:

$$\tilde{\mathbf{f}} = P\mathbf{c}\,. \tag{24}$$

Note that in the notation of [18], P is the refinement matrix R introduced in Eq. 9.

Although $\mathbf{f}$ cannot be reconstructed from $\mathbf{c}$ alone, if the missing information is preserved, an exact reconstruction of $\mathbf{f}$ is possible. Letting $\mathbf{r}$ denote the difference between $\mathbf{f}$ and $\tilde{\mathbf{f}}$:

$$\mathbf{r} = \mathbf{f} - \tilde{\mathbf{f}}\,, \tag{25}$$

we call $\mathbf{r}$ the reconstruction residual. Given $\mathbf{r}$, we can reconstruct $\mathbf{f}$ as $\mathbf{r} + P\mathbf{c}$.

$\mathbf{r}$ has the same dimension as $\mathbf{f}$, but it is possible to represent $\mathbf{r}$ as a coarse data vector or $\mathbf{d}$ (which is called the detail vector) using a subdivision-like matrix Q:

$$\mathbf{r} = Q\mathbf{d}\,. \tag{26}$$

The structure of Q is similar to that of P in that it increases the resolution of the details. Note that it has been proven that Q and $\mathbf{d}$ exist such that $\mathbf{r}$ can be reconstructed [20].

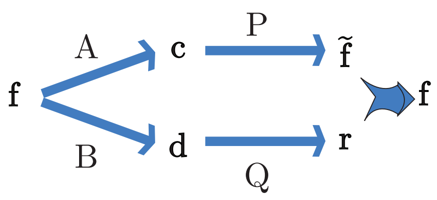
For binary subdivision, $\mathbf{d}$ and $\mathbf{c}$ have the same dimension (half that of $\mathbf{f}$). Thus, we can exactly reconstruct $\mathbf{f}$ without additional data:

$$P\mathbf{c} + Q\mathbf{d} = \tilde{\mathbf{f}} + \mathbf{r} = \mathbf{f}\,. \tag{27}$$

Finally, the details $\mathbf{d}$ are calculated from $\mathbf{f}$ using the matrix B:

$$\mathbf{d} = B\mathbf{f}\,. \tag{28}$$

In summary, given a high resolution data vector $\mathbf{f}$, two masks A and B may be applied to $\mathbf{f}$ in order to obtain coarse points $\mathbf{c}$ and details $\mathbf{d}$, respectively. Then, to reconstruct $\mathbf{f}$, $\mathbf{c}$ and $\mathbf{d}$ may be subdivided using P and Q and summed (Fig. 6).



$$\begin{tikzcd} & \mathbf{c} \arrow[r, "P"] & \tilde{\mathbf{f}} \arrow[dr] & \\ \mathbf{f} \arrow[ur, "A"] \arrow[dr, "B"'] & & & \mathbf{f} \\ & \mathbf{d} \arrow[r, "Q"'] & \mathbf{r} \arrow[ur] & \end{tikzcd}$$

Figure 6: $\mathbf{f}$ is decomposed to $\mathbf{c}$ and $\mathbf{d}$ using A and B respectively. Then $\mathbf{f}$ is reconstructed by subdividing $\mathbf{c}$ and $\mathbf{d}$ using P and Q respectively.

These multiresolution masks have one important property. The block matrix formed by P and Q is the inverse of that formed by A and B:

$$\begin{bmatrix} P & | & Q \end{bmatrix} \begin{bmatrix} A \\ \hline B \end{bmatrix} = I = \begin{bmatrix} A \\ \hline B \end{bmatrix} \begin{bmatrix} P & | & Q \end{bmatrix}\,. \tag{29}$$

Deriving masks for P, Q, A, and B such that they satisfy Eq. 29, is the main challenge in obtaining a discrete multiresolution system. The solution is not unique and different criteria can be used to develop these masks. Minimizing $\|\mathbf{r}\|_2$ is an important criteria [18], which we use along with bandedness to derive PUPs multiresolution.

5.2. *Optimal Banded Multiresolution*

Given a subdivision filter of length k, we aim to find filters for A, B, and Q that minimize the residual r while also being banded. Thus, P and Q have the form

$$P = \begin{bmatrix} & \vdots & \vdots & \\ & p_1 & 0 & \\ & p_2 & 0 & \\ & p_3 & p_1 & \\ \dots & \vdots & \vdots & \dots \\ & p_k & p_{k-2} & \\ & 0 & p_{k-1} & \\ & 0 & p_k & \\ & \vdots & \vdots & \end{bmatrix}, \quad Q = \begin{bmatrix} & \vdots & \vdots & \\ & q_1 & 0 & \\ & q_2 & 0 & \\ & q_3 & q_1 & \\ \dots & \vdots & \vdots & \dots \\ & q_k & q_{k-2} & \\ & 0 & q_{k-1} & \\ & 0 & q_k & \\ & \vdots & \vdots & \end{bmatrix}, \tag{30}$$

constructed by shifting a column filter. Similarly, A and B have the form

$$A = \begin{bmatrix} & & & & \vdots & & & & \\ \dots & a_1 & a_2 & a_3 & \dots & a_k & 0 & 0 & \dots \\ \dots & 0 & 0 & a_1 & \dots & a_{k-2} & a_{k-1} & a_k & \dots \\ & & & & \vdots & & & & \end{bmatrix}, \quad (31)$$

$$B = \begin{bmatrix} & & & & \vdots & & & & \\ \dots & b_1 & b_2 & b_3 & \dots & b_k & 0 & 0 & \dots \\ \dots & 0 & 0 & b_1 & \dots & b_{k-2} & b_{k-1} & b_k & \dots \\ & & & & \vdots & & & & \end{bmatrix}, \quad (32)$$

obtained by shifting row filters. To simplify derivations, we assume k is always even (for odd k an equivalent even length filter can be generating by appending a zero to the mask).

Our filters must satisfy Eq. 29, thus

$$\begin{bmatrix} A \\ B \end{bmatrix} \begin{bmatrix} P & Q \end{bmatrix} = \begin{bmatrix} AP & AQ \\ BP & BQ \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}, \quad (33)$$

which provides four conditions that must be satisfied:

$$AP = I,\ AQ = 0,\ BP = 0,\ BQ = I\,. \quad (34)$$

Below we present the general method we use to derive multiresolution filters. As a simple illustration of the method we calculate the multiresolution filters for the Chaikin subdivision filter (0.25, 0.75, 0.75, 0.25) in parallel with our derivation. Using the condition $BP = 0$ we first aim to derive B. This condition implies that B forms the null space of P^T and is thus satisfied by defining (a similar construction is used in [18]):

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{k-1} \\ b_k \end{bmatrix} = \begin{bmatrix} -p_k \\ p_{k-1} \\ \vdots \\ -p_2 \\ p_1 \end{bmatrix}, \quad (35)$$

For example, assume P_C is a circulant Chaikin subdivision matrix with three columns:

$$P_C = \begin{bmatrix} 0.25 & 0 & 0.75 \\ 0.75 & 0 & 0.25 \\ 0.75 & 0.25 & 0 \\ 0.25 & 0.75 & 0 \\ 0 & 0.75 & 0.25 \\ 0 & 0.25 & 0.75 \end{bmatrix}. \quad (36)$$

Then, based on Eq. 35, the corresponding B_C is

$$B_C = \begin{bmatrix} -0.25 & 0.75 & -0.75 & 0.25 & 0 & 0 \\ 0 & 0 & -0.25 & 0.75 & -0.75 & 0.25 \\ -0.75 & 0.25 & 0 & 0 & -0.25 & 0.75 \end{bmatrix}. \quad (37)$$

Given B, we can then derive Q using the fourth condition $BQ = I$. For the Chaikin example, this condition becomes

$$B_C \begin{bmatrix} q_1 & 0 & q_3 \\ q_2 & 0 & q_4 \\ q_3 & q_1 & 0 \\ q_4 & q_2 & 0 \\ 0 & q_3 & q_1 \\ 0 & q_4 & q_2 \end{bmatrix} = I\,. \quad (38)$$

Which, due to the regular structure of B_C and Q_C, reduces to:

$$\begin{bmatrix} -0.75 & 0.25 & 0 & 0 \\ -0.25 & 0.75 & -0.75 & 0.25 \\ 0 & 0 & -0.25 & 0.75 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad (39)$$

This equation can be solved to obtain the filter for Q_C. The fourth condition $BQ = I$ implies that multiplying B by a column of Q yields an elementary basis vector. As the columns of Q are shifted copies, this constraint reduces to the following system

$$\begin{bmatrix} b_{k-1} & b_k & 0 & \dots & 0 & 0 & 0 \\ & & & \vdots & & & \\ b_3 & b_4 & b_5 & \dots & b_k & 0 & 0 \\ b_1 & b_2 & b_3 & \dots & b_{k-2} & b_{k-1} & b_k \\ 0 & 0 & b_1 & \dots & b_{k-4} & b_{k-3} & b_{k-2} \\ & & & \vdots & & & \\ 0 & 0 & 0 & \dots & 0 & b_1 & b_2 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_k \end{bmatrix} = \mathbf{e}_{\frac{k}{2}}, \quad (40)$$

where $\mathbf{e}_{\frac{k}{2}}$ is the $\frac{k}{2}$ elementary basis vector (with $k-1$ elements). Eq. 40 is under-determined, thus there exists many suitable filters for Q. Consequently, we choose the filter that minimizes the residual while satisfying Eq. 40.

As shown in [18], the minimum reconstruction residual is attained when Q is the orthogonal complement of P (i.e. $P^T Q = 0$). By adding this constraints to Eq. 40 we aim to support orthogonality in the resulting filter:

$$\begin{bmatrix} b_{k-1} & b_k & 0 & \dots & 0 & 0 & 0 \\ & & & \vdots & & & \\ b_3 & b_4 & b_5 & \dots & b_k & 0 & 0 \\ b_1 & b_2 & b_3 & \dots & b_{k-2} & b_{k-1} & b_k \\ 0 & 0 & b_1 & \dots & b_{k-4} & b_{k-3} & b_{k-2} \\ & & & \vdots & & & \\ 0 & 0 & 0 & \dots & 0 & b_1 & b_2 \\ p_{k-1} & p_k & 0 & \dots & 0 & 0 & 0 \\ & & & \vdots & & & \\ p_3 & p_4 & p_5 & \dots & p_k & 0 & 0 \\ p_1 & p_2 & p_3 & \dots & p_{k-2} & p_{k-1} & p_k \\ 0 & 0 & p_1 & \dots & p_{k-4} & p_{k-3} & p_{k-2} \\ & & & \vdots & & & \\ 0 & 0 & 0 & \dots & 0 & p_1 & p_2 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_k \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \quad (41)$$

This new linear system is now over-determined with two groups of constraints. The first ensures that the masks are banded and satisfy $BQ = I$, whereas the second (shown in red) ensures minimal residual. As regular banded masks are necessary in our construction, we treat the first group of constraints as hard constraints and the second group as soft constraints. Then, by solving a constrained least squares system [31] we calculate the best Q filter.

For the Chaikin example, adding orthogonality constraints to Eq. 39 yeilds the following constrained least squares system

$$\begin{bmatrix} -0.75 & 0.25 & 0 & 0 \\ -0.25 & 0.75 & -0.75 & 0.25 \\ 0 & 0 & -0.25 & 0.75 \\ 0.75 & 0.25 & 0 & 0 \\ 0.25 & 0.75 & 0.75 & 0.25 \\ 0 & 0 & 0.25 & 0.75 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} . \quad (42)$$

The corresponding solution is

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} 0.25 \\ 0.75 \\ -0.75 \\ -0.25 \end{bmatrix} \quad (43)$$

with a least squares error of 0.53033.

Once the B and Q filters are calculated, we can find the A filter. We have two conditions for A: $AP = I$ and $AQ = 0$. The orthogonality condition $AQ = 0$ indicates that we can calculate the A filter by flipping and alternating the signs of the Q filter:

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{k-1} \\ a_k \end{bmatrix} = \begin{bmatrix} q_k \\ -q_{k-1} \\ \vdots \\ q_2 \\ -q_1 \end{bmatrix} , \quad (44)$$

to satisfy this condition. Note that the order of the sign alternation differs from Eq. 35. This solution must also satisfy $AP = I$. We prove this condition holds by showing that $AP = BQ$ (as $BQ = I$ by construction) when Eq. 44 is used as the A filter.

Let F be a square anti-diagonal matrix with an even number of rows whose *NE-SW* diagonal consists of alternating -1 and 1 entries:

$$F = \begin{bmatrix} & \vdots & \vdots & \vdots & \iddots \\ \dots & 0 & 0 & -1 & \dots \\ \dots & 0 & 1 & 0 & \dots \\ \dots & -1 & 0 & 0 & \dots \\ \iddots & \vdots & \vdots & \vdots & \end{bmatrix} . \quad (45)$$

Note that $F^{-1} = -F$, and that multiplying a vector $\mathbf{v}$ by F yields

$$F \begin{bmatrix} v_1 \\ \vdots \\ v_k \end{bmatrix} = \begin{bmatrix} -v_k \\ v_{k-1} \\ \vdots \\ -v_2 \\ v_1 \end{bmatrix} , \quad (46)$$

the vector whose entries are the inverted entries of $\mathbf{v}$ with alternated signs.

Now starting with the transpose of AP we obtain:

$$(AP)^T = P^T I A^T = -P^T F F A^T , \quad (47)$$

Multiplying P^T by $-F$ yields B as the rows of P^T are flipped and its elements sign is altered. In addition, multiplying F by A^T yields Q as columns of A^T are flipped and its elements sign is altered. Therefore, $AP = BQ$ and $BQ = I$ by constructoin, thus satisfying the condition $AP = I$.

Based on Eq. 44, for the Chaikin example we have

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} -0.25 \\ 0.75 \\ 0.75 \\ -0.25 \end{bmatrix} , \quad (48)$$

which is the final required reverse subdivision filter. These Chaikin filters are exactly the same as those derived by Samavati and Bartels [32] and are guaranteed to provide the minimum reconstruction error (in sense of local least squares). This indicates that our method is consistent with previous multiresolution techniques if they are banded and optimal.

5.3. Shifting Columns

In the previous subsection, we explained how to find banded multiresolution masks by solving a constrained least squares system. In this subsection, we aim to improve our multiresolution filters by considering other arrangements for these masks. As a motivating example, let us consider B_C from the Chaikin example (Eq. 37) and shift its columns two times to the right

$$B_C = \begin{bmatrix} 0 & 0 & -0.25 & 0.75 & -0.75 & 0.25 \\ -0.75 & 0.25 & 0 & 0 & -0.25 & 0.75 \\ -0.25 & 0.75 & -0.75 & 0.25 & 0 & 0 \end{bmatrix} , \quad (49)$$

or to the left

$$B_C = \begin{bmatrix} -0.75 & 0.25 & 0 & 0 & -0.25 & 0.75 \\ -0.25 & 0.75 & -0.75 & 0.25 & 0 & 0 \\ 0 & 0 & -0.25 & 0.75 & -0.75 & 0.25 \end{bmatrix} . \quad (50)$$

Both of these arrangements are also valid banded multiresolution masks. Here we consider all such shifted arrangements to find the one minimizing the least squares error in Eq. 41, thus improving the quality of the resulting multiresolution masks. To shift the columns of A and B by $2i$ we multiply these matricies by the elementary matrix E_i:

$$e_{k,l} = \begin{cases} 1 & \text{if} \quad l - k = 2i \\ 0 & \quad \text{o.w} \end{cases} , \quad (51)$$

where i is an integer, $i > 0$ shifts to the left and $i < 0$ to the right (note that $E_0 = I$). Hence, the results of right-side multiplication by E_i

$$\begin{aligned} A_i &= AE_i \\ B_i &= BE_i \end{aligned} , \quad (52)$$

are still banded, but differ in the starting entry for the non-zero band.

First, let us check how shifting impacts the filter values. Replacing B with B_i, the third condition $BP = 0$ provides

$$\begin{aligned} P^T B_i^T &= 0 \\ P^T E_i^T B^T &= 0 \end{aligned} . \tag{53}$$

Thus, multiplying E_i^T by B^T shifts the rows of B^T $2i$ times. However, Eq. 35 still provides a valid solution for B_i.

Considering the fourth condition $BQ = I$, we derive

$$\begin{aligned} B_i Q &= I \\ B E_i Q &= I \end{aligned} . \tag{54}$$

Multiplying E_i by Q shifts the rows of Q $2i$ times up or down. As the result of this multiplication, the right hand side of Eq. 40 changes (however, the left hand side is unchanged). Consequently, the filter for Q is obtained by solving Eq. 40 after replacing $\mathbf{e}_{\frac{k}{2}}$ with $\mathbf{e}_s$ (where $s = \frac{k}{2} + i^{\text{th}}$). The filter therefore depends on E_i (i.e. *half* of the shift value). Thus, the original equation (Eq. 40) is obtained when we use E_0.

We are free to use different elementary vectors on the right hand side. This new degree of freedom allows us to further minimize the residual by varying the shift i in E_i. First, note that we are free to use $k-1$ possible elementary vectors, which implies that we can shift A and B columns at most $\frac{k}{2} - 1$ times to the right or left. For each of these under-determined linear systems, we can form a constrained least squares problem similar to Eq. 41 and calculate a solution. Each least squares solution has an associated least squares error in terms of satisfying the soft constraints. We call this error the *orthogonality error*. Using this error as a criterion, we select the solution that is associated with the minimum least squares error. Such a solution satisfies the orthogonality condition $P^T Q = 0$ better than the other solutions and, consequently minimizes the reconstruction residual.

After calculating the optimal Q filter, we construct the A filter using Eq. 44 and then shift the columns of A using E_i. Note that by replacing A with A_i, the necessary conditions $A_i P = I$ and $A_i Q = 0$ are still satisfied similarly to B_i.

Returning to our Chaikin example, we consider the two possible shifted arrangements, and obtain

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} -1.30932 \\ 0.0720339 \\ 0.572034 \\ 0.190678 \end{bmatrix}, \tag{55}$$

for E_{-1} and

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} -0.190678 \\ -0.572034 \\ -0.0720339 \\ 1.30932 \end{bmatrix}, \tag{56}$$

for E_1 with 1.02588 and 1.02588 as the respective (and identical) least squares error. Consequently, E_0 supports the optimal arrangement and Eq. 43 represents the optimal solution for the Chaikin filter.

5.4. *Increasing Bandwidth*

The solution we proposed in the previous section assumes that all filters have the same size k. However, for a given P filter of size k, we can find longer filters for Q and A. Increasing the bandwidth of Q and A allows us to further reduce the reconstruction residual by introducing more degrees of freedom into our over-determined linear system (Eqs. 41).

Let l be an *even* positive integer, indicating the number of extra elements. Then we assume the P filter has $k + l$ elements with l zero elements added to the end of the initial P filter:

$$\begin{bmatrix} p_1 \\ \vdots \\ p_k \\ p_{k+1} \\ \vdots \\ p_{k+l} \end{bmatrix} = \begin{bmatrix} p_1 \\ \vdots \\ p_k \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{57}$$

Using this new P filter, we can calculate the B filter in the same way as before (via Eq. 35):

$$\begin{bmatrix} b_1 \\ \vdots \\ b_l \\ b_{l+1} \\ \vdots \\ b_{k+l} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -p_k \\ \vdots \\ p_1 \end{bmatrix} \tag{58}$$

To calculate Q, we form Eq. 41 using the new B filter. Because the new B filter has l zero elements at the beginning, the last $\frac{l}{2}$ rows of hard constraints are zero and can be removed. Moreover, the first $\frac{l}{2}$ rows of the soft constraints are zero as well. After removing the zero rows, the new over-determined linear system has $2k - 2 + l$ rows and $l + k$ columns, where $k - 1 + \frac{l}{2}$ rows correspond to the hard constraints. Therefore, in comparison to Eq. 41, this new over-determined linear system has $\frac{l}{2}$ more degrees of freedom, which serves to reduce the orthogonality error when solving Eq. 41 .

To demonstrate this process, let us extend the Chaikin filter by adding two more elements and recalculating its corresponding multiresolution masks. Based on Eq. 35 we obtain

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -0.25 \\ 0.75 \\ -0.75 \\ 0.25 \end{bmatrix}, \tag{59}$$

which forms the following over-determined linear system

$$\begin{bmatrix} -0.75 & 0.25 & 0 & 0 & 0 & 0 \\ -0.25 & 0.75 & -0.75 & 0.25 & 0 & 0 \\ 0 & 0 & -0.25 & 0.75 & -0.75 & 0.25 \\ 0 & 0 & 0 & 0 & -0.25 & 0.75 \\ 0.75 & 0.25 & 0 & 0 & 0 & 0 \\ 0.25 & 0.75 & 0.75 & 0.25 & 0 & 0 \\ 0 & 0 & 0.25 & 0.75 & 0.75 & 0.25 \\ 0 & 0 & 0 & 0 & 0.25 & 0.75 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{bmatrix} = \mathbf{e_s}\,, \tag{60}$$

where the optimal solution for Q is

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{bmatrix} = \begin{bmatrix} -0.0857226 \\ -0.257168 \\ 0.03642 \\ 0.795041 \\ -0.619237 \\ -0.206412 \end{bmatrix} \tag{61}$$

with E_0 and 0.34716 as the least squares error. Obviously, the quality is improved in comparison to Eq. 43.

There exists no limitation on extending the bandwidth and, depending on the application, one can utilize a custom filter size. Note that increasing the bandwidth improves reconstruction error but, on the other hand, reduces the efficiency of using the multiresolution masks. Because of this trade-off, it is best to find a Q filter with minimal size such that its corresponding orthogonality error is less than a threshold. To this end, we start with $l = 0$ and calculate the optimal multiresolution masks. Then, if the orthogonality error is less than the given threshold we stop. Otherwise, we increase l by 2 and repeat the procedure until the desired level of error is achieved. Alg. 5.1 describes the procedure of finding A, B, and Q filters, provided an input P filter and a threshold value. The algorithm employs a main loop, which in each of its iterations the bandwidth is extended until the threshold is reached. In each iteration, B mask is calculated first. Then, a least squares system is formed for calculating Q mask corresponding to Eq. 41. Next, in another loop, the best shift is found by solving the system for all possible shift values. At last, A mask is calculated based on the best Q.

Using Alg. 5.1 we have developed multiresolution filters for polynomial, CINPACT, and interpolating CINPACT functions. For example, the resulting Q filter for interpolating CINPACT with $c = 5$ and $\sigma = 4.79$ is $[0.00876249, 0, -0.0473956, 0, -0.779718, 0.36458, 0.221193, 0, -0.0473956, 0, 0.00876249]$. This filter is the result of a shift using E_{-1}. The other filters have been provided in the supplementary materials. The first set of filters are calculated without extension and hence, they have the minimum possible length.The second set provides extended filters for selected polynomial, CINPACT, and interpolating CINPACT subdivision schemes. As shown in Fig. 7, increasing bandwidth decreases the orthogonality error and improves reconstruction. It is also important to note that multiresolution filters of the same size may have different orthogonality errors (see Fig. 7). Of the multiresolution masks we considered, the best masks (i.e.

Algorithm 5.1 This algorithm describes the procedure of calculating optimal banded multiresolution filters provided by a subdivision filter.

```
Require: $\mathbf{p} = [p_1, \ldots, p_k]$, $threshold$                    ▷ $P$ filter
                                                  ▷ Orthogonality error threshold
if $k$ is odd then                               ▷ Ensuring even number of elements
    $p_{k+1} \leftarrow 0$                               ▷ Add a zero element to $P$ filter
    $k \leftarrow k+1$
end if
$Err_{min} \leftarrow \infty$
$l \leftarrow 0$
$shift \leftarrow 0$
while $Err_{min} > threshold$ do
    Let $\mathbf{a}, \mathbf{b}, \mathbf{q}$ be three $k \times 1$ zero vector.
    for $i = 1$ to $k$ do                                  ▷ Calculating $B$ filter
        if $i$ is odd then
            $b_i \leftarrow -p_{k-i+1}$
        else
            $b_i \leftarrow p_{k-i+1}$
        end if
    end for
    Let $C$ and $M$ be two $(k-1) \times k$ zero matrix.
    for $i = 1$ to $k-1$ do
        for $j = \max(1, 2i-k+1)$ to $\min(2i, k)$ do
            $C_{i,j} \leftarrow b_{k-2i+j}$                        ▷ Hard constraints
            $M_{i,j} \leftarrow p_{k-2i+j}$                        ▷ Soft constraints
        end for
    end for
    $C \leftarrow C[1 : k-1-\frac{l}{2}, :]$                    ▷ Removing zero rows
    $M \leftarrow M[1+\frac{l}{2} : k-1, :]$                    ▷ Removing zero rows
    for $i = 1$ to $k-1-\frac{l}{2}$ do                  ▷ For all possible shifts
        Let $\mathbf{t}$ be a $(2k+l-2) \times 1$ zero vector
        $t_i \leftarrow 1$
        $\mathbf{q}_{temp} \leftarrow$ Solve$(C, M, t)$
        $Err_{temp} \leftarrow$ LSQError$(C, M, q_{temp})$
        if $Err_{temp} < Err_{min}$ then
            $Err_{min} \leftarrow Err_{temp}$
            $\mathbf{q} \leftarrow \mathbf{q}_{temp}$
            $shift \leftarrow 2\left(i - \frac{k+l}{2}\right)$
        end if
    end for
    for $i = 1$ to $k$ do                                  ▷ Calculating $A$ filter
        if $i$ is odd then
            $a_i \leftarrow q_{k-i+1}$
        else
            $a_i \leftarrow -q_{k-i+1}$
        end if
    end for
    if $Err_{min} > threshold$ then
        $p_{k+1} \leftarrow 0$
        $p_{k+2} \leftarrow 0$
        $k \leftarrow k+2$
        $l \leftarrow l+2$
    end if
end while
Return $\mathbf{a}, \mathbf{b}, \mathbf{q}, shift$
```

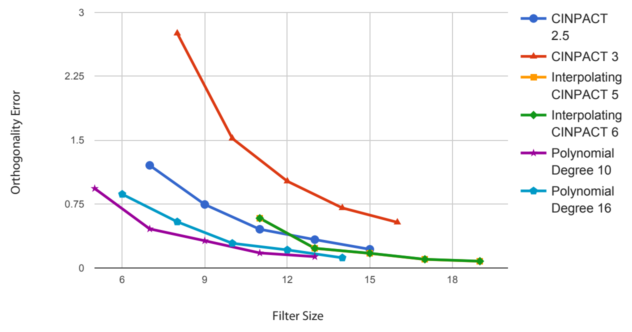


Figure 7: Plot of the orthogonality error against filter size: the orthogonality error is decreased by increasing bandwidth. As depicted, the polynomial multiresolution masks have less orthogonality error compaired to the CINPACT masks.

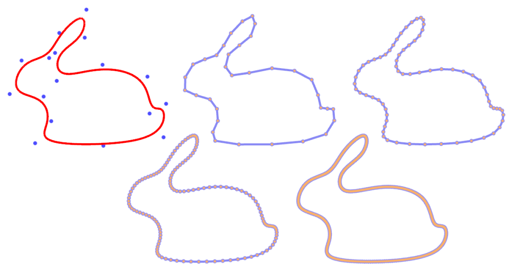


Figure 8: An example of subdivision using CINPACT filters ($c = 3.684$, $\sigma = 17.27$).

those with the smallest orthogonality error) correspond to the polynomial (degree 10) and interpolating CINPACT masks.

6. Results

This section presents example applications of PUPs multiresolution. We have employed several subdivision and multiresolution schemes developed using polynomials and CINPACT-splines to produce different curves and tensor-product examples. We also compare our schemes with previous subdivision and multiresolution schemes and show advantages of PUPs subdivision and multiresolution schemes. All of the employed filters are provided in supplementary materials. The high resolution contours used to generate the following examples were obtained from [30].

The curves in Fig. 8 illustrate the result of applying CINPACT subdivision filters, which converges to the initial PUPs curve. Fig. 9 shows the result of cubic B-Spline subdivision and two example polynomial PUPs subdivision. 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 illustrated in Fig. 9, the sixth degree polynomial produces curves with less energy while both cubic B-Spline and the sixth degree polynomial have the same number of refinement coefficients. More subdivision results and an in-depth discussion are provided in [4].

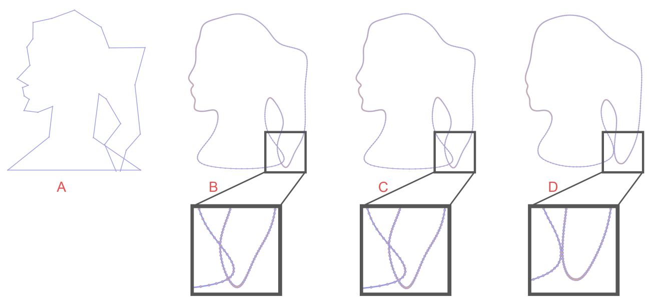


Figure 9: Comparison of cubic B-Spline and polynomial subdivision. (A) The initial control net obtained from [30]. (B-D) The result of four applications of cubic B-Spline (B) degree 10 polynomial (C) and degree 6 (D) subdivision. Wide range of shapes are generated by changing polynomial coefficients and degree without changing bandwidth.

Fig. 10 shows applications of polynomial reverse subdivisions on a sample curve. The number of points (resolution) is halved in each iteration. Fig. 11 shows the applications of reverse interpolating CINPACT subdivision. As interpolating CINPACT subdivisions interpolate the points through subdivision, we expect their reverse subdivision to inverse the process and preserve half of the control points. Although this is not exactly achieved (because of the details), the reverse subdivision of interpolating CINPACT functions approximately maintain half of the control points. Consequently, the filters work well for creating the cliche of an object, as the results mimic the initial high resolution point set.

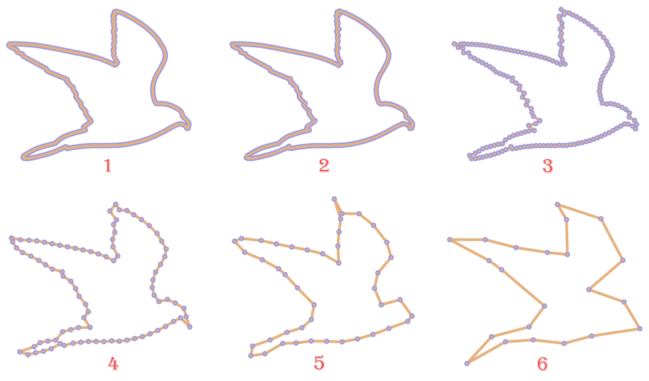


Figure 10: Example application of polynomial reverse subdivision (degree 10).

Fig. 12 shows the difference between minimal and extended multiresolution masks of the CINPACT function with $c = 3, \sigma = 7.27$. As explained, the orthogonality error exponentially decreases by increasing bandwidth of the masks, and consequently the reconstruction error is improved. Row (A) of Fig. 12 shows the result of minimal multiresolution masks whose orthogonality error is 7.58657. The curve has high energy and is exaggerated after a few steps of reverse subdivision. On the other hand, row (B) shows the better results obtained by the extended masks whose orthogonality error is 0.285485. Based on our experiments, the visual quality of the results is decent if the orthogonality error is less than 1.

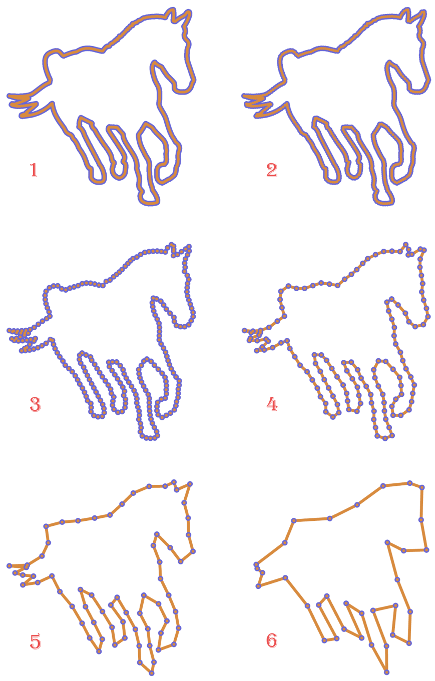


Figure 11: Example application of interpolating CINPACT reverse subdivision ($c = 5, \sigma = 4.79$).

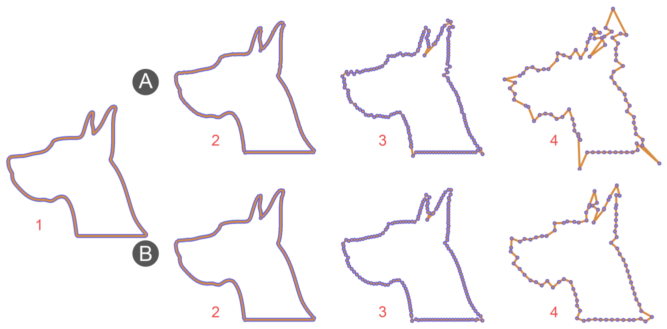


Figure 12: Example application of CINPACT reverse subdivision ($c = 3, \sigma = 7.27$) without extension (A) and with extension (B). As the orthogonality error decreases by extending multiresolution filters, the quality of the results is improved. Based on the provided tables the orthogonality error is 7.58657 and 0.285485 for (A) and (B) respectively.

Fig. 13 and Fig. 14 show additional applications of PUPs multiresolution for macroscopic editing and transferring features. In Fig. 13, the high-resolution data of the horse head is decomposed first. The result represents the overall macroscopic shape of the horse. Next, the low resolution data is modified by moving one of its points. At last, the horse is reconstructed using the details obtained from the decomposition. As illustrated, a designer can easily modify a curve or surface using this technique and produce visually pleasing results.

In Fig. 14, we have used multiresolution masks to transfer the horse features to a simple circle curve. To this end, we decomposed the horse data for several steps and extracted the

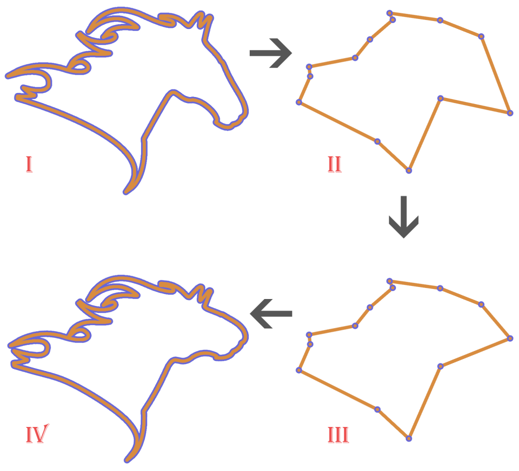


Figure 13: Example application of PUPs multiresolution for editing and modifying curves. (I) The initial high resolution data (obtained from [30]). (II) The result of 8 decomposition steps using interpolating CINPACT ($c = 5$, $\sigma = 4.79$). (III) The low resolution curve is modified by moving one of the points. (IV) The result of reconstruction using the modified low resolution data. The horse head is different after modification.

details. Then, the details were added to the circle through subdivision. As illustrated, the wavy and coiled back of the horse is moved to the circle.

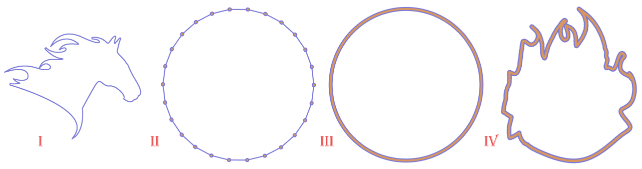


Figure 14: Example application of PUPs multiresolution for transferring features from a source to a destination curve. (I) The initial high resolution data (obtained from [30]) that is used for extracting features. (II) The initial points. (III) The result of subdivision without using features. (IV) The result of subdividing using the horse features.

Application of PUPs multiresolution is extendable to tensor-product meshes. Fig. 15 shows the initial high resolution terrain of Mount Shasta. The resolution of the mesh is reduced using interpolating CINPACT filter ($c = 5, \sigma = 4.79$). As illustrated in Fig. 15, the key features of the terrain (such as Mount Shasta) are preserved while the resolution is reduced. Such results support application of PUPs multiresolution in level-of-detail visualization.

We can also use the developed filters to reduce the resolution of images and volumes. These types of datasets have a regular structure enabling us to apply our filters to them. In particular, we apply our filters to images one time row-wise and one time column-wise in each iteration to reduce horizontal and vertical resolutions. Fig. 16 show applications of PUPs multiresolution masks in image compression. We applied extended multiresolution filters of interpolating CINPACT ($c = 5, \sigma = 4.79$) to the Lena test image [34]. For the sake of comparison, we have also applied CDF 9/7 [1] wavelets that are used for jpeg–2000 lossy compression[35]. For boundaries, we used half-sample symmetry [36] and we calculated the PSNR of each image's luminosity as a quality criterion. Visually comparing the results shows that interpolating CINPACT masks produce high quality results better than or almost the same as CDF wavelets. This is further supported by higher PSNR values of interpolating CINPACT masks.

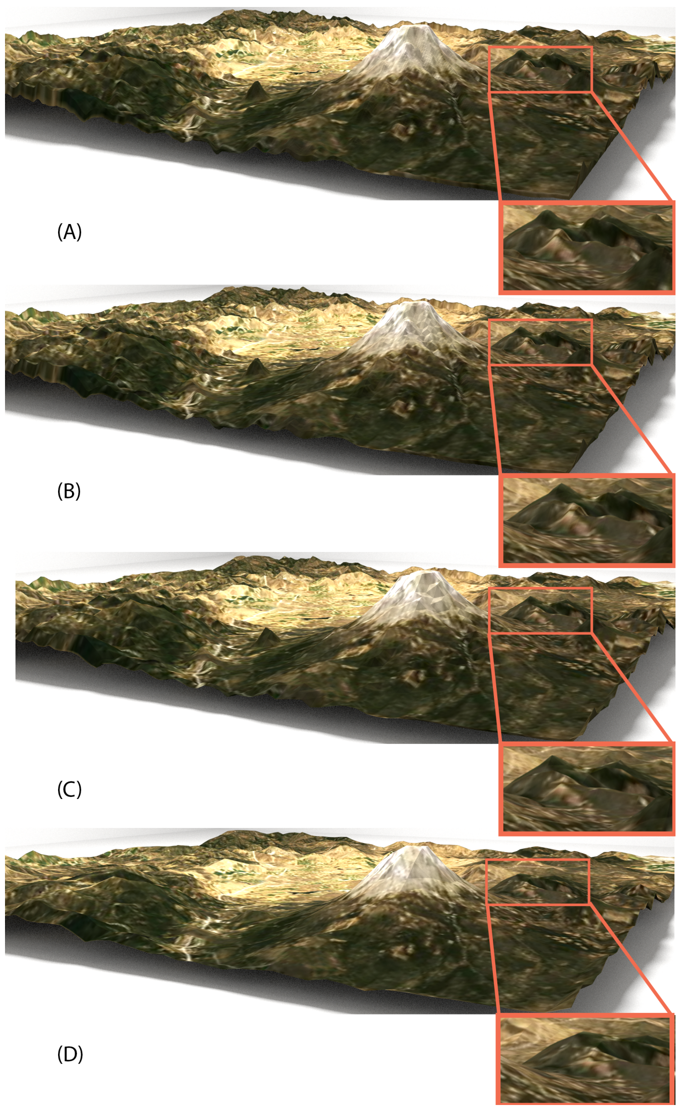


Figure 15: Application of PUPs multiresolution in terrain simplification: (A) the initial high resolution terrain of Mount Shasta (with 262144 vertices). The elevation data is obtained from [33]. (B-D) the result of terrain simplification using interpolating CINPACT filters ($c = 5, \sigma = 4.79$) after one, two, and three decomposition steps respectively. The number of vertices are 56536, 16384, and 4096 respectively.

7. Conclusion

In this paper, we have presented a framework to systematically construct subdivision and multiresolution schemes for Partition of Unity Parametrics. 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. Moreover, we can control the amount of reconstruction error by extending the bandwidth of multiresolution masks.

Regarding future works, more challenges are raised in the context of PUPs subdivision. At present, our current subdivision 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 error 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 or using phantom points, it is possible to derive special filters for boundaries as well.

Furthermore, it is important that we analyze the smoothness of our subdivision schemes in the presence of least squares error. One approach is to prove that the limit function converges *uniformly*, which is a stronger condition than point-wise convergence and depends on the weight functions. Uniform convergence preserves smoothness; and for example, one can show CINPACT subdivision schemes have C^∞ smoothness if their subdivision converges uniformly. Another approach to smoothness analysis is that of analyzing eigen-values of the local refinement matrices [16, 9]. However, existing techniques for analyzing such matrices are not directly applicable to our PUPs refinement matrices, because rows of these matrices may not sum to one (due to the passage through Grassman space).

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. Currently, our subdivision schemes support tensor product surfaces. We

[1]Cohen-Daubechies-Feauveau

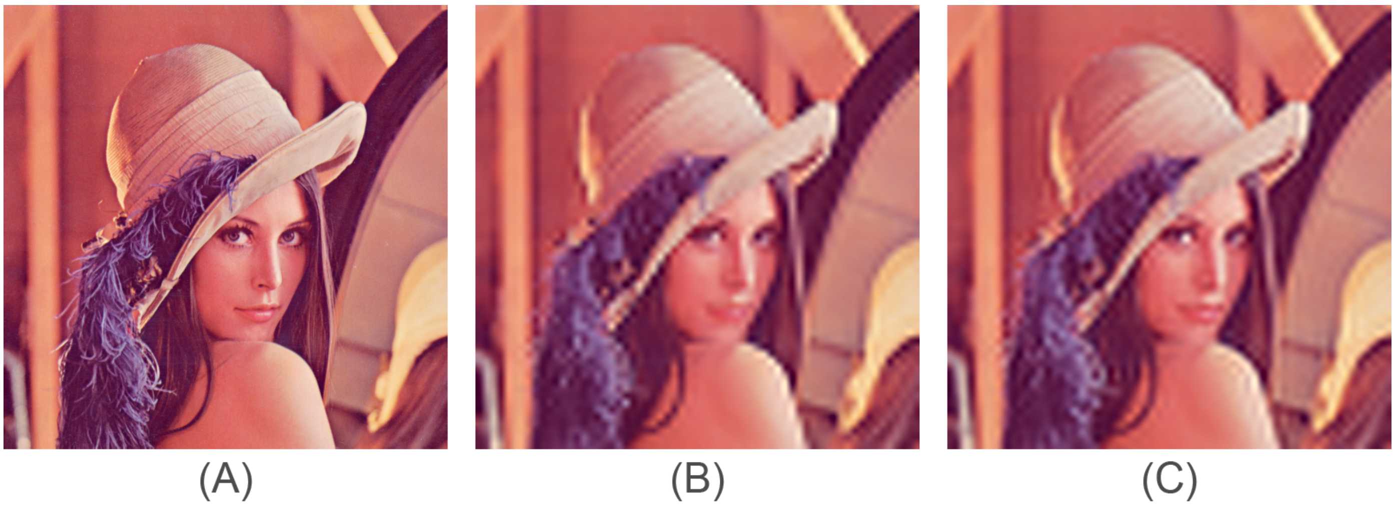


Figure 16: (A) The original Lena image obtained from [34]. (B) and (C) The results of compressing the Lena image using CDF9/7 and extended ($l = 32$) interpolating CINPACT filters ($c = 5, \sigma = 4.79$) respectively. Three decomposition steps are used. The PSNR value of luminosity is 25.3941 and 25.5484 respectively.

can extend their applications to other classes of meshes if we provide subdivision rules for extra-ordinary vertices which do not follow a regular tensor connectivity.

Finally, exploring extensive applications of PUPs multiresolution is left for future works. For example, we can use PUPs multiresolution masks for texture synthesis via utilizing the details. Furthermore, it is also worth exploring other kinds of soft constraints in place of the orthogonality constraints to produce multiresolution schemes supporting other properties.

Acknowledgement

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

[1] Runions A, Samavati F. Partition of Unity Parametrics: A Framework for Meta-modeling. The Visual Computer 2011;27(6-8):495–505.
[2] Runions A, Samavati F. Cinpact-splines: A class of C^∞ curves with compact support. In: Curves and Surfaces; vol. 9213 of *Lecture Notes in Computer Science*. Springer International Publishing; 2015, p. 384–98.
[3] Caron J, Mould D. Texture synthesis using label assignment over a graph. Computers & Graphics 2014;39:24–36.
[4] Moltaji A, Runions A, Samavati F. A Subdivision Framework for Partition of Unity Parametrics. In: Proceedings of the 42nd Graphics Interface Conference. GI '16; Canadian Information Processing Society; 2016, p. 11.
[5] Akram B, Alim U, Samavati F. CINAPACT-splines: A family of infinitely smooth, accurate and compactly supported splines. In: Proceedings of the International Symposium on Visual Computing. 2015, p. to appear.
[6] Cashman TJ. Beyond Catmull-Clark: A survey of advances in subdivision surface methods. Computer Graphics Forum 2012;31(1):42–61.
[7] Dyn N, Levin D. Subdivision schemes in geometric modelling. Acta Numerica 2002;11:73–144.
[8] Marinov M, Dyn N, Levin D. Geometrically controlled 4-point interpolatory schemes. In: Dodgson N, Floater M, Sabin M, editors. Advances in Multiresolution for Geometric Modelling. Mathematics and Visualization; Springer Berlin Heidelberg; 2005, p. 301–15.
[9] Zorin D, Schroder P, DeRose T, Kobbelt L, Levin A, Sweldens W. Subdivision for modeling and animations. In: ACM SIGGRAPH Courses(2000). 2000, p. 1–194.
[10] Cashman TJ, Dodgson NA, Sabin MA. Non-uniform B-spline subdivision using refine and smooth. In: Proceedings of the 12th IMA International Conference on Mathematics of Surfaces XII. Springer-Verlag; 2007, p. 121–37.
[11] Cashman TJ, Dodgson NA, Sabin MA. Selective knot insertion for symmetric, non-uniform refine and smooth B-spline subdivision. Computer Aided Geometric Design 2009;26(4):472–9.
[12] Cashman TJ, Dodgson NA, Sabin MA. A symmetric, non-uniform, refine and smooth subdivision algorithm for general degree B-splines. Computer Aided Geometric Design 2009;26(1):94–104.
[13] Cashman TJ, Augsdörfer UH, Dodgson NA, Sabin MA. NURBS with extraordinary points: High-degree, non-uniform, rational subdivision schemes. ACM Transactions on Graphics 2009;28(3):1–9.
[14] Cashman TJ, Hormann K, Reif U. Generalized Lane-Riesenfeld algorithms. Computer Aided Geometric Design 2013;30(4):398 – 409.
[15] Schaefer S, Vouga E, Goldman R. Nonlinear subdivision through nonlinear averaging. Computer Aided Geometric Design 2008;25(3):162 –80.
[16] Micchelli CA, Prautzsch H. Uniform refinement of curves. Linear Algebra and its Applications 1989;114115(0):841 –70.
[17] Finkelstein A, Salesin DH. Multiresolution curves. In: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '94; 1994, p. 261–8.
[18] Samavati FF, Bartels RH. Multiresolution curve and surface representation: Reversing subdivision rules by least-squares data fitting. Computer Graphics Forum 1999;18(2):97–119.
[19] Bartels RH, Samavati FF. Reversing subdivision rules: local linear conditions and observations on inner products. Journal of Computational and Applied Mathematics 2000;119(1–2):29–67.
[20] Bartels RH, Samavati FF. Multiresolutions numerically from subdivisions. Computers & Graphics 2011;35(2):185–97.
[21] Sweldens W. The lifting scheme: A new philosophy in biorthogonal wavelet constructions. In: in Wavelet Applications in Signal and Image Processing III. 1995, p. 68–79.
[22] Sweldens W. The lifting scheme: A custom-design construction of biorthogonal wavelets. Applied and Computational Harmonic Analysis 1996;3(2):186–200.
[23] Sadeghi J, Samavati FF. Smooth reverse subdivision. Computers & Graphics 2009;33(3):217–25.
[24] Sadeghi J, Samavati FF. Smooth reverse loop and catmull-clark subdivision. Graphical Models 2011;73(5):202–17.
[25] Goldman R. 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.

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

[27] Farin GE. NURB Curves and Surfaces : From Projective Geometry to Practical Use. A.K. Peters; 1995.

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

[29] Strang G, Zhou DX. The limits of refinable functions. Transactions of the American Mathematical Society 2001;353(5):1971–84.

[30] Freepik.com . Free graphics resources. 2016. URL: `http://www.freepik.com`; online; Accessed: 2016-03-03.

[31] Golub GH, Van Loan CF. Matrix Computations (3rd Ed.). Johns Hopkins University Press; 1996.

[32] Samavati FF, Bartels RH. Local filters of b-spline wavelets. In: Proceedings of the International Workshop on Biometric Technologies (BT'04). 2004, p. 105–10.

[33] USGS . Us geological survey. 2016. URL: `https://www.usgs.gov/`; online; Accessed: 2016-04-03.

[34] of Southern California U. The USC–SIPI image database. 2016. URL: `http://sipi.usc.edu/database/database.php`; online; Accessed: 2016-03-03.

[35] Salomon D. Data Compression: The Complete Reference. Springer-Verlag New York, Inc.; 2006.

[36] Kiya H, Nishikawa K, Iwahashi M. A development of symmetric extension method for subband image coding. IEEE Transactions on Image Processing 1994;3(1):78–81.