

BETTER GLOBAL POLYNOMIAL APPROXIMATION FOR IMAGE RECTIFICATION*

Christopher O. Ward

Department of Mathematics and Computer Science

University of West Indies (St. Augustine),

Trinidad & Tobago

West Indies

christopher.ward@sta.uwi.edu

December 2, 2013

Abstract

When using images to locate objects, there is the problem of correcting for distortion and misalignment in the images. An elegant way of solving this problem is to generate an error correcting function that maps points in an image to their corrected locations. We generate such a function by fitting a polynomial to a set of sample points. The objective is to identify a polynomial that passes “sufficiently close” to these points with “good” approximation of intermediate points. In the past, it has been difficult to achieve good global polynomial approximation using only sample points. We report on the development of a global polynomial approximation algorithm for solving this problem.

Key Words: Polynomial approximation, interpolation, image rectification.

1 Introduction

The problem that is addressed here occurred in the context of the development of a simple, low-cost robotic exhibit for demonstrating the concept of intelligent robotics to the general public. Intelligent robotics deals with the use of sensors to enhance a robots performance in an uncertain environment. For the exhibit, we implemented a visually-guided pick-and-place robot. The robot uses an image to determine the location of objects placed arbitrarily on a flat surface and demonstrates success in locating the objects by manipulating them. The exhibit consists of a robotic arm that is fixed in front of a flat work surface. Two pedestals are placed anywhere within reach of the

*The original Paper entitled Better Global Polynomial Approximation for Image Rectification, was published in the International Journal of Modelling and Simulation, Vol. 28, No. 3, 2008, pp 299-308.

arm. One pedestal is blue and the other is green. A blue ball is placed on the blue pedestal and the robot must pick up the ball and place it on the green pedestal. An ordinary webcam is placed in a frame above the work surface and is used to determine the location of the pedestals so that the arm can be guided accordingly. Fig. 1(a) shows the camera's view of the workspace. Once the exhibit has been set up, camera, robot and workspace are all fixed relative to one another.

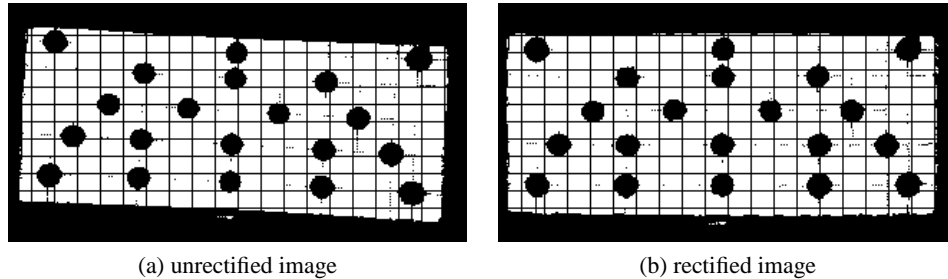


Figure 1: Rectification of an image of a test pattern placed over the robot's workspace taken from a severely misaligned camera. A grid has been superimposed on the images to aid in comparing the horizontal and vertical alignment of image features. Although the original images are colour coded, the images shown are monochrome. If the upper left dot and the upper right dot are ignored, the rest of the dots in (b) sample the area of the workspace that is within reach of the robot.

There is therefore the problem of determining the location of the colour-coded pedestals based on their image. This problem is compounded by the fact that the webcam produces significant image distortion, and by the fact that the position and orientation of the camera relative to the work surface may not be exactly the same every time the exhibit is set up. A similar problem occurs when any automated mechanism is taken apart for maintenance. The mechanism usually must be recalibrated when it is reassembled. In this sense, we are addressing the problem of easy recalibration of the vision component of our robotic exhibit.

In processing the image, there is a need to rectify significant image distortion caused by the optical properties of the camera and by errors in positioning the camera during set up of the exhibit. We solve this problem by determining a mapping from the pixel locations of image points to the physical locations of corresponding source points. The mapping has to be determined empirically in order to recalibrate the vision system each time the apparatus is set up. The pixel positions of the images of key points in a test pattern are mapped to the known locations of these key points. This partial mapping is then used to approximate a mapping for the entire image.

In [1], Brown surveyed and classified several established methods for determining the mapping from pixel position to source location for a camera. Using Brown's classification, the errors caused by distortion and misalignment are static in the sense that they do not change from image to image taken with the same camera in the same position. Static distortions can be rectified in a one-time-only setup process via calibration techniques. Our method may be regarded as a calibration technique.

Brown classifies the distortion due to camera properties as internal and the mis-

alignment as external. Our approach does not require the use of any sort of model of the characteristics of the camera or the geometry of how an image is captured. We treat the mechanism by which source points are captured as image points as a “black box”, the inner workings of which is not modelled. We therefore handle these internal and external distortions without having to distinguish between them.

Furthermore, we are concerned with geometric distortions as distinct from photometric distortions. The effect of photometric distortions are minimised by using primary colours (red, green, blue) for the key features in the scene and either black or white for the other features. Features can therefore be identified by colour without regard for intensity values. Colour distortions are handled by filtering out colours below a fixed saturation value using a median filter to remove speckles and discretising the remaining colours to fully saturated red, green or blue [2]. Visual artifacts are then located by looking for regions within the image with the appropriate colour.

The problem is thus reduced to one of finding an adequate approximation of a total mapping from pixel positions to object locations using a set of sample points. Three popular approaches to solving this problem are: the use of an artificial neural network; the use of a piecewise interpolation technique to fit curves of a chosen form in between the sample points; the use of polynomial interpolation to fit a polynomial to the sample points.

Artificial neural networks are popular among engineers for their ability to generalise from sample data. We find it difficult to argue on practical grounds against this approach. However, for us, the main difficulty with this method is its inability to yield a representation of the approximating function that is independent from that of the neural network itself. However, recent advances in the use of algebraic training of neural networks to produce closed form analytic solutions are addressing this problem [3, 4].

Piecewise interpolation is popular among the data visualisation community for its ability to handle localised distortions [1]. Given sample points on a planar surface, the domain must be divided into polygonal segments with sample domain values as vertices. A smooth surface segment is then fitted over each polygon such that the sample points associated with the vertices lie on the surface, and the edges associated with adjacent segments meet in a prescribed way. There is usually more than one way to segment the domain space, and different segmentations may yield very different interpolations.

Global (as opposed to piecewise) polynomial interpolation is a theoretical possibility that has proven elusive in practice. The Stone-Weierstrass Theorem [5, 6] establishes the existence of a polynomial approximation for every real-valued continuous function defined on a closed interval. Therefore, as long as the sample data does not admit the existence of discontinuities, we should be able to fit a polynomial to the points with an arbitrary degree of precision. The problem (exemplified by Runge’s function [5, 7]) is that successive interpolations do not necessarily converge as more data points are added to the set of sample points (c.f. Fabers Theorem [5, 7]). We will refer to this problem later as the *convergence problem*.

Our solution is similar to global polynomial interpolation. However, the fact that we are dealing with noisy data allows us to relax the requirement for an exact fit and to focus instead on the suggested shape of the function. It is because of this relaxed focus on an exact fit that we refer to our problem as an approximation problem rather than an

interpolation problem. By focusing on the suggested shape of the function, we directly address the convergence problem.

In this paper, we present the generic function approximation algorithm and use image rectification as an example of its use. In section 2, we develop the theory behind the approximation algorithm in the univariate case. Section 3 discusses the bivariate version of the approximation algorithm that is used to solve our robot vision problem. Section 4 summarizes our findings.

2 Polynomial approximation of functions

In the univariate case, the polynomial approximation problem can be defined as follows:

Definition 1 (The univariate polynomial approximation problem) *Find a polynomial $P(x)$ that fits a set of m sample points (x_i, y_i) ; where $i = 1, 2, \dots, m$ and the x_i are distinct; such that:*

$$\max_{i=1}^m |y_i - P(x_i)| \leq \epsilon, \text{ for } \epsilon \geq 0$$

That is, we are trying to identify a polynomial, $P(x)$, that fits a finite set of points to a degree of precision determined by ϵ . The condition that the x_i must be distinct ensures that there are no discontinuities in the target function. Otherwise, a solution is not assured.

The Weierstrass approximation theorem [3, 5, 7] assures us that this is a solvable problem for $\epsilon > 0$:

Theorem 1 (Weierstrass) *If F is any continuous function on the finite closed interval $[a, b]$, then for every $\epsilon > 0$ there exists a polynomial $P_n(x)$ of degree n (where n depends on ϵ) such that:*

$$\max_{x \in [a, b]} |F(x) - P_n(x)| < \epsilon$$

Our finite set of points is not generated from a known function that we are trying to approximate, so we will be looking instead for an approximation that generates intermediate points that are “good” in some generic sense. We require the algorithm to yield an approximating polynomial that does not “curve” unnecessarily between sample points. In our algorithm, we try to conservatively fit a shape to the sample points using a linear combination of Chebyshev polynomials. A preference for fitting lower order Chebyshev polynomials reflects a preference for simple shapes. This emphasis on simple shapes addresses the convergence problem by only adding detail when it results in a better fit.

Section 2.1 presents the basis for our algorithm in its purest form and explains why it is inadequate. We refer to this algorithm as the Cartesian vector based (CVB) interpolation. This algorithm is of theoretical interest only, since it produces results similar to Lagrange interpolation, with the same convergence problems illustrated by Runge and covered by Fabers theorem [5,7]. It sets the scene for presentation of the modified version of the algorithm. In section 2.4, we present the modified algorithm that gives better results on the same data in terms of capturing the suggested shape of the curve. We refer to this second algorithm as the CVB approximation algorithm.

2.1 Casting the problem in Cartesian vector space

We address the approximation problem by finding a linear combination of the first n Chebyshev polynomials of the first kind that exhibits the desired properties of $P(x)$. Thus:

$$P(x) = \sum_{i=0}^{n-1} a_i T_i(x) \quad (1)$$

where $T_i(x)$ is defined for $x \in [-1, 1]$ as

- $T_0(x) = 1$,
- $T_1(x) = x$,
- $T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x)$, for $i > 1$.

The problem is to find a set of coefficients, a_i , $i = 0, \dots, n-1$ that establish a fit. Although other sets of basis polynomials exist, Chebyshev polynomials reputedly yield good interpolation results. In what follows, we will assume without loss of generality that x falls within the interval $[-1, 1]$.

The fact that we are using a finite set of sample points allows us to reformulate the problem as one involving vectors in a Cartesian m -space; where m is the number of sample points. For this purpose the problem is restated as follows:

Definition 2 (The univariate Cartesian vector based (CVB) polynomial approximation problem)

Given

- *a set of points, $(x_i, y_i) \in \mathbb{R}^2$, $i = 1, \dots, m$;*
- *a set of m -dimensional Cartesian vectors, τ_j , $j = 0, \dots, n-1$, such that the i 'th component of τ_j is equal to $T_j(x_i)$, for $i = 1, \dots, m$;*
- *a Cartesian vector, γ such that the i 'th component of γ is equal to y_i , for $i = 1, \dots, m$;*
- *a real value ϵ ($\epsilon > 0$);*

find values for a set of scalar quantities, a_j , such that:

- $\rho = \sum_{j=0}^{n-1} a_j \tau_j$;
- $\delta = \gamma - \rho$;
- $|\delta^i| \leq \epsilon$ for each component, δ^i , of δ ($i = 1, \dots, m$).

For example, the representation for the expression

$$x^2 + 2x + 1$$

as a linear combination of Chebyshev polynomials is:

$$1.5T_0(x) + 2T_1(x) + 0.5T_2(x)$$

Table 1: The CVB representation of a curve fit.

i	x_i	y_i	$T_0(x_i)$	$T_1(x_i)$	$T_2(x_i)$	$a_0 T_0(x_i)$	$a_1 T_1(x_i)$	$a_2 T_2(x_i)$	$P(x_i)$
1	-1.0	0.0	1.0	-1.0	1.0	1.5	-2.0	0.5	0.0
2	0.0	1.0	1.0	0.0	-1.0	1.5	0.0	-0.5	1.0
3	1.0	4.0	1.0	1.0	1.0	1.5	2.0	0.5	4.0

Table 1 shows the situation for this function for three evenly spaced sample points. In this example, each Chebyshev polynomial yields a 3-dimensional Cartesian vector. A linear combination of these vectors yields a solution. We will refer to the vectors generated by the polynomial terms as *term vectors*.

If the term vectors τ_j are orthogonal, they may be taken as basis vectors and solution values for the coefficients, a_j , may be directly obtained from the projection of γ on each τ_j , respectively.

However, orthogonality does not hold in general, as illustrated by the set of points in Table 1. An obvious solution to this dilemma is to first identify an orthogonal set of vectors corresponding to the term vectors and use this set to determine a solution. We developed an algorithm to do just this. We refer to it as the *CVB interpolation*.

2.2 The CVB interpolation

This algorithm computes an orthogonal set of vectors, $\{\mathbf{o}_0, \mathbf{o}_1, \dots, \mathbf{o}_{m-1}\}$, from the first m linearly independent term vectors, derives the projection of γ on each of these vectors and modifies the coefficients of the term vectors accordingly.

The orthogonalisation of the term vectors is based on the following theorem on which the well-known *Gram-Schmidt orthogonalisation process* is based:

Theorem 2 *Let $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\}$ be an orthonormal set of vectors in a vector space, V . Then for any vector $\mathbf{w} \in V$, the vector:*

$$\mathbf{o} = \mathbf{w} - \sum_{i=1}^n (\mathbf{w} \cdot \mathbf{v}_i) \mathbf{v}_i$$

is orthogonal to each \mathbf{v}_i , $i = 1, \dots, n$.

This theorem can easily be generalised to apply to orthogonal (as distinct from *orthonormal*) sets as follows:

Theorem 3 *Let $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\}$ be an orthogonal set of vectors in a vector space, V . Then for any vector $\mathbf{w} \in V$, the vector:*

$$\mathbf{o} = \mathbf{w} - \sum_{i=1}^n \frac{(\mathbf{w} \cdot \mathbf{v}_i)}{(\mathbf{v}_i \cdot \mathbf{v}_i)} \mathbf{v}_i$$

is orthogonal to each \mathbf{v}_i , $i = 1, \dots, n$.

The additional factor in the summation normalises each \mathbf{v}_i .

In our algorithm, we identify an orthogonal set of vectors, $\{\mathbf{o}_0, \mathbf{o}_1, \dots, \mathbf{o}_{n-1}\}$ and a set of scalars $p_{j,k}$, such that:

$$\mathbf{o}_0 = \tau_0 \quad (2)$$

$$p_{j,k} = \frac{(\tau_j \cdot \mathbf{o}_k)}{(\mathbf{o}_k \cdot \mathbf{o}_k)} \quad (3)$$

$$\mathbf{o}_j = \tau_j - \sum_{k=0}^{j-1} p_{j,k} \mathbf{o}_k, \text{ for } j = 1, \dots, n-1 \quad (4)$$

We refer to each \mathbf{o}_j as an *orthogonal component* of the corresponding τ_j , for $j = 0, \dots, n-1$.

For the CVB interpolation, we need to express \mathbf{o}_j as a function of τ_k . Using (4) we write:

$$\begin{aligned} \mathbf{o}_j &= \sum_{k=0}^j q_{j,k} \tau_k \\ &= \tau_j - \sum_{k=0}^{j-1} p_{j,k} \mathbf{o}_k \\ &= \tau_j - \sum_{k=0}^{j-1} p_{j,k} \sum_{l=0}^k q_{k,l} \tau_l \\ &= \tau_j - \sum_{l=0}^{j-1} \tau_l \sum_{k=l}^{j-1} p_{j,k} q_{k,l} \\ &= \tau_j - \sum_{k=0}^{j-1} \tau_k \sum_{l=k}^{j-1} p_{j,l} q_{l,k} \end{aligned} \quad (5)$$

From (5) we get:

$$q_{j,j} = 1 \quad (6)$$

$$q_{j,k} = - \sum_{l=k}^{j-1} p_{j,l} q_{l,k} \quad (7)$$

Fig. 2 shows the CVB interpolation in pseudocode. Table 2 shows a trace of the CVB interpolation on the problem depicted in Table 1.

2.3 Difficulties with polynomial interpolation

The following types of data serve to illustrate the difficulties encountered with global polynomial interpolation [8]:

- “Humped and flat data” that suggest a flat curve in some regions and not in others (Fig. 3).

```

1. Set all the coefficients,  $a_j$ , to zero.
2. Compute the orthogonal components,  $o_j$  and the scalars  $q_{j,k}$ .
3. For  $j = 0, \dots, n-1$  do
    (a) Compute the error vector,  $\delta$ .
    (b) Set  $O_{inc}$  to  $\frac{o_j \cdot \delta}{o_j \cdot o_j}$ 
    (c) For  $k = 0, \dots, j$  do
        Set  $a_k$  to  $a_k + O_{inc} \cdot q_{j,k}$ .
    Endfor
Endfor
4. Return the set of coefficients,  $a_j$ .

```

Figure 2: The CVB interpolation (see section 2.2). Computational details have been omitted that deal with avoidance of representational and computational error.

Table 2: A trace of the CVB interpolation on the problem depicted in Table 1.

a_0	a_1	a_2	$\ \delta\ $	$P(x_1)$	$P(x_2)$	$P(x_3)$
1.666666667	0.0	0.0	2.943920289	1.666666667	1.666666667	1.666666667
1.666666667	2.0	0.0	0.816496581	-0.333333333	1.666666667	3.666666667
1.5	2.0	0.5	0.0	0.0	1.0	4.0

- “Noisy straight line data” with y values given at unevenly spaced x values (Fig. 4).
- Data from functions that exhibit the convergence problem, the classical example of which is the Runge function (Fig. 5). Faber’s theorem establishes that the Runge function is not the only function that exhibits this problem.

These types of data all exhibit a common problem: there is significant deviation of the fitted polynomial from the shape of the curve suggested by the data points. With regard to the convergence problem, progressively adding more data points does not lead to progressively better agreement between the fitted polynomial and the suggested shape of the curve (c.f. Faber’s Theorem). The problem of large deviations from ground truth at the intermediate points is usually addressed in one of two ways: either additional information is supplied as data, or additional constraints are imposed as part of the method of solution.

Hermite interpolation is an attempt to improve the quality of the interpolation by using additional information (first derivatives at the data points), but suffers from the fact that the additional information is local to the data points and so does not sufficiently constrain what happens at the intermediate points. What seems to be needed are global constraints that affect the quality of the solution over the entire interval.

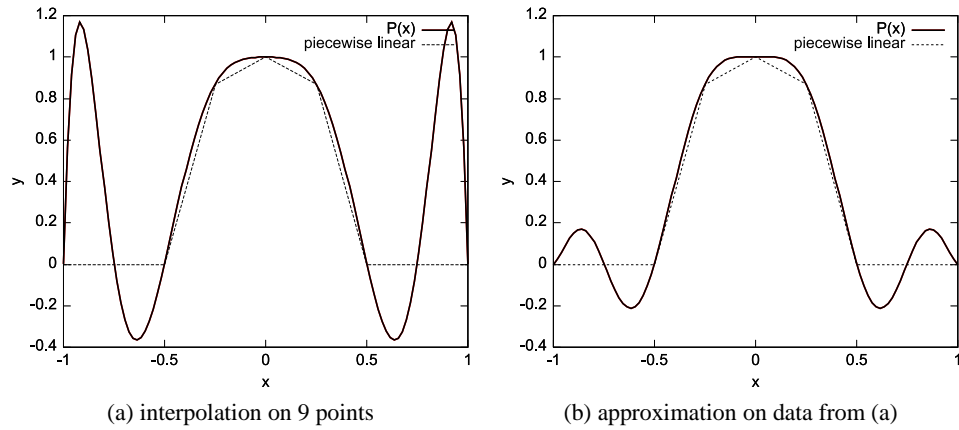


Figure 3: Interpolation and approximation of data that is flat in some places and humped in others.

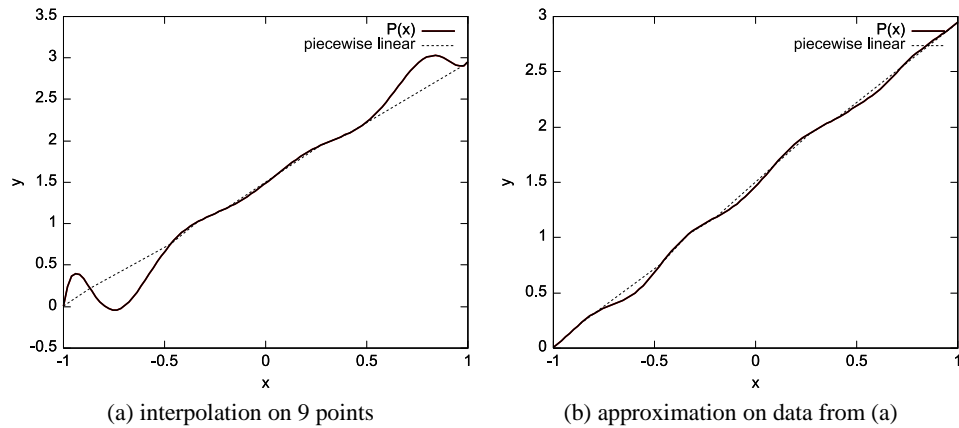


Figure 4: Interpolation and approximation of noisy straight line data with y values given at unevenly spaced x values.

The difficulty with applying global constraints in interpolation is the requirement that a perfect fit be achieved at the sample points. This requirement gives the sample points a special status relative to the intermediate points. The fact that a constraint must respect this special status would render it less than global, as the constraint would have to behave differently in the vicinity of the sample points. If a perfect fit is not required, then global *approximation* may be used. Theory states that polynomial approximations exist that follow closely any curve that can be described by a continuous real-valued function on a closed interval. Global constraints can be more easily applied to selecting the best approximation than to finding a perfect fit.

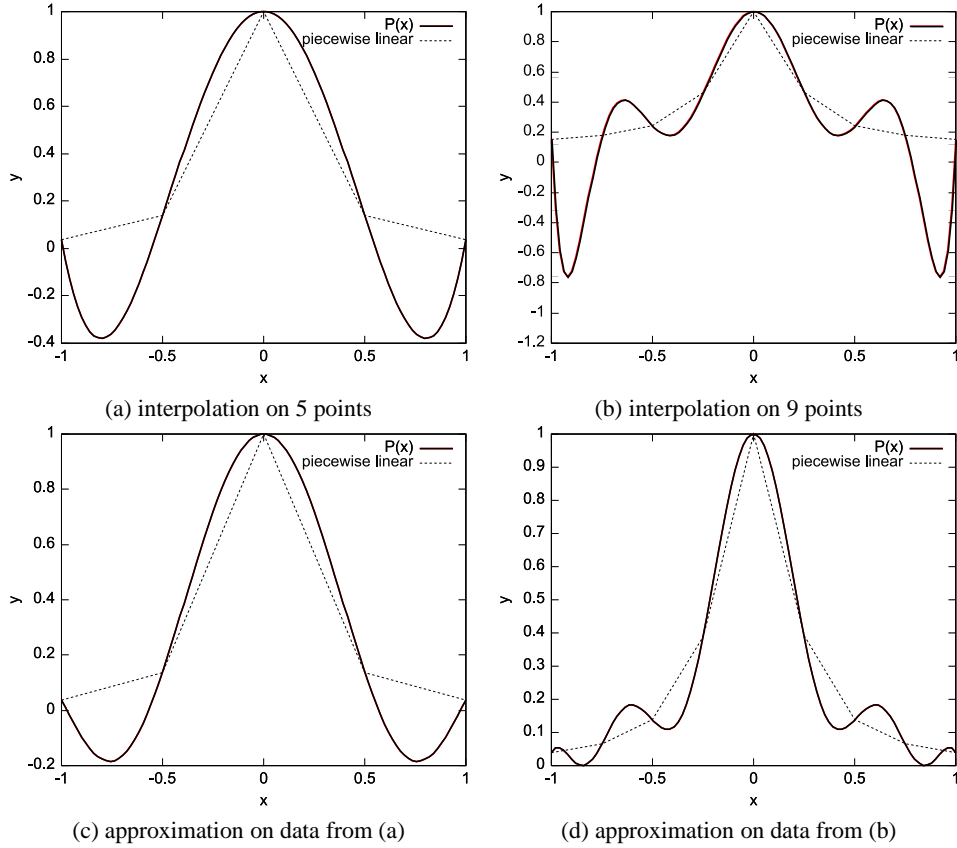


Figure 5: Performance on the convergence problem. Data sets were taken from the Runge function. Comparison of (a) with (b) illustrates that interpolating with more data points does not always give better results. Comparison of (c) and (d) illustrate convergence on the same data using the CVB approximation algorithm presented in section 2.4.

2.4 The CVB approximating algorithm

In this section, we present our *CVB approximating algorithm*. This algorithm gives better convergence than the CVB interpolation. This improvement was achieved by exploiting the observation that the shape of a function that fits the error vector, δ (see Definition 2), is captured in part by the direction of δ , together with the fact that the direction of a Cartesian vector is invariant under scalar multiplication. We define the shape of a function as follows:

Definition 3 (The shape of a function) Consider two continuous real-valued functions, F_1 and F_2 , defined on the same closed interval, I . F_1 and F_2 are of the same shape if there exists a finite non-zero real scale factor, s , such that $F_1 = sF_2$.

Thus, a particular shape is denoted by the set of all the functions that are of that shape. We will only explore in this paper the properties of this concept of shape that are

relevant to our exposition. In the first instance, we will confine our discussion to real-valued functions defined on the interval $I = [-1, 1]$.

Definition 4 (Value ratios) *If two functions, F_1 and F_2 , are of the same shape, then for any two distinct values, $x_1, x_2 \in I$, such that $F_1(x_2) \neq 0$ (and, by extension, $F_2(x_2) \neq 0$), it is the case that $\frac{F_1(x_1)}{F_1(x_2)} = \frac{F_2(x_1)}{F_2(x_2)}$. That is, the ratio of $F_1(x_1)$ to $F_1(x_2)$ is the same as the ratio of $F_2(x_1)$ to $F_2(x_2)$.*

It is this invariance of value ratios across functions of the same shape that defines our concept of shape.

Now consider a Chebyshev polynomial, T , and its associated term vector, τ , defined on a set of sample points. The ratios of the components of τ to one another are value ratios of T . But these ratios also define the direction of the Cartesian vector, τ . So that the shape of T determines the direction of τ . However, since the finite dimensional term vector τ cannot capture the infinity of value ratios that specify the shape of T , the direction of τ only partly captures the shape of T .

If we wish to compare the shape of a Chebyshev polynomial, T , with the shape suggested by δ , we can compare the direction of the corresponding term vector, τ , with the direction of δ . In particular, we look at the projection of δ on τ and express the projection as a scalar multiple of τ . The scalar factor thus identified is used to adjust the coefficient of T in equation (1).

The main difference between the CVB approximating algorithm and the CVB interpolation is an emphasis on fitting the shape of the term vectors as distinct from an orthogonal component thereof. However, since term vectors are not orthogonal in general, more than one approximation may be possible. As a means of selecting a preferred solution, we adopt an heuristic based on fitting lower order terms first.

Like the CVB interpolation, the CVB approximation algorithm works by finding projections of the error vector on a vector space identified by a finite number of term vectors. It gives preference to using the term vectors associated with lower order Chebyshev terms. The process is as follows (see Fig. 6 for the algorithm in pseudocode):

- The first projection to be eliminated is the projection of δ on τ_0 . Whenever the projection of δ on a term vector, τ , has been eliminated, τ will be referred to as having been *visited*.
- If τ_j has been visited, then τ_{j-1} must be re-visited. This rule is applied recursively to revisits until τ_0 is revisited.
- After τ_j has been visited and all consequent revisits have taken place, then τ_{j+1} is visited.
- The process terminates when the required degree of accuracy of fit has been obtained.

Revisits are necessary in order to maintain the fit of lower order terms. More preferred terms are revisited after lesser preferred ones in order to minimise the effect of revisits on their fit. Thus the most preferred term is revisited last.

```

1. Set all the coefficients,  $a_j$ , to zero.
2. Compute the error vector,  $\delta$ .
3. Set  $j$  to 0.
4. While  $j < n$  and  $\max(|\delta^i|) > \epsilon$  for  $i = 1, \dots, m$  do
    (a) Set  $a_j$  to  $a_j + \frac{\tau_j \cdot \delta}{\tau_j \cdot \tau_j}$ .

    (b) Compute the error vector,  $\delta$ .
    (c) For  $k = (j-1), (j-2), \dots, 0$  do
        Compute the error vector,  $\delta$ .
        Set  $a_k$  to  $a_k + \frac{\tau_k \cdot \delta}{\tau_k \cdot \tau_k}$ .
        Compute the error vector,  $\delta$ .
    Endfor
    (d) Set  $j$  to  $j+1$ 
EndWhile
5. Return the set of coefficients,  $a_j$ .

```

Figure 6: The CVB approximation algorithm (see section 2.4). Computational details have been omitted that deal with avoidance of representational and computational error.

2.4.1 Convergence of the CVB approximation algorithm

To prove convergence, we consider the general situation in which we have visited the first p term vectors and still have a non-zero residual error vector. We note that we can eliminate this error if we can approximate the error vector using a linear combination of the Chebyshev polynomials that have not yet been visited. We use the real-valued version of the Stone-Weierstrass theorem [6] to prove that this is possible.

Theorem 4 () *Let X be a compact set and let $C(X)$ denote the space of continuous real-valued functions defined on X . Assume that A is a subalgebra of $C(X)$. Then A is dense in $C(X)$ in the uniform norm iff A separates points and for each $x \in X$ there exists an $f \in A$ satisfying $f(x) \neq 0$.*

Proof

To prove that the polynomial vector space P with basis set $\{T_n \mid n > p\}$ is dense in $C(X)$, where X is the interval $[-1, 1]$, prove that:

1. X is compact.
2. P is a subalgebra of $C(X)$.
3. P separates points (for any distinct points, $x_1, x_2 \in X$, there is a $T_n, n > p$ s.t. $T_n(x_1) \neq T_n(x_2)$).
4. For each $x \in X$ there is a $T_n, n > p$ s.t. $T_n(x) \neq 0$.

Taking each condition in turn:

1. X is a closed and bounded interval on the real number line and is therefore compact.
2. P is a subalgebra of $C(X)$ since P is closed for the usual multiplication and addition of functions, and for scalar multiplication by real values. This is sufficient to ensure that the defining properties of an algebra hold for the subspace, P as they do for $C(X)$.
3. Let $x_1, x_2 \in [-1, 1]$ such that $x_1 \neq x_2$ and $T_n(x_1) = T_n(x_2)$. Given that $T_n(x) = \cos(n \cos^{-1}(x))$, let $x_1 = \cos(\theta_1)$ and $x_2 = \cos(\theta_2)$ with $\theta_1 > \theta_2$ and $\theta_1, \theta_2 \in [0, \pi]$. If $T_n(x_1) = T_n(x_2)$, we get $\cos(n\theta_1) = \cos(n\theta_2)$. This implies that either $n\theta_1 \pmod{2\pi} = n\theta_2 \pmod{2\pi}$ or $-(n\theta_1 \pmod{2\pi}) = 2\pi - (n\theta_2 \pmod{2\pi})$, which gives

$$\theta_1 - \theta_2 = \frac{2\pi N}{n} \text{ for some integer, } N \quad (8)$$

Since $\theta_1, \theta_2 \in [0, \pi]$ and $\theta_1 > \theta_2$, it follows from (8) that:

$$0 < \frac{N}{n} < \frac{1}{2} \quad (9)$$

Now, $T_{n+1}(x_1) = \cos((n+1)\theta_1)$. So that from (8):

$$(n+1)\theta_1 = 2\pi \left(N + \frac{N}{n} \right) + (n+1)\theta_2 \quad (10)$$

From (9) and (10) we can conclude that $\cos((n+1)\theta_1) \neq \cos((n+1)\theta_2)$. So that T_{n+1} separates the points.

4. The zeroes of T_n are given by $\zeta_j^{(n)} = \cos\left(\frac{(2j-1)\pi}{2n}\right)$ for $j = 1, \dots, n$. So that $T_n(\zeta_j^{(n)}) = 0$. Let $\theta_j^{(n)} = \frac{(2j-1)\pi}{2n}$ so that $\zeta_j^{(n)} = \cos(\theta_j^{(n)})$. If $\zeta_j^{(n)}$ is also a zero of T_{n+1} then $T_{n+1}(\zeta_j^{(n)}) = \cos((n+1)\theta_j^{(n)}) = 0$. This implies that $\theta_j^{(n)}$ is a multiple of π . However, by definition, this is not the case. So $\zeta_j^{(n)}$ cannot also be a zero of T_{n+1} . Therefore, for any $x \in [-1, 1]$, if $T_n(x) = 0$, then $T_{n+1}(x) \neq 0$ and the result follows. ■

Since we are solving the problem in the Cartesian m -space identified by the m sample points, we must prove that the subspace identified by the set of τ_n , where $n > p$, contains term vectors a linear combination of which will yield a vector arbitrarily close to the error vector.

Proof

Since the term vectors are derived from the Chebyshev polynomials and the subspace

P is dense in $C(X)$, it follows that there is a linear combination of Chebyshev polynomials in this subspace that is arbitrarily close to a function passing through the points of the error vector. This will identify a linear combination of term vectors that is arbitrarily close to the error vector. There will therefore always be at least one more term vector that reduces the magnitude of the error vector.

■

As for the rate of convergence, it is a well-known result in approximation theory that if m is the number of sample points, good results can be obtained by placing your sample points at the zeroes of T_{m+1} . This yields m orthogonal term vectors. For these points, the CVB approximation algorithm yields the same solution as the CVB interpolation algorithm. Furthermore, the quality of the approximation is consistent with the shape fitting properties of the CVB approximation algorithm.

Since we were using noisy data, we did not exploit this result in our application. Instead, we used heavy sampling of the particular area of interest (the region of the workspace that the robot can actually reach) with minimal sampling of points at the limits of the x and y intervals (i.e. the corners of the rectangular workspace). In general, because the norm of the residual error vector decreases as new terms are fitted, and in most cases only a component of the error vector is removed with each term visited, convergence slows down as the algorithm progresses.

3 The bivariate CVB approximation algorithm

We use a bivariate version of the CVB approximation algorithm to map image pixels onto object locations. The bivariate approximation problem is defined as follows:

Definition 5 (The bivariate polynomial approximation problem) *Find a polynomial $P(x, y)$ that fits a set of m sample points (x_i, y_i, z_i) ; where $i = 1, 2, \dots, m$ and the ordered pairs (x_i, y_i) are distinct; such that:*

$$\max_{i=1}^m |z_i - P(x_i, y_i)| \leq \epsilon, \text{ for } \epsilon \geq 0$$

Thus:

$$P(x, y) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_{i,j} T_i(x) T_j(y) \quad (11)$$

where T_k is the $(k+1)$ th Chebyshev polynomial of the first kind; $x, y \in [-1, 1]$; $a_{i,j} = 0$ if $i + j \geq n$ (i.e. the coefficients form a triangular array). As defined, $P(x, y)$ is a bivariate polynomial of degree $n - 1$.

Casting this problem in Cartesian vector space, we define the bivariate CVB polynomial approximation problem as follows:

Definition 6 (The bivariate CVB polynomial approximation problem) *Given*

- *a set of points, $(x_i, y_i, z_i) \in \mathbb{R}^3, i = 1, \dots, m$;*

- a set of m -dimensional Cartesian vectors, $\tau_{j,k}$, for $j + k < n$ and $j, k = 0, \dots, n - 1$, such that the i 'th component of $\tau_{j,k}$ is equal to $T_j(x_i)T_k(y_i)$, for $i = 1, \dots, m$;
- a Cartesian vector, γ such that the i 'th component of γ is equal to z_i , for $i = 1, \dots, m$;
- a real value ϵ ($\epsilon > 0$);

find values for a set of scalar quantities, $a_{j,k}$, such that

- $\rho = \sum a_{j,k} \tau_{j,k}$ for $j + k < n$ and $j, k = 0, \dots, n - 1$;
- $\delta = \gamma - \rho$;
- $|\delta^i| \leq \epsilon$ for each component, δ^i , of δ ($i = 1, \dots, m$).

So that the bivariate CVB polynomial approximation problem involves a search for a linear combination of m -dimensional Cartesian vectors; where m is the number of sample points. Once the problem has been cast as a CVB problem, the coefficients of the solution are found through a progressive reduction of the residual error vector as for the univariate CVB polynomial approximation problem.

In the univariate version of the CVB approximation algorithm, terms were visited in order of increasing degree of the corresponding Chebyshev polynomial. In the bivariate case, it is not immediately obvious how to define the order of preference of terms, since several terms can be of the same degree. The following preferencing rules produce acceptable results for our application:

Let $\langle i, j \rangle$ denote the term with coefficient $a_{i,j}$:

- The term $\langle a, b \rangle$ is visited before term $\langle c, d \rangle$ if $a + b < c + d$. That is, $i + j$ is used as a primary measure of the “preference” for fitting one term over another.
- If $a + b = c + d$, then $\langle a, b \rangle$ is visited before term $\langle c, d \rangle$ if $\min(a, b) < \min(c, d)$. That is, $\min(i, j)$ is used as a secondary heuristic preference metric.
- If $a + b = c + d$ and $\min(a, b) = \min(c, d)$, then $\langle a, b \rangle$ is visited before term $\langle c, d \rangle$ if $a < c$. This is an arbitrary ordering rule to sequence terms of equal preference.
- After term $\langle i, j \rangle$ is visited, only terms $\langle a, b \rangle$ where $a \leq i$ and $b \leq j$ are revisited and revisits take place in reverse order to visits.

Informally, when visiting, these rules give preference to a term that is of lower degree in the first instance, or if of equal degree, has a component that is of lower degree than any component of a lesser preferred term. When revisiting, the same preferences are observed, but revisits are restricted to terms with components no greater than the corresponding components of the last visited term. So that in general, preference is given to lower order terms. Proof of convergence is similar to that for the univariate case given in section 2.4.1.

3.1 Using the bivariate CVB approximation algorithm

Fig. 1 (a) shows a deliberately severe misalignment of the camera. The image of the workspace appears to be rotated in a clockwise direction. The camera also produces a pincushion distortion that is reportedly imperceptible to most observers but is significant with respect to locating objects on the workspace. If uncorrected, distortion results in an error of as much as eight millimetres in the location of an object (corresponding to a misplacement of four pixels in the location of an image point).

In comparison, Fig. 1 (b) shows the results of using the bivariate CVB approximation algorithm to rectify the image shown in Fig. 1 (a). The rectified image shows a rectangular border to the workspace with edges that are better aligned horizontally and vertically, as can be seen by comparing their alignment with the superimposed grid. In practice, the misalignment will also be imperceptible, but both distortion and misalignment are sufficient to cause errors in locating objects on the workspace.

No attempt was made to ensure that the key points in the test pattern were evenly spaced, or placed according to the zeros of a Chebyshev polynomial. Instead, all but two of the key points¹ were used to sample the region of the workspace that is within reach of the robot. The two extra points were used primarily to fix the corners of the workspace for illustrative purposes.

An even spacing of the key points in the test pattern is of questionable utility since this does not guarantee an even spacing of their images due to distortion and misalignment. Fig. 4 illustrates the stability of approximations as compared to interpolations when unevenly spaced points are used. This characteristic of approximations was exploited here.

As for using the zeroes of a Chebyshev polynomial, there is the question of which Chebyshev polynomial to use, since, for the CVB approximation algorithm, it is not predetermined how many Chebyshev terms will be included in the approximation. For the rectification depicted in Fig. 1, the CVB approximation algorithm yielded a solution with 27 bivariate Chebyshev terms. This yields a polynomial of degree 7. A bivariate polynomial of degree n can have up to $\frac{1}{2}(n+1)(n+2)$ terms. Given that twenty sample points were used and a bivariate polynomial of degree $20^2 = 400$ can have up to 80601 terms, this represents a significant saving in computational overhead.

4 Summary and conclusions

We have developed what we believe to be a novel algorithm for global polynomial approximation. We call this algorithm the Cartesian Vector Based approximation algorithm, or CVB approximation algorithm. This algorithm has several desirable features:

- It does not require one to fix the degree of the approximating polynomial ahead of time. The algorithm is capable of progressively adding polynomial terms until the required precision of fit is achieved (or some specified limit on resources is reached).

¹The key points are the centres of the solid circles in the test pattern shown in Fig. 1.

- Since the algorithm progressively converges on a “closest fitting function”, an approximate solution is available after each iteration. The longer the algorithm runs, the better the fit.
- The algorithm yields better results on the type of data that usually presents difficulties for global polynomial interpolation.

It has been proven that the algorithm will yield an approximation that is within ϵ of a perfect fit, where $\epsilon > 0$ and the uniform norm is used as the distance metric. The rate of convergence depends on the choice of data points.

Our focus has been on presenting the CVB approximation algorithm and we use image rectification to illustrate its use. We show how global polynomial approximation can be used to calibrate the vision component of a visually guided pick-and-place robot. Calibration problems are sufficiently prevalent within the field of robotics to render this example relevant.

There are, however, other types of mappings within the field of robotics to which the algorithm may not be directly applicable. For instance, in time series analysis a study is made of a time-varying information-carrying signal for the purpose of predicting its future behaviour. Since time series analysis attempts to predict future events, there is an emphasis on extrapolation. Our algorithm is based on interpolation rather than extrapolation. It is an open question whether it can be adapted for time series analysis.

Systems identification is another area that uses function approximation [9]. For instance, ARMAX/NARMAX models are parametrised models of systems consisting of time-varying input values and output values, where the assumption is that the output values depend in some way on current and past input values. System identification involves finding an instantiation of the parameters that yields a predictor of system behaviour. If a parametrised system model can be contrived that is polynomial in form with the parameters appearing as coefficients, and the data can be transformed to represent points on this polynomial, then any polynomial approximation method (including ours) may be used to estimate the parameters.

In a case where several approximations exist, there is the question of how the characteristics of the approximation relate to the adequacy of the resulting system model. More specifically, it may be necessary to minimise some domain specific cost function that includes more than the error in the approximation and the contribution of higher order polynomial terms. By limiting the scope of this paper to the details of the CVB approximation algorithm per se, we leave such domain specific questions open for future discussion.

§§§

References

- [1] L. G. Brown, “A survey of image registration techniques,” *ACM Computing Surveys*, vol. 24, no. 4, pp. 325–376, 1992.

- [2] P. F. Whelan and D. Molloy, *Machine Vision Algorithms in Java: Techniques and Implementation*. Secaucus, NJ, USA: Springer-Verlag London Limited, 2001.
- [3] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Trans. Neural Netw.*, vol. 9, pp. 987–995, Sep 1998.
- [4] S. Ferrari and R. F. Stengel, "Smooth function approximation using neural networks," *IEEE Trans. Neural Netw.*, vol. 16, pp. 24–38, Jan 2005.
- [5] E. Cheney, *Introduction to approximation Theory*. New York: McGaw-Hill Book Company, 1966.
- [6] A. Pinkus, "Density in approximation theory." <http://www.math.technion.ac.il/sat>.
- [7] G. E. Forsythe, M. A. Malcolm, and C. B. Moler, *Computer Methods for Mathematical Computations*. Prentice Hall Professional Technical Reference, 1977.
- [8] L. V. Fausett, *Numerical methods: algorithms and applications*. Pearson Education, Inc., 2003.
- [9] L. Ljung, *System Identification: theory for the user*. Prentice Hall Information and System Sciences Series, 1999.
- [10] C. O. Ward, "Using polynomial approximation to rectify distorted images," in *Proceedings of IASTED MS2005*, ACTA Press, 2005.