STUDIA UNIV. "BABEŞ–BOLYAI", MATHEMATICA, Volume **XLVII**, Number 4, December 2002

# COMBINED SHEPARD-LEAST SQUARE OPERATORS – COMPUTING THEM USING SPATIAL DATA STRUCTURES

MARIA GABRIELA TRÎMBIŢAŞ

*Dedicated to Professor D.D. Stancu on his 75th birthday*

**Abstract**. The low degree of exactness and large number of computation required are well-known drawbacks of classical Shepard operator. They can be overcome using combined Shepard operators and local interpolation schemes. Spatial data structures could support efficient evaluation of such operators.

## 1. **Introduction**

Unfortunately, the classical Shepard operator (see [1]) defined by

$$\left(S_{n,\mu}f\right)(x)=\sum_{k=0}^{n} w_k(x)f(x_k) \tag{1}$$

$$w_k(x)=\frac{|x-x_k|^{-\mu}}{\sum\limits_{k=0}^{n}|x-x_k|^{-\mu}}, \tag{2}$$

where $|.|$ denotes the Euclidean norm in $\mathbb{R}^s$, and $X=\{x_0,x_1,\ldots,x_n\}\subset\mathbb{R}^s$ is a set of $n+1$ pairwise distinct points, has a low degree of exactness (i.e. 0) and requires a large amount of computation. The solution is to replace the values of $f$ by a suitable polynomial interpolation operator $(L_kf)(x;x_k)$, which can depend on $k$ (see [2, 3], and the references therein) and the weight functions given by (2) with the so called Franke-Little weights:

$$\bar{w}_k(x)=\frac{\dfrac{(R-|x-x_k|)_+^{\mu}}{R^{\mu}|x-x_k|^{\mu}}}{\sum\limits_{i=0}^{n}\dfrac{(R-|x-x_i|)_+^{\mu}}{R^{\mu}|x-x_i|^{\mu}}} \tag{3}$$

Received by the editors: 08.10.2002.

(see [7, 6, 5]). In (3), $R$ is a given positive real constant, and the $+$ subscript denotes the positive part.

The local variant of (1) is

$$\left(\bar{S}^L_{n,\mu}f\right)(x) = \sum_{k=0}^{n} \bar{w}_k(x)(L_kf)(x;x_k), \tag{4}$$

called the *combined local Shepard-type operator.*

In order to compute the various local Shepard-type interpolants we are interested to report efficiently the point located into the ball $B(x,R)$. The naive approach (computing $d_k = |x - x_k|$ and checking $d_k < R$) needs a time $O(n)$ for each point $x$. Computational geometry techniques and data structures allow us to perform this task in polylogarithmic time [4].

In this paper $L_mf$ will be a least square approximation polynomial.

## 2. Combined Shepard least-square local operators

We shall consider two kind of discrete least-square approximation polynomials:

1. polynomials which reproduces the values of $f$ in $x_k$, $k = \overline{0,n}$;
2. polynomials which reproduces the values of $f$ and of the first order partial derivatives of $f$ in $x_k$, $k = \overline{0,n}$.

Only the bivariate operators will be considered.

**Proposition 1.** *The following relation hold*

$$\left(\bar{S}^L_{n,\mu}f\right)(x_j,y_j) = (L_jf)(x_j,y_j)$$

*and*

$$\begin{aligned}\frac{\partial}{\partial x}\left(\bar{S}^L_{n,\mu}f\right)(x_j,y_j) &= \frac{\partial}{\partial x}(L_jf)(x_j,y_j)\\ \frac{\partial}{\partial y}\left(\bar{S}^L_{n,\mu}f\right)(x_j,y_j) &= \frac{\partial}{\partial y}(L_jf)(x_j,y_j)\end{aligned}$$

**Proof.** It can be shown that, for all $k$ and $j$, the weights (3) satisfy

$$\bar{w}_k\left(x_j, y_j\right) = \delta_{kj} = \begin{cases} 1 & \text{if } k = j, \\ 0 & \text{otherwise.} \end{cases}, \tag{5}$$

$$\sum_{k=0}^{n} \bar{w}_k\left(x, y\right) = 1, \tag{6}$$

and

$$\frac{\partial}{\partial x}\bar{w}_k\left(x_j, y_j\right) = \frac{\partial}{\partial y}\bar{w}_k\left(x_j, y_j\right) = 0. \tag{7}$$

(5) implies

$$\left(\bar{S}_{n,\mu}^{L} f\right)\left(x_j, y_j\right) = \sum_{k=0}^{n} \bar{w}_k\left(x_j, y_j\right)\left(L_k f\right)\left(x_j, y_j\right) = \left(L_j f\right)\left(x_j, y_j\right);$$

From (5) and (7), one obtains

$$\begin{aligned} \frac{\partial}{\partial x}\left(\bar{S}_{n,\mu}^{L} f\right)\left(x_j, y_j\right) = \sum_{k=0}^{n} \Bigg[ & \frac{\partial}{\partial x}\bar{w}_k\left(x_j, y_j\right)\left(L_k f\right)\left(x_j, y_j\right) \\ & + \bar{w}_k\left(x_j, y_j\right)\frac{\partial}{\partial x}\left(L_k f\right)\left(x_j, y_j\right)\Bigg] = \frac{\partial}{\partial y}\left(L_j f\right)\left(x_j, y_j\right), \end{aligned}$$

and analogously in $y$. □

Thus $\left(\bar{S}_{n,\mu}^{L} f\right)$ maintains the local shape properties of $L_k f$.

Let $f_k$ be $f(x_k, y_k)$, $f_{x,k}$ be $\frac{\partial}{\partial x} f(x_k, y_k)$ and $f_{y,k}$ be $\frac{\partial}{\partial y} f(x_k, y_k)$ respectively.

For the first case $L_k f$ is defined by

$$\begin{aligned} \left(L_k f\right)(x, y) := {} & c_{k1}^{*}(x - x_k)^2 + c_{k2}^{*}(x - x_k)(y - y_k) + c_{k3}^{*}(y - y_k)^2 \\ & + c_{k4}^{*}(x - x_k) + c_{k5}^{*}(y - y_k) + f_k. \end{aligned} \tag{8}$$

The coefficients are the solution of the following discrete least-square problem

$$\begin{aligned} \sum_{\substack{i=0 \\ i \neq k}}^{n} \omega_i(x_k, y_k) & \left[c_{k1}(x - x_k)^2 + c_{k2}(x - x_k)(y - y_k) + c_{k3}(y - y_k)^2\right. \\ & \left. + c_{k4}(x - x_k) + c_{k5}(y - y_k) + f_k - f_i\right]^2 \longrightarrow \min, \end{aligned} \tag{9}$$

where

$$\omega_i(x, y) = \left[\frac{\left(R_q - d_i\right)_+}{R_q d_i}\right]^2,$$

$R_q$ is a radius of influence about node $(x_i, y_j)$ (in general not equal to $R$) and $d_i$ is the Euclidean distance between $(x, y)$ and $(x_i, y_i)$. The problem (9) leads us to a $5 \times 5$ system of linear equations.

Another possible choice for $L_k$ is

$$(L_k f)(x, y) := c_{k1}^*(x - x_k) + c_{k2}^*(y - y_k) + f_k; \tag{10}$$

analogously, in this case, we obtain a $2 \times 2$ system of linear equation. It is easy to show, using Proposition 1, that the combined local Shepard operators obtained in this way reproduce the values of $f$ in $x_k$, $k = \overline{0, n}$.

For the second case we choose

$$\begin{aligned}(L_k f)(x, y) := {} & c_{k1}^*(x - x_k)^2 + c_{k2}^*(x - x_k)(y - y_k) + c_{k3}^*(y - y_k)^2 \\ & + f_{x,k}(x - x_k) + f_{y,k}(y - y_k) + f_k.\end{aligned} \tag{11}$$

The corresponding least square problem is

$$\begin{aligned}\sum_{\substack{i=0 \\ i \neq k}}^{n} \omega_i(x_k, y_k) & \left[c_{k1}(x - x_k)^2 + c_{k2}(x - x_k)(y - y_k) + c_{k3}(y - y_k)^2\right. \\ & \left. + f_{x,k}(x - x_k) + f_{y,k}(y - y_k) + f_k - f_i\right]^2 \longrightarrow \min,\end{aligned} \tag{12}$$

and it leads us to a $3 \times 3$ system of linear equations.

Another possibility is given by

$$\begin{aligned}(L_k f)(x, y) := {} & c_{k1}^*(x - x_k)^3 + c_{k2}^*(x - x_k)^2(y - y_k) \\ & + c_{k3}^*(x - x_k)(y - y_k)^2 + c_{k,4}^*(y - y_k)^3 \\ & + c_{k1}^*(x - x_k)^2 + c_{k2}^*(x - x_k)(y - y_k) \\ & + c_{k3}^*(y - y_k)^2 + f_{x,k}(x - x_k) + f_{y,k}(y - y_k) + f_k.\end{aligned} \tag{13}$$

The choice (8) appears in [6, 7, 5, 9, 8, 10, 11], (10) in [11], but (11) and (13) are original.

The efficient computation of the operator given by (4) requires the efficient solving of a circular range searching problem.

Let $P := \{p_1, \ldots, p_n\}$ be a set of point from $\mathbb{R}^s$ and $R$ a region from the same space. A $s$-dimensional *range searching problem* asks for the points from $P$ lying inside the query region $R$. If the region is a hyperparallelepiped, i.e. $R = [x_1, x_1'] \times \cdots \times [x_s, x_s']$, then we have an *orthogonal range-searching problem.* If $R$

is a ball from $\mathbb{R}^s$, we have a *circular range searching problem.* Our approach is to solve a simpler orthogonal range searching problem instead of the circular range searching (since this approach eliminates a large number of points) and then to check the reported points.

## 3. Implementation

One of the most used data structure for orthogonal range query is the *range tree*[4]. A solution based on range tree is given in [12].

Another solution is inspired from a paper of Renka[8]. The smallest bounding box containing the nodes $\prod_{k=1}^{s}\left[x_{\min}^k, x_{\max}^k\right]$ is partitioned into an uniform grid of cells, having $NR$ cells on each dimension. Each cell points to the list of point indices contained in that cell. Such an example for the 2D case is given in Figure 1.

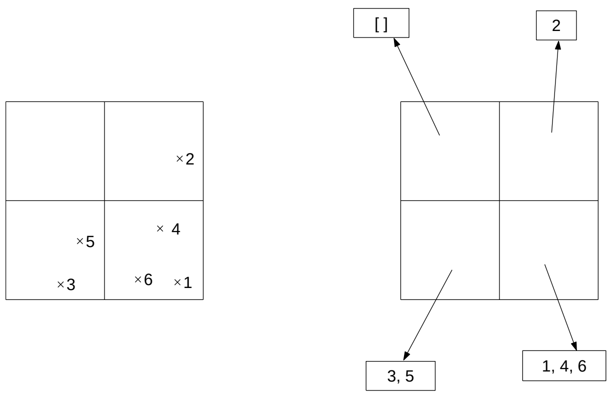


FIGURE 1. A 2D grid of cell and its representation

The algorithm 1 describes the creation of the data structure. If the second argument $NR$ is not provided we can initialize it with a default value; Renka suggests in [9]

$$NR = \lfloor (N/3)^{1/\dim} \rfloor.$$

The orthogonal range searching is easy to implement using this data structure (the algorithm 2): first the cell which must be scanned are determined (i.e. the cell which intersects the searching domain), and then the list of points corresponding to that cell are concatenated. The points from the outer cells which lie outside the searching range must be eliminated.

**Algorithm 1** Creating the cell grid

```
**Input**: the set of $N$ points $P$, the number of cells, NR (optional);
**Output**: a grid of cell LCELL, each containing the list of points in the cell
set all cells to **nil**;
{compute the cell sizes}
$dc_1 := \min(NR, \lfloor x^1_{\max} - x^1_{\min} \rfloor + 1)$;
$\vdots$
$dc_s := \min(NR, \lfloor x^s_{\max} - x^s_{\min} \rfloor + 1)$;
**for** $K := N$ **downto** 1 **do**
  {find the cell}
  $i_1 := \min(NR, \lfloor x^k_1 - x^1_{\min} \rfloor + 1)$;
  $\vdots$
  $i_s := \min(NR, \lfloor x^k_s - x^s_{\min} \rfloor + 1)$;
  add $K$ to the list $LCELL(i_1, \dots, i_s)$;
**end for**
```

Now we are able to compute the local Shepard interpolant on a set of points $X$:

- build the spatial data structure;
- for each point $x$ in $X$
  - perform the orthogonal range searching into the hypercube centered in $x$ and with the radius $R$
  - apply formulas (3) and (4).

This approach has a drawback: the accuracy tends to decrease into the areas where the interpolation nodes are sparse. We can avoid this situation, allowing $R_q$ and $R$ to vary with $k$: the radii are choosen such that the ball $B(x_j, R_q)$ contains at least $N_q$ nodes and the ball $B(x, R)$ contains at least $N_w$ nodes. Thus, instead of an orthogonal range searching we perform a $N_q$-th (or a $N_w$-th) nearest neighbor search of $x_j$ and $x$, respectively. This can be done scanning the grid in a circular fashion starting with the cell containing $x$. In order to facilitate the scanning we can associate a Boolean indicator to each cell, which is true when the cell was already scanned.

**Algorithm 2** The orthogonal range searching

```
$PTLIST :=$ **nil**;
{determine the outer cells, i. e. the scan limits}
$imin_1 := \max(1, \lfloor (liminf_1 - x^1_{\min})/dc_1 \rfloor + 1)$;
$imax_1 := \min(NR, \lfloor (limsup_1 - x^1_{\min})/dc_1 \rfloor + 1)$
$\vdots$
$imin_s := \max(1, \lfloor (liminf_s - x^s_{\min})/dc_s \rfloor + 1)$;
$imax_s := \min(NR, \lfloor (limsup_s - x^s_{\min})/dc_s \rfloor + 1)$
for $i_1 := imin_1$ to $imax_1$ do
  $\vdots$
  for $i_s := imin_s$ to $imax_s$ do
    $JL := LCELL(i_1, \ldots, i_s)$;
    if the cell $(i_1, \ldots, i_s)$ is peripheral then
      remove the points which lay cell outside the searching range from $JL$;
    end if
    concatenate $PTLIST$ and $JL$
  end for
  $\vdots$
end for
```

The algorithms described above are implemented in MATLAB[1]. The cell grid is represented as a structure which contains information about the grid: dimension, number of cell over each coordinates, the size of a cell, minimum and maximum in each coordinate and a cell array, where each cell contains an array with point indices; this representation allows easy location of cell and concatenation of point lists.

The linear system which gives the solution of least square system can be ill-conditioned. For this reason the system is solved using a QR factorization. If the results are not satisfactory (system too ill-conditioned) more points are added, and the solving process is redone.

[1]MATLAB©is a trademark of the MathWorks, Inc. Natick, MA 01760-2098

## 4. Examples and graphs

One of the most frequent function used as example to illustrate Shepard interpolation is the Franke's function [6, 8], $f_1 : [0,1] \times [0,1] \to \mathbb{R}$, given by:

$$\begin{aligned} f_1(x,y) =& 0.75 \exp\left(-\left((9x-2)^2+(9y-2)^2\right)/4\right) \\ &+ 0.75 \exp\left(-(9x+1)^2/49-(9y+1)/10\right) \\ &+ 0.5 \exp\left(-\left((9x-7)^2+(9y-3)^2\right)/4\right) \\ &- 0.2 \exp\left(-(9x-4)^2-(9y-7)^2\right). \end{aligned}$$

Its graph appears in Figure 2(a). Figures 2(b) and 2(c) give the graphs of local Shepard operator combined with a least square polynomial having the degree 1 (given by formula (10)) and 2 (formula (8)), respectively. The graph of the local Shepard interpolant combined with a 2nd degree least square polynomial, considering first order partial derivatives (formula (11)) is given in Figure 2(d). All the interpolants were obtained taking $\mu = 2$, $N_q = 17$ and $N_w = 23$. The best result is obtained for the 2nd degree least square polynomial, without derivatives. This phenomenon deserves further investigations.

## References

[1] D. Shepard – A Two Dimensional Interpolation Function for Irregularly Spaced Data, *Proc. 23rd Nat. Conf. ACM*, 517-523, 1968.

[2] Gh. Coman, R. Trîmbiţaş – Bivariate Shepard interpolation, "Babeş-Bolyai" University, Faculty of Mathematics and Computer Science, *Seminar on Numerical and Statistical Calculus, Preprint* No. 1, 41–83, 1999.

[3] R. Trîmbiţaş – Local bivariate Shepard interpolation, Proceedings of the 4-th International Conference on Functional Analysis and Approximation Theory, Acquafedda di Maratea, Italy, September 2000, *Supplemento ai Rendiconti del Circolo Matematico di Palermo*, Serie II, No. 68, 865–878, 2002.

[4] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf – Computational Geometry. Algorithms and Applications, Springer Verlag, Berlin, Heidelberg, New York, 1997.

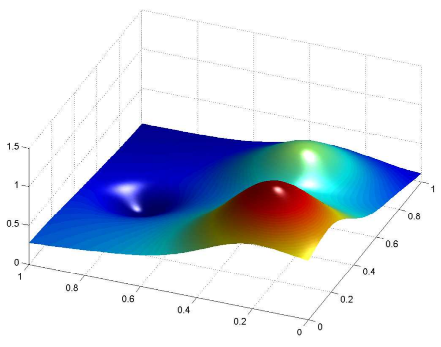
[5] R. Franke, G. M. Nielson – Smooth interpolation of large sets of scattered date, *International Journal for Numerical Methods in Engineering*, 15, 1691–1704, 1980.

[6] R. Franke – Scattered data Interpolation: Tests of Some Methods, *Math. Comp.* 38, 181–200, 1982.

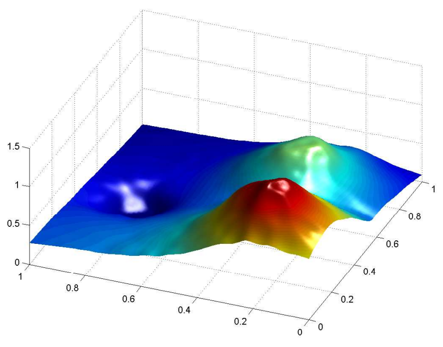
[7] R. Franke, G. M. Nielson – Scattered Data Interpolation and Applications: A tutorial and Survey, in *Geometric Modeling - Methods and Applications*, H. Hagen, D. Roller (eds), Springer Verlag, 1990.

[8] R. J. Renka – Multivariate Interpolation of Large Sets of Scattered Data, *ACM Transactions on Mathematical Software*, Vol. 14, No. 2, 139-148, 1988.

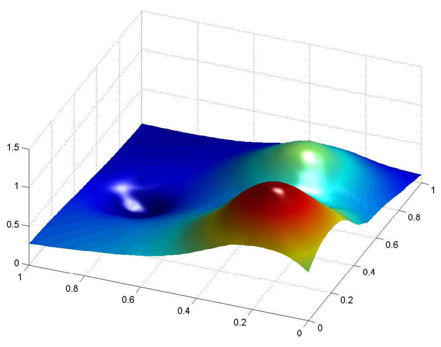
[9] R. J. Renka – Algorithm 660, QSHEP2D: Quadratic Shepard Method for Bivariate Interpolation of Scattered Data, *ACM Transactions on Mathematical Software*, Vol. 14, No. 2, 149, 1988.



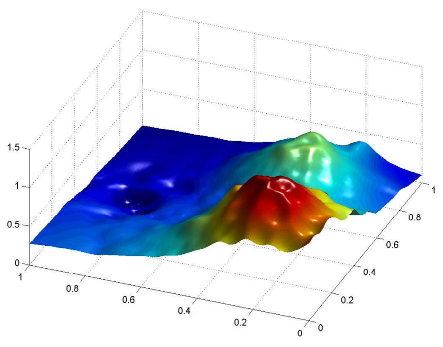
(a) The graph of $f_1$



(b) Local Shepard-least square, degree 1



(c) Local Shepard-least square, degree 2



(d) Local Shepard-least square, degree 2, with derivatives

FIGURE 2. $f_1$ and various Shepard-least square interpolants for $\mu = 2$

[10] R. J. Renka – Algorithm 661, QSHEP3D: Quadratic Shepard Method for Trivariate Interpolation of Scattered Data, *ACM Transactions on Mathematical Software*, Vol. 14, No. 2, 151, 1988.

[11] K. Basso, P. R. A. Zingano, C. M. S. Freitas, Interpolation of Scattered Data: Investigating Alternatives for the Modified Shepard Method, Proceedings of SIBGRAPI ' 98, Rio de Janeiro, 1998.

[12] M. G. Trîmbiţaş, R. Trîmbiţaş, Range searching and local Shepard interpolation, Proceedings of Roger 2002, to appear.

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE,
"BABEŞ-BOLYAI" UNIVERSITY, STR. KOGĂLNICEANU NO. 1,
RO-3400 CLUJ-NAPOCA, ROMANIA
*E-mail address*: `gabitr@cs.ubbcluj.ro`