Afrigraph 2010, pp.17-20, ACM, ISBN:978-1-4503-0118-3, 2010

Progressive RBF Interpolation

Vaclav Skala
Department of Computer Science and Engineering
Faculty of Applied Sciences, University of West Bohemia
CZ 306 14 Plzen, Czech Republic
skala@kiv.zcu.cz

Abstract

Interpolation based on Radial Basis Functions (RBF) is very often used for scattered scalar data interpolation in *n*-dimensional space in general. RBFs are used for surface reconstruction of 3D objects, reconstruction of corrupted images etc. As there is no explicit order in data sets, computations are quite time consuming that leads to limitation of usability even for static data sets. Generally the complexity of computation of RBF interpolation for *N* points is of $O(N^3)$ or $O(k\,N^2)$, *k* is a number of iterations if iterative methods are used, which is prohibitive for real applications. The inverse matrix can also be computed by the Strassen algorithm based on matrix block notation with $O(N^{2.807})$ complexity. Even worst situation occurs when interpolation has to be made over non-constant data sets, as the whole set of equations for determining RBFs has to be recomputed. This situation is typical for applications in which some points are becoming invalid and new points are acquired.

In this paper a new technique for incremental RBFs computation with complexity of $O(N^2)$ is presented. This technique enables efficient insertion of new points and removal of selected or invalid points. Due to the formulation it is possible to determine an error if one point is removed that leads to a possibility to determine the most important points from the precision of interpolation point of view and insert gradually new points, which will progressively decrease the error of interpolation using RBFs. The *Progressive RBF Interpolation* enables also fast interpolation on "sliding window" data due to insert/remove operations which will also lead to a faster rendering.

Categories and Subject Descriptors

I.3.5 [Computational Geometry and Object Modeling]: Boundary representations, Splines; I.4.5 [Image Processing and Computer Vision]: Reconstruction; G.1.1 [Interpolation]: Interpolation formulas, Smoothing, Spline and piecewise polynomial interpolation; Interpolation formulas; G.1.3 [Numerical Linear Algebra] Matrix inversion

General Terms: Algorithms, Performance, Reliability, Theory.

Keywords

RBF, interpolation, graphics and vision, incremental computation.

1. Introduction

Radial basis functions interpolation was originally introduced by [Hardy 1971] by introduction of multiquadric method, which he called Radial Basis Function (RBF) method, which is based on interpolation formula

$$f(x) = \sum_{i=1}^{N} \lambda_i\, \phi(r_i)$$

where: $\phi(r_i) = \phi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)$ and $\boldsymbol{x}$ is generally *n*-dimensional vector and λ_i are weights. Since then many different RFBF interpolation schemes have been developed with some specific properties, e.g. [Duchon 1977] uses $\phi(r) = r^2 lg\, r$, which is called Thin-Plate Spline (TPS), a function $\phi(r) = e^{-(\epsilon r)^2}$ was proposed by [Shagen 1979] and [Wetland 2005] introduced Compactly Supported RBF (CSRBF) as

$$\phi(r) = \begin{cases} (1-r)^q\, P(r), & 0 \le r \le 1 \\ 0, & r > 1 \end{cases},$$

where: $P(r)$ is a polynomial function and q is a parameter.

Theoretical problems with stability and solvability were solved by [Micchelli 1986] and [Wright 2003] and he has extended the RBF by adding a polynomial function $P_k(\boldsymbol{x})$ of degree *k* to the RBF that resulted to:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \lambda_i\, \phi(\|\boldsymbol{x} - \boldsymbol{x}_i\|) + P_k(\boldsymbol{x}) = \sum_{i=1}^{N} \lambda_i\, \phi_i(\boldsymbol{x}) + P_k(\boldsymbol{x})$$

and additional conditions were introduced:

$$\sum_{i=1}^{N} \lambda_i = 0 \qquad \sum_{i=1}^{N} \lambda_i\, \boldsymbol{x} = \boldsymbol{0}$$

Usually a linear polynomial is used, i.e. the polynomial $P_k(\boldsymbol{x})$ is taken as

$$P_k(\boldsymbol{x}) = a_0 + \boldsymbol{a}^T \boldsymbol{x}$$

As the values $f(\boldsymbol{x}_i)$ at points $\boldsymbol{x}_i$ are known, the equations above form a system of linear equations that has to be solved in order to determine coefficients λ_i and $a_0, \boldsymbol{a}$, i.e.

$$f(\boldsymbol{x}_j) = \sum_{i=1}^{N} \lambda_i\, \phi(\|\boldsymbol{x}_j - \boldsymbol{x}_i\|) + P_k(\boldsymbol{x}_j) = \sum_{i=1}^{N} \lambda_i\, \phi_{i,j} + P_k(\boldsymbol{x}_j)$$
$$j = 1, \dots, n$$

It can be seen that for *n*-dimensional case and *N* points given a system of $(N + n + 1)$ has to be solved, where *N* is a number of points in the dataset and *n* is dimensionality of data. For *n=2* vectors $\boldsymbol{x}_i$ and $\boldsymbol{a}$ are given as $\boldsymbol{x}_i = [x_i, y_i]^T$ and $\boldsymbol{a} = [a_x, a_y]^T$.

Afrigraph 2010, pp.17-20, ACM, ISBN:978-1-4503-0118-3, 2010

Using the matrix notation we can write for 2-dimensions:

$$\begin{bmatrix} \phi_{1,1} & .. & \phi_{1,N} & x_1 & y_1 & 1 \\ : & & : & : & : & : \\ \phi_{N,1} & .. & \phi_{N,N} & x_N & y_N & 1 \\ x_1 & .. & x_N & 0 & 0 & 0 \\ y_1 & .. & y_N & 0 & 0 & 0 \\ 1 & .. & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ : \\ \lambda_N \\ a_x \\ a_y \\ a_0 \end{bmatrix} = \begin{bmatrix} f_1 \\ : \\ f_N \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{B} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{a} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix} \qquad \boldsymbol{Ax} = \boldsymbol{b}$$

$$\boldsymbol{a}^T \boldsymbol{x}_i + a_0 = a_x \, x_i + a_y \, y_i + a_0$$

It can be seen that for 2-dimensional case and N points given a system of $(N + 3)$ linear equations has to be solved. If "global" functions, e.g. TPS ($\phi(r) = r^2 lg\, r$), are used the matrix $\boldsymbol{B}$ is "full", if CSRBF functions are used, the matrix $\boldsymbol{B}$ can be sparse.

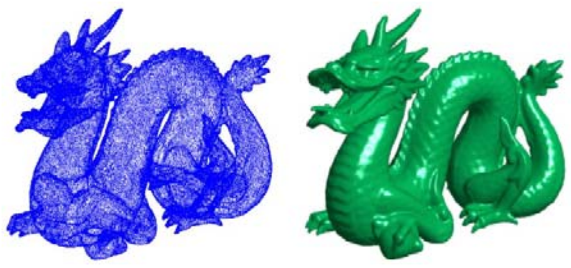
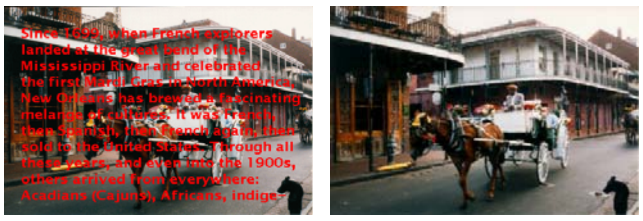


Figure 1: Surface reconstruction (438000 points) [Carr et al. 2001]


Since 1699, when French explorers
landed at the great bend of the
Mississippi River and celebrated
the first Mardi Gras in North America,
New Orleans has brewed a fascinating
melange of cultures. It was French,
then Spanish, then French again, then
sold to the United States. Through all
these years, and even into the 1900s,
others arrived from everywhere:
Acadians (Cajuns), Africans, indige-

Original image [Bertalmio et al. 2000] Reconstructed image [Uhlir and Skala 2006]

Figure 2: Reconstruction of inpainting

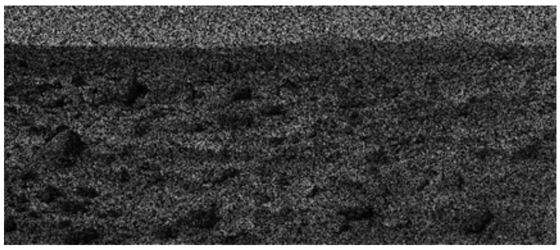


Figure 3a: Original image with 60% of damaged pixels

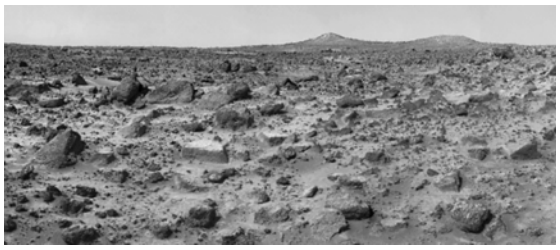


Figure 3b: Reconstructed image

Some interesting problems can be solved using RBF interpolation quite effectively, e.g. surface reconstruction from scattered data [Carr et al. 2001], [Ohtake et al. 2005], reconstruction of damaged images [Uhlir and Skala 2006], [Zapletal et al. 2009], inpainting removal [Bertalmio et al. 2000], [Wang and Kwok 2009] etc.

All those applications of RBFs based interpolation has one significant disadvantage – the cost of computation. This is especially severe in applications where data are not static. There are actually two cases:

1. **Position of points is fixed**, but the value associated with a point is changed. In this case iterative methods are usually faster than explicit computation of an inverse matrix.
2. **Position of points is changed**. It means that the whole system of linear equations has to be form and recomputed which leads generally to $O(N^3)$ computational complexity and unacceptable time consuming computation.

In some applications a "*sliding window*" on data is required, especially in time related applications, when old data should not be used in the interpolation and new data should be included. This is typical situation in signal processing applications.

Considering facts above there is a questions how to compute RBF incrementally with a lower computational complexity?. This question will be answered in the following section.

2. Incremental RBF computation

The main question to be answered is:

Is it possible to use already computed RFB interpolation if a new point is included to the data set?

If the answer is positive it should lead to significant decrease of computational complexity. In the following we will present how a new point can be inserted, a selected point can be removed and also how to select the best candidate for a removal according to an error caused by this point removal.

Let us consider some operations with block matrices (we will assume that all operations are correct and matrices are non-singular in general etc.).

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^{-1} = \begin{bmatrix} (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C})^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1} \\ -(\boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}\boldsymbol{C}\boldsymbol{A}^{-1} & (\boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1} \end{bmatrix}$$

Let us consider a matrix $\boldsymbol{M}$ of $(n+1) \times (n+1)$ and a matrix $\boldsymbol{A}$ of $n \times n$ in the following block form:

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{b}^T & c \end{bmatrix}$$

Then the inverse of the matrix $\boldsymbol{M}$ applying the rule above can be written as:

$$\boldsymbol{M}^{-1} = \begin{bmatrix} \left(\boldsymbol{A} - \frac{1}{c}\boldsymbol{b}\boldsymbol{b}^T\right)^{-1} & -\frac{1}{k}\boldsymbol{A}^{-1}\boldsymbol{b} \\ -\frac{1}{k}\boldsymbol{b}^T\boldsymbol{A}^{-1} & \frac{1}{k} \end{bmatrix}$$

$$= \begin{bmatrix} \boldsymbol{A}^{-1} + \frac{1}{k}\boldsymbol{A}^{-1}\boldsymbol{b}\boldsymbol{b}^T\boldsymbol{A}^{-1} & -\frac{1}{k}\boldsymbol{A}^{-1}\boldsymbol{b} \\ -\frac{1}{k}\boldsymbol{b}^T\boldsymbol{A}^{-1} & \frac{1}{k} \end{bmatrix}$$

where: $k = c - \boldsymbol{b}^T\boldsymbol{A}^{-1}\boldsymbol{b}$

We can easily simplify this equation if the matrix $\boldsymbol{A}$ is symmetrical as:

$$\boldsymbol{\xi} = \boldsymbol{A}^{-1}\boldsymbol{b} \qquad k = c - \boldsymbol{\xi}^{\boldsymbol{T}}\boldsymbol{b}$$

$$\boldsymbol{M}^{-1} = \frac{1}{k}\begin{bmatrix} k\boldsymbol{A}^{-1} + \boldsymbol{\xi}\otimes\boldsymbol{\xi}^{\boldsymbol{T}} & -\boldsymbol{\xi} \\ -\boldsymbol{\xi}^{\boldsymbol{T}} & 1 \end{bmatrix}$$

where: $\boldsymbol{\xi}\otimes\boldsymbol{\xi}^{\boldsymbol{T}}$ means the tensor multiplication. It can be seen that all computations needed are of $O(N^2)$ computational complexity.

It means that we can compute an inverse matrix incrementally with $O(N^2)$ complexity instead of $O(N^3)$ complexity required originally in this specific case. It can be seen that the structure of the matrix $\boldsymbol{M}$ is "similar to the matrix of the RBF specification.

Now, there is a question how the incremental computation of an inverse matrix can be used for RBF interpolation?

We know that the matrix **A** in the equation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is symmetrical and non-singular if appropriate rules for RBFs are kept.

2.1 Point Insertion

Let us imagine a simple situation. We have already computed the interpolation for *N* points and we need to include a new point into the given data set. A brute force approach of full RBF computation on the new data set can be used with $O(N^3)$ complexity computation.

Let us consider RBF interpolation for *N+1* points and the following system of equations is obtained:

$$\begin{bmatrix} \phi_{1,1} & .. & \phi_{1,N} & \phi_{1,N+1} & x_1 & y_1 & 1 \\ : & & .. & : & : & : & 1 \\ \phi_{N,1} & : & \phi_{N,N} & \phi_{N,N+1} & x_N & y_N & 1 \\ \phi_{N+1,1} & & \phi_{N+1,N} & \phi_{N+1,N+1} & x_{N+1} & y_{N+1} & 1 \\ x_1 & .. & x_N & x_{N+1} & 0 & 0 & 0 \\ y_1 & .. & y_N & y_N & 0 & 0 & 0 \\ 1 & .. & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ : \\ \lambda_N \\ \lambda_{N+1} \\ a_x \\ a_y \\ a_0 \end{bmatrix} = \begin{bmatrix} f_1 \\ : \\ f_N \\ f_{N+1} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

where: $\phi_{i,j} = \phi_{j,i}$

Reordering the equations above we get:

$$\begin{bmatrix} 0 & 0 & 0 & x_1 & .. & x_N & x_{N+1} \\ 0 & 0 & 0 & y_1 & .. & y_N & y_{N+1} \\ 0 & 0 & 0 & 1 & .. & 1 & 1 \\ x_1 & y_1 & 1 & \phi_{1,1} & .. & \phi_{1,N} & \phi_{1,N+1} \\ : & : & : & : & & : & : \\ x_N & y_N & 1 & \phi_{N,1} & .. & \phi_{N,N} & \phi_{N,N+1} \\ x_{N+1} & y_{N+1} & 1 & \phi_{N+1,1} & .. & \phi_{N+1,N} & \phi_{N+1,N+1} \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_0 \\ \lambda_1 \\ : \\ \lambda_N \\ \lambda_{N+1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ f_1 \\ : \\ f_N \\ f_{N+1} \end{bmatrix}$$

We can see that last row and last column is "inserted". As RBF functions are symmetrical the recently derived formula for iterative computation of the inverse function can be used. So the RBF interpolation is given by the matrix $\boldsymbol{M}$ as

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{b}^T & c \end{bmatrix}$$

where the matrix $\boldsymbol{A}$ is the RBF matrix $(N+3) \times (N+3)$ and the vector $\boldsymbol{b}$ $(N+3)$ and scalar value c are defined as:

$$\boldsymbol{b} = [x_{N+1} \quad y_{N+1} \quad 1 \quad \phi_{1,N+1} \quad .. \quad \phi_{N,N+1}]^T$$

$$c = \phi_{N+1,N+1}$$

It means that we know how to compute the matrix $\boldsymbol{M}^{-1}$ if the matrix $\boldsymbol{A}^{-1}$ is known.

That is exactly what we wanted!

Recently we have proved that iterative computation of inverse function is of $O(N^2)$complexity, that offers a significant performance improvement for points insertion. It should be noted that some operations can be implemented more effectively, especially $\boldsymbol{\xi}\otimes\boldsymbol{\xi}^{\boldsymbol{T}} = \boldsymbol{A}^{-1}\boldsymbol{b}\boldsymbol{b}^T\boldsymbol{A}^{-1}$ as the matrix $\boldsymbol{A}^{-1}$ is symmetrical etc.

2.2 Point Removal

In some cases it is necessary to remove a point from the given data set. It is actually an inverse operation to the insertion operation described above. Let us consider a matrix $\boldsymbol{M}$ of the size $(N+1) \times (N+1)$ as

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{b}^T & c \end{bmatrix}$$

Now, the inverse matrix $\boldsymbol{M}^{-1}$ is known and we want to compute matrix $\boldsymbol{A}^{-1}$, which is of the size $N \times N$.

Recently we derived opposite rule:

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{b}^T & c \end{bmatrix}$$

$$\boldsymbol{\xi} = \boldsymbol{A}^{-1}\boldsymbol{b} \qquad k = c - \boldsymbol{\xi}^{\boldsymbol{T}}\boldsymbol{b}$$

$$\boldsymbol{M}^{-1} = \begin{bmatrix} \boldsymbol{A}^{-1} + \frac{1}{k}\boldsymbol{\xi}\otimes\boldsymbol{\xi}^{\boldsymbol{T}} & -\frac{1}{k}\boldsymbol{\xi} \\ -\frac{1}{k}\boldsymbol{\xi}^{\boldsymbol{T}} & \frac{1}{k} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}_{11} & \boldsymbol{Q}_{12} \\ \boldsymbol{Q}_{21} & \boldsymbol{Q}_{22} \end{bmatrix}$$

It can be seen that

$$\boldsymbol{Q}_{11} = \boldsymbol{A}^{-1} + \frac{1}{k}\boldsymbol{\xi}\otimes\boldsymbol{\xi}^{\boldsymbol{T}}$$

and therefore

$$\boldsymbol{A}^{-1} = \boldsymbol{Q}_{11} - \frac{1}{k}\boldsymbol{\xi}\otimes\boldsymbol{\xi}^{\boldsymbol{T}}$$

Now we have both operations, i.e. insertion and removal, with effective computation of $O(N^2)$ computational complexity instead of $O(N^3)$. It should be noted that vectors related to the point assigned for a removal must be in the last row and last column of the matrix $\boldsymbol{M}^{-1}$.

2.3 Point selection

As the number of points within the given data set could be high, the point removal might be driven by a requirement of removing a point which causes a *minimal error of the interpolation*. This is a tricky requirement as there is probably no general answer. The requirement should include additional information which interval of $\boldsymbol{x}$ is to be considered.

Generally we have a function

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \lambda_i\, \phi_i(\boldsymbol{x}) + P_k(\boldsymbol{x})$$

and we want to remove a point $\boldsymbol{x}_j$ which causes a minimal error ε_j of interpolation, i.e.

$$f_j(\boldsymbol{x}) = \sum_{i=1, i\neq j}^{N} \lambda_i\, \phi_i(\boldsymbol{x}) + P_k(\boldsymbol{x})$$

and we want to minimize

$$\varepsilon_j = \int_{\Omega} \left| f(\boldsymbol{x}) - f_j(\boldsymbol{x}) \right| d\boldsymbol{x}$$

where *Ω* is the interval on which the interpolation is to be made. It means that if the point $\boldsymbol{x}_j$ is removed the error ε_j is determined as:

$$\varepsilon_j = \lambda_j \int_{\Omega} \phi(\|\boldsymbol{x} - \boldsymbol{x}_j\|) d\boldsymbol{x}$$

As we know the interval *Ω* on which the interpolation is to be used, we can compute or estimate the error ε_j for each point $\boldsymbol{x}_j$ in the given data set and select the best one. For many functions ϕ the error ε_j can be computed or estimated analytically as the evaluation of ε_j is simple for many functions, e.g.

$$\int r^m \ln dr = r^{m+1} \frac{\ln r}{m+1} - \frac{1}{(m+1)^2}$$

It means that for TPS function $r^2 \ln r$ the error ε_k is easy to evaluate. In the case of CSRBF the estimation is even simpler as they have a limited influence, so generally λ_j determines the error ε_j.

It should be noted, that a selection of a point with the lowest influence to the interpolation precision in the given interval *Ω* is of *O(N)* complexity only.

We have shown a novel approach to RBF computation which is convenient for larger data sets. It is especially convenient for t-varying data and for applications, where a "sliding window" is used. Basic operations – point insertion and point removal – have been introduced. These operations have *O(N^2)* computational complexity only, which makes a significant difference from the original approach used for RBFs computation.

3. Conclusion

The proposed *Progressive RBF Interpolation* method has advantages over the standard techniques based RBF interpolation due to possibility to insert / remove points with decreased computational complexity from *O(N^3)* to *O(N^2)*. This enables to apply this approach in applications when interpolation or rendering of data in a "sliding window" and / or t-varying interpolation data are required; in applications when some data are becoming invalid and new data are acquired and need to be included into the interpolated data set. Due to lower computational complexity it is also possible to handle data sets in which scalar values associated with t-varying points, i.e. it is possible to handle non-static data as well.

It is expected that the presented approach can lead to development of new algorithms especially in surface reconstruction of 3D objects. As the proposed *Progressive RBF Interpolation* uses vector / matrix operations exclusively it is suitable for GPU / Larabee architectures as well.

Future work will be devoted to development of methods minimizing the error of interpolation with maximization of number of points removed from the dataset. This should lead to data compression techniques based on RBF representation.

4. Acknowledgments

The author thanks to colleagues at the University of West Bohemia for their critical comments, suggestions, to anonymous reviewers for their critical view and comments that improved the manuscript significantly. Special thanks belong to former PhD and MSc. students, especially to Vit Ondracka, for their help with implementations and recent experiments made.

The research was supported by the Ministry of Education of the Czech Republic, project LC-CPG No. LC06008.

5. References

Baxter,B.J.Ch: The Interpolation Theory of Radial Basis Functions, PhD thesis, Trinity College, Cambridge University, U.K., 1992

Bertalmio, M., Sapiro, G., Ballester, C. and V. Caselles, V.: Image Inpainting, Proceedings of SIGGRAPH'00, Computer Graphics, New Orleans, 23-28 July 2000, pp.417-424.

Carr, J. C., Beatson, R. K., Cherrie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B. C., Evans, T. R.: "Reconstruction and representation of 3D objects with radial basis functions", Computer Graphics (SIGGRAPH 2001 proceedings), pp. 67-76, August 2001.

Duchon, J.: "Splines minimizing rotation-invariant semi-norms in Sobolev space", Constructive Theory of Functions of Several Variables, Springer Lecture Notes in Math, 21 (1977), pp. 85-100.

Hardy, L.R.: Multiquadric equations of topography and other irregular surfaces, J. Geophy. Res., 76 (1971), pp. 1905-1915.

Micchelli, C.A.: "Interpolation of scattered data: distance matrice and conditionally positive definite functions", Constr. Approx., 2 (1986), pp. 11-22.

Pan,R.J., Skala,V.: Implicit surface modeling suitable for inside/outside tests with radial basis funtions, 10th International Conference on Computer Aided Design and Computer Graphics (CAD/Graphics 2007)

Schagen, I.P.: Interpolation in Two Dimension – A New Technique, J. Inst. Maths Applics, 23 (1979), pp. 53-59.

Uhlir,K., Skala,V.: Radial basis function use for the restoration of damaged images. *In* Computer vision and graphics. Dordrecht: Springer , 2006. s. 839-844. ISBN 1-4020-4178-0

Wang,Ch.C.L., Kwok,T.-H.: Interactive Image Inpainting using DCT Based Expemplar Matching, ISVC 2009, LNCS 5876, pp.709-718, 2009

Wendland, H.: Computational aspects of radial basis function approximation, in K. Jetter et al. (eds.) *Topics in Multivariate Approximation and Interpolation* , Elsevier B.V., pp. 231-256, 2005.

Wright, G.B., "Radial Basis Function Interpolation: Numerical and Analytical Developments", University of Colorado, Thesis, 2003.

Zapletal,J., Vanecek,P., Skala,V.: RBF-based Image Restoration Utilising Auxiliary Points, CGI 2009 proceedings, ACM ISBN 978-60558-687-8, pp.39-44, 2009.

Ohtake,Y., Belyaev,A., Seidel,H.-P.: 3D Scattered Data Interpolation and Approximation with Multilevel Compactry Supported RBFs, Graphical Models, Vol.67, No.3., pp.150-165, 2005.

FastRBF: http://www.farfieldtechnology.com/