

IMAGE INTERPOLATION USING ADAPTIVE LINEAR FUNCTIONS AND DOMAINS

D. Darian Muresan and Thomas W. Parks

School of Electrical and Computer Engineering
Cornell University, Ithaca NY. 14853
darian, parks@ee.cornell.edu

ABSTRACT

In this paper we present a generalized framework to image interpolation and suggest a few approaches on how interpolation can be learned using an adaptive linear model. We'll also discuss how this approach can be applied to other more sophisticated learning methods.

1. INTRODUCTION

Image interpolation is a very interesting theoretical problem. Its solution is important not only from a theoretical point of view, but also from a consumer's market view.

With the introduction of digital cameras into the consumer market, the need for creating high resolution prints from 640 by 480 digital images brings interpolation in direct contact with the consumer market. Digital images need to be first interpolated to a higher sampling rate before being printed on high resolution printers.

2. NOTATION

Definition 1 *An image is a finite set of real n-tuples*

$$I = \{(\bar{x}_i, y_i)\},$$

such that for each i , \bar{x}_i is a real $(n-1)$ -tuple, and y_i is a real number. The size of the image is the size of set I and it is denoted by $\|I\|$.

We denote by $I_{\downarrow M} = \{(\bar{\alpha}_i, \beta_i)\}$ any image which is M times smaller than image I . Usually, $I_{\downarrow M}$ is obtained by decimating I M times, or by first filtering I with a low pass filter and then decimating M times. Image $I_{\downarrow M}$ is referred to as the low density image, while image I is referred to as the high density image. We will use Greek letters for all the n-tuples belonging to a low density

image and Roman letters for all n-tuples belonging to a high resolution image.

Definition 2 *Given a pair of images (low density, high density) $(I_{\downarrow M}, I)$, **interpolation** is an invertible function $\mathcal{F}_{I_{\downarrow M}}^I$ (linear or otherwise) such that*

$$\begin{aligned} \mathcal{F}_{I_{\downarrow M}}^I(I_{\downarrow M}) &\simeq I & (1) \\ \mathcal{F}_{I_{\downarrow M}}^I(\{(\bar{\alpha}_i, \beta_i)\}) &= \{(\hat{x}_i, \hat{y}_i)\} \simeq \{(\bar{x}_i, y_i)\} & (2) \end{aligned}$$

The symbol \simeq means approximately equal. Two finite sets S and T are approximately equal if they have the same size. They are equal if the sets have the exact same elements. The notation $\mathcal{F}_{I_{\downarrow M}}^I$ indicates that we are trying to obtain image I from $I_{\downarrow M}$. If we had the pair $(I_{\downarrow M}, J)$ the interpolation function would be denoted as $\mathcal{F}_{I_{\downarrow M}}^J$.

3. IMAGE ENLARGEMENT

In doing image enlargement there are three main problems that need to be solved.

1. What distance metric do we use? What do we mean by the distance between two images I_1 and I_2

$$d(I_1, I_2)?$$

An OK measure is to assume that the distance is simply a mean squared error between the two images, but this does not take at all into account the human visual system.

2. Given the (low density, high density) pair(s) $(I_{\downarrow M}, I)$ how do we find $\mathcal{F}_{I_{\downarrow M}}^I$. This problem can be thought of as a training or data fitting problem. One approach is to assume some type of structure on $\mathcal{F}_{I_{\downarrow M}}^I$ and then find the $\mathcal{F}_{I_{\downarrow M}}^I$ with the chosen structure, that best fits the data.

3. Given only $I_{\downarrow M}$, how do we find $\mathcal{F}_{I_{\downarrow M}}^I$? In enlarging image $I_{\downarrow M}$, we need to calculate $\mathcal{F}_{I_{\downarrow M}}^I(I_{\downarrow M})$. While the evaluation of $\mathcal{F}_{I_{\downarrow M}}^I$ is usually straight forward, the problem is that $\mathcal{F}_{I_{\downarrow M}}^I$ is usually not available to us. To solve this problem, there are three plausible solutions

- (a) Let $\mathcal{F}_{I_{\downarrow M}}^I$ be non-trained. Removing the requirement that $\mathcal{F}_{I_{\downarrow M}}^I$ is trained takes away the learning part and thus all we need in determining $\mathcal{F}_{I_{\downarrow M}}^I$ is $I_{\downarrow M}$. Convolution based interpolations such as linear and cubic are not trained.
- (b) Let the (low density, high density) pair be $(I_{\downarrow M^2}, I_{\downarrow M})$. Assuming that $\mathcal{F}_{I_{\downarrow M^2}}^{I_{\downarrow M}}$ and $\mathcal{F}_{I_{\downarrow M}}^I$ are pretty similar, find $\mathcal{F}_{I_{\downarrow M^2}}^{I_{\downarrow M}}$ using one of the methods from step (2) above and let $\mathcal{F}_{I_{\downarrow M}}^I = \mathcal{F}_{I_{\downarrow M^2}}^{I_{\downarrow M}}$.
- (c) Obtain the (low density, high density) training pairs of images by using other pairs of (low density, high density) obtained from a database of images. The lookup in the image database should be based on $I_{\downarrow M}$. Assuming that $\mathcal{F}_{I_{\downarrow M^2}}^{I_{\downarrow M}}$ and $\mathcal{F}_{I_{\downarrow M}}^I$ are pretty similar, find $\mathcal{F}_{I_{\downarrow M^2}}^{I_{\downarrow M}}$ using one of the methods in step (2) above and let $\mathcal{F}_{I_{\downarrow M}}^I = \mathcal{F}_{I_{\downarrow M^2}}^{I_{\downarrow M}}$. One difficulty with this method is searching the database. An exhaustive search is usually extremely slow, especially for a large image database.

In the remainder of this paper we look at how to solve problem (2). Once that problem is solved, interpolation is done by learning $\mathcal{F}_{I_{\downarrow M}}^I$ and applying it to the given image. We let the (low density, high density) pair of images be (decimated given image, given image) and train for $\mathcal{F}_{I_{\downarrow M^2}}^{I_{\downarrow M}}$ using the solution of problem (2). Finally, we assume that $\mathcal{F}_{I_{\downarrow M}}^I = \mathcal{F}_{I_{\downarrow M^2}}^{I_{\downarrow M}}$.

4. LEARNING $\mathcal{F}_{I_{\downarrow M}}^I$ FROM THE PAIR(S) OF $(I_{\downarrow M}, I)$

The interpolating function $\mathcal{F}_{I_{\downarrow M}}^I$ takes as input set $I_{\downarrow M} = \{(\bar{\alpha}_i, \beta_i)\}$ and we would like it to return $\{(\bar{x}_i, y_i)\}$ which is M times larger in size. This suggests two possible general characteristic about the behavior of function $\mathcal{F}_{I_{\downarrow M}}^I$:

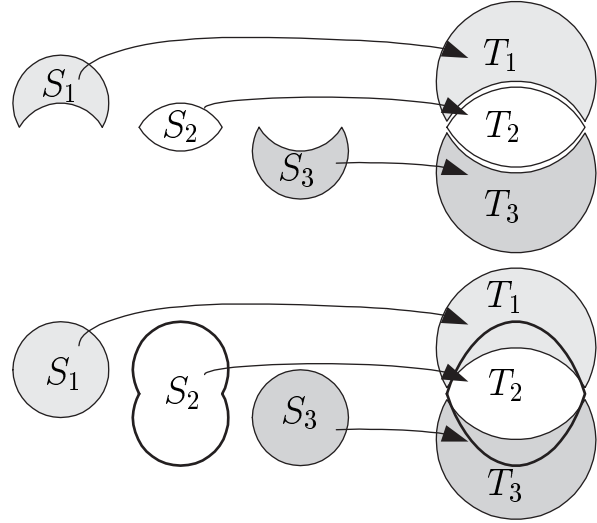


Figure 1: No Overlap One to Many $\mathcal{F}_{I_{\downarrow M}}^I$ behavior (left) and Overlap Many to Many $\mathcal{F}_{I_{\downarrow M}}^I$ behavior (right). Notice that in both cases $\bigcup_i S_i = I_{\downarrow M}$ and $\bigcup_i T_i = I$

1. No Overlap One to Many. This means that we divide the input set $I_{\downarrow M}$ into non overlapping sets S_i . For a given subset S_i of $I_{\downarrow M}$ the function $\mathcal{F}_{I_{\downarrow M}}^I$ returns a subset T_i of I , which is M times larger than S_i . This behavior is depicted in Fig. 1 (left).
2. Overlap Many to Many. In this case, we allow overlap when we divide the input set $I_{\downarrow M}$ into smaller subsets S_i . For a given subset S_i of $I_{\downarrow M}$ the function $\mathcal{F}_{I_{\downarrow M}}^I$ returns a subset T_i of I , which can now be any size we want, even smaller than the size of S_i . We also allow subsets T_i to overlap and we add the restriction that

$$\bigcup_i S_i = I_{\downarrow M} \text{ and } \bigcup_i T_i = I$$

This behavior is depicted in Fig. 1 (right).

In both cases mentioned above, all we need to discuss is how we learn $\mathcal{F}_{I_{\downarrow M}}^I$ for one specific S_i to T_i mapping

$$\mathcal{F}_{I_{\downarrow M}}^I : S_i \rightarrow T_i$$

To generalize the action of $\mathcal{F}_{I_{\downarrow M}}^I$ over the entire set $I_{\downarrow M}$, all we do is learn $\mathcal{F}_{I_{\downarrow M}}^I$ for all the different S_i subsets of $I_{\downarrow M}$.

Assume that $\mathcal{F}_{I_{\downarrow M}}^I$ is an overlap many to many function. How subsets S_i and T_i are chosen is a difficult problem

Input (S_i)	Output (T_i)
$(\bar{\alpha}_1, \beta_1), (\bar{\alpha}_2, \beta_2)$	(\bar{x}_1, y_1)
$(\bar{\alpha}_3, \beta_3), (\bar{\alpha}_4, \beta_4)$	(\bar{x}_2, y_2)
$(\bar{\alpha}_1, \beta_1), (\bar{\alpha}_3, \beta_3)$	(\bar{x}_3, y_3)
\vdots	\vdots

Table 1: Example of a mapping, which we would like to use as training data.

in itself. We will discuss a few possible solutions in section (5), but for now let's assume that sets S_i and T_i are given. More specifically, we know how n-tuples in S_i are mapped into n-tuples in T_i .

4.1. Metric Methods

If we had at our existence a metric that could correctly measure the distance between two images then it would be feasible that given a table mapping, we would want to find a continuous function that would minimize the error over the entire table mapping. Given set S_i and T_i for an image, a table mapping might look like Table 1. We want to find a function that will take in the three n-tuples from the left and return the n-tuple from the right. Equivalently, for the y entry in the n-tuple, we want to find a function \mathcal{F} such that

$$\begin{aligned} \mathcal{F}(\beta_1, \beta_2) &= y_1 \\ \mathcal{F}(\beta_3, \beta_4) &= y_2 \\ \mathcal{F}(\beta_1, \beta_3) &= y_3 \\ &\vdots = \vdots \end{aligned}$$

Additionally we need to have a function \mathcal{G} that would tell us how $\bar{\alpha}$ gets mapped to \bar{x} . Then, $\mathcal{F}_{I \downarrow M}^I = (\mathcal{G}, \mathcal{F})$. The way we take care of \mathcal{G} is to fix the same mapping over the entire training set. For example, we could say that any pixel is always estimated by the closest left and right pixels. Then this defines our \mathcal{G} . Next, let's focus on the mapping \mathcal{F} .

4.1.1. Least Squares and Support Vector Regression

One solution to finding such an \mathcal{F} would be to use least squares, as done in [5]. If we assumed that y_i is a linear combination of β_i and β_j , solving for \mathcal{F} via least squares

is the same as finding $\bar{w} = [w_1, w_2]$ in

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} \beta_1 & \beta_2 \\ \beta_3 & \beta_4 \\ \beta_1 & \beta_3 \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \quad (3)$$

Support vector regression (SVR) is another method of solving this problem. SVR tries to find a function of the form

$$\mathcal{F}(\bar{\beta}_i) = \text{ker}(\bar{w}, \bar{\beta}_i) + b, \text{ with } b \in \mathfrak{R}$$

Where the kernel $\text{ker}(\bar{w}, \bar{\beta})$ specifies the type of regression we are dealing with. Special cases are radial basis function (RBF) nets and polynomial regression [6]. It turns out that the SVR problem can be solved via least squares when the input dimension is fairly small. The solutions of SVR, as presented in [6] are relevant to high dimensional problems where least squares is not a feasible solution. Our problem is usually of small dimensions and thus least squares will do.

To see how SVR reduces to least squares, let's look at a simple example with a linear kernel. In this case, the SVR problem is

$$\begin{aligned} y_i = \mathcal{F}(\bar{\beta}_i) &= \langle \bar{w}, \bar{\beta} \rangle + b \\ &= [w_1 \quad w_2] \begin{bmatrix} \beta_i \\ \beta_j \end{bmatrix} + b \end{aligned}$$

Which can be solved for \bar{w} and b by solving the following least squares problem

$$\begin{bmatrix} \mathcal{F}(\bar{\beta}_1) \\ \vdots \\ \mathcal{F}(\bar{\beta}_i) \end{bmatrix} = \begin{bmatrix} \beta_{1,1} & \beta_{1,2} & 1 \\ \vdots & \vdots & \vdots \\ \beta_{i,1} & \beta_{i,2} & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix}$$

The least square problem can be also viewed as an optimal recovery problem, as we have shown in [3, 1]. Viewing the problem as an optimal recovery problem helps if we had additional linear information about how the data within the set S might be correlated.

An obvious extension of direct least squares is to assume that the data is linear only on certain domains. The difficulty is finding these domains. For this, we could use a modified method of the one proposed in [8, 7] for curve splitting.

Besides least squares we could use other data fitting models, such as those presented in [4].

5. CHOOSING S_I , T_I AND $S_I \mapsto T_I$

The efficiency of our interpolating function $\mathcal{F}_{I \downarrow M}^I$ depends on two things. First, it depends on how well we

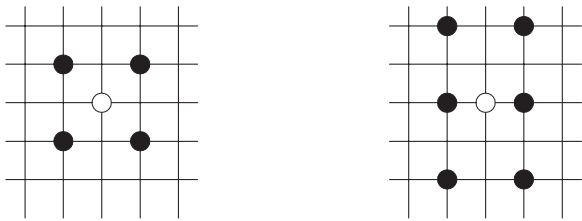


Figure 2: Black dots are the locations of β_i and white is the location of y_1 . The configuration on the left introduces wavy artifacts in textured regions, while the one on the right does not.

can train our function $\mathcal{F}_{I_{\downarrow M}}^I$ from the table mapping. Second, it depends on how our table mapping represents our (low density, high density) pair of images. In other words, we do not want to take n-tuples representing an edge in $I_{\downarrow M}$ and map them to n-tuples representing a uniform region in I . Clearly, the table mapping must be local. In this section we look at locality and other factors that measure how well the table mapping represents the $(I_{\downarrow M}, I)$ pair.

For simplicity, assume that we are trying to find $\mathcal{F}(\beta_1, \dots, \beta_4) = y_1$. How do we build our mapping that is to be used in the training of \mathcal{F} ? In this paper we have looked at two approaches in building our training table mapping.

The first factor is based on the location of β_1, \dots, β_4 with respect to y_1 . Do we chose y_1 in the center of β_i or do we chose it to the side? Experimentally we have noticed that if the β_i values are taken symmetrically around y_i (Fig. 2) the image tends to be overly smooth. Texture regions in particular become very wavy, giving a water flow feeling to the image. Biasing the direction of the β_i neighborhood in one direction (Fig. 2) seems to remove the wavy artifacts introduced in textured regions, while maintaining good interpolation around edges.

Related to the location of β_i , an interesting extension is to assume that instead of y_i being some linear function of $\bar{\beta}$, the entire vector \bar{y} is a linear function of $\bar{\beta}$. This is the approach of [2]. However, their training set was the entire image, which does not seem to work as well as if it is more localized.

A second decision in building our mapping table is made after the location of β_i has been fixed. After initially generating a training set, we prune this set by throwing out the entries which have $(\hat{\beta}_1, \dots, \hat{\beta}_4)$ far from $(\beta_1, \dots, \beta_4)$.

6. CONCLUSION

In this paper we presented a learning based framework to image interpolation. Our results seem to suggest that this a feasible approach and good results can be obtained using this method. We encourage the reader to visit our web site (www.ece.cornell.edu/~splab.) to view all the tif image results.

7. REFERENCES

- [1] D. D. Muresan and T. W. Parks, "Optimal Recovery Approach to Image Interpolation," *ICIP*, Greece 2001.
- [2] T. Chen, H. R. Wu, B. Qiu, "Image Interpolation using Across-Scale Pixel Correlation," *ICASSP*, Salt Lake City, Utah 2001.
- [3] D. D. Muresan and T. W. Parks, "Adaptive, Optimal-Recovery Image Interpolation," *ICASSP*, Salt Lake City, Utah 2001.
- [4] Richard O. Duda, Peter E. Hart, and David G. Stork, "Pattern Classification," *John Wiley and Sons Inc.*, Second Edition, 2001.
- [5] Xin Li and Michael T. Orchard, "New Edge Directed Interpolation," *IEEE International Conference on Image Processing*, Vancouver, Sept. 2000.
- [6] Alex J. Smola and Bernhard Scholkopf, "A tutorial on Support Vector Regression," *NeuroCOLT2 Technical Report Series*, pp. 1-73, October, 1998.
- [7] McMaster, R. B., "A Statistical Analysis of Mathematical Measures for Linear Simplification," *American Cartographer*, 13, pp. 103-116, 1986.
- [8] Ramer U., "An Alternative Procedure for the Polygonal Approximation of Plane Curves," *Computer Graphics and Image Processing*, 1, pp. 244-256, 1972.