Abstract

Hyper-Resolution, a new technique for super-resolution reconstruction of images, is based on matching low-resolution target image details to their high-resolution counterparts from an image database. Central to the algorithm is a novel transform of image content from the orthogonal pixel space to a parametric space structured around edges. This approach offers improved quality, more flexibility and significantly faster performance than previous work in the field. Implementation strategies for achieving this efficiency are carefully outlined. The algorithm is evaluated by controlled assessment, qualitative evaluation, and applications to facial detail reconstruction and identification. The algorithm is finally analyzed through the comparison with alternative techniques.

Keywords: Computer graphics methodologies; Fundamentals; Image processing and computer vision; Applications; Enhancement; Reconstruction

1. Introduction

In this age of high-resolution imaging, researchers constantly strive for more pixels well beyond the limits of current image capture technology. As such, techniques for accurately and efficiently reconstructing high-resolution image details are in great demand. Numerous applications benefit from these techniques, including photographic enlargement, scalable texture mapping (image-based rendering), forensics and surveillance imaging, restoration of fine artwork, diagnosis via higher-resolution medical images, and video processing with high-definition TV reconstruction.

Research on resolution enhancement has been conducted for over two decades [1]. The most traditional method, interpolation [2,3], estimates missing data between sampled pixels by using the surrounding pixels. Unfortunately, interpolation results in blurred images lacking distinct edges and fine textures. While many improvements on interpolation have been proposed [4,5], it is still mathematically impossible for any interpolation algorithm to reconstruct subpixel detail above the spatial frequency of the original image. This fundamental problem has given rise to a number of so-called super-resolution techniques [6–10]: if one “target” image does not contain enough information to reconstruct the desired level of detail, we can use multiple lower-resolution “source” images of the same or similar subjects (captured from different perspectives) to reconstruct the missing data points in a single high-resolution frame.

Registration-based super-resolution relies on multiple images captured by a precisely moving camera. Based on the camera’s motion vector, we can accurately solve for missing samples at subpixel intervals to reconstruct a super-resolved image [1,6,8–11]. Recently, training-based
super-resolution algorithms have been explored in [12–18]. These techniques rely on the premise that we can reconstruct visually plausible high-resolution detail for one low-resolution target image based on the pattern recognition of “similar” details in a potentially large database of high-resolution source images. The most closely matching details and textures from the source images (typically organized as a set of small image patches) are then substituted in place of their low-resolution counterparts in the target image to create a super-resolution image.

In this area, Hertzmann et al. [15] developed a training-based approach called “image analogies”, where fine details are learned and predicted between images. Baker and Kanade [14] devised a so-called reconstruction (recognition based reconstruction) algorithm in which local features in the low-resolution target image are recognized and then enhanced using parent structure vector matching. This method produced impressive results on human faces, but still lacks robustness in certain illumination conditions. Most of training-based techniques require a specific model adapted to the subject material of the target images and cannot be used on arbitrary images. Example-based super-resolution, developed by Freeman et al. [12], gets around this problem by using a database of small pixel patches from many training images to reconstruct plausible high-frequency detail in enlarged images; the matching is performed by correlating these small patches (e.g., 7 × 7 pixels). Most recently, an image hallucination approach by using primal sketch priors was developed by Sun et al. [19]. The image resolution is enhanced to a very high quality. This method relies on the good match of the primal sketch priors between target image and source image, its robustness is a major concern for further applications. Motivated by the existing work mentioned above, we have developed a new technique to attack this problem.

1.1. Introducing Hyper-Resolution

Image sharpness and detail are often judged by the quality of an image’s strong edges. However, most existing super-resolution algorithms work on either the pixel or pixel patch level. This raises the obvious question: if image detail is in edges by definition, why match in the pixel domain to begin with? This observation leads us to a novel idea: instead of matching patches in the spatial domain [12], we can first transform each image into a new parametric vector space structured by the image’s edges. Our detail “patches” are then composed of only the texture details sampled on and around image edges, with coordinates relative to these edges. Note that several most recent work reported in [20–22] show that it is very promising to use feature-based texture or silhouette map to increase the texture rendering quality. The impressive results were presented in the reported work, however, only the simple-texture images with salient object contours were tested. We propose a new algorithm to trace the image contents around edges using the texture matching approach so that more complex image details are expected to be reconstructed.

![HyperRes pipeline overview](image)

**Fig. 1.** HyperRes pipeline overview. HyperRes is split into a number of discrete stages for both conceptual and performance reasons. To emphasize the extensive parallelism inherent in the algorithm, stacked blocks are shown for parallelized stages.
This proposed approach is called Hyper-Resolution, or HyperRes. In our algorithm, edge detection is used to obtain the edges of all input images. After fitting each edge to a parametric curve representation, we use a coordinate system transform to sample points along and normal to the edge, thus forming a “parametric map”. Each parametric map, which is invariant across affine transforms of its corresponding edge, is decomposed into a hierarchy of smaller segments; these segments are then entered as keys in a database. To reconstruct the high-resolution version of a given target edge, we locate similar edge segments in the database, project the matching source edge data onto the curvature of the appropriate target edges, and add this high-resolution data back into the original low-resolution target image. While edge structured imaging has been described before [20–24], our technique differs in that an image is described as the textures surrounding the edges rather than as a superposition of contour edges themselves.

The HyperRes process, as shown in Fig. 1, is logically split into two parts. The assimilation phase transforms both source and target images to their parametric edge representations and adds them to a database, while the reconstruction phase maps appropriate high-resolution edge details from the database onto the target image to reconstruct it at a higher resolution than its original pixels provided. The HyperRes algorithm is detailed in Section 2. After we provide experimental results and assessments in Section 3, we will discuss its applications to enhancing face feature detection and identification. Our algorithm’s advantages and weaknesses are analyzed in Section 4. Finally, concluding remarks and future work are given in Section 5.

2. Hyper-Resolution algorithm

2.1. Target preparation

Prior to starting the true HyperRes algorithm, the target image must be interpolated up to the desired output size (for instance, by 200% or 400%). By doing so, we reduce the Hyper-Resolution problem to one of simply matching the equivalent of Gaussian blurred lowpass target data to high-resolution source data. HyperRes uses the Gaussian convolution interpolation for this purpose, to avoid jagged edge artifacts.

HyperRes does most pixel operations on highpass filtered versions of the original source and target images, just as in [12]. This is because we are only interested in analyzing and matching regions of high spatial detail. Low-frequency components are not relevant since they contribute little to the perceived or quantitative sharpness. Highpass filtering an input image in the highpass stage results in a working image, on which all further processing is done.

2.2. Edge detection and processing

To structure an image around its edges, we must first detect those edges. Typically the image is smoothed prior to edge detection to reduce the impact of noise. In the MapPrep stage (see Fig. 1), we use the Canny edge detection [25] with the non-maximal suppression to derive a bitmap of image pixels on strong edges (as defined by the intensity gradient of the image). Connected edge pixels are then recognized and grouped into sorted lists of pixel coordinates in the MapTrace stage. To ensure high performance, our edge tracing algorithm divides the binary edge pixel map into a number of small tiles (e.g., \(32 \times 32\) pixels) and processes each tile in parallel. This technique is unique in that the tiles are small enough to obtain effectively constant time performance through the use of boolean logic and lookup tables directly on the 32-bit machine words storing the bitmap rows. To process a given tile, the MapTrace stage starts with an arbitrary edge pixel and follows the chain of neighboring pixels as far as possible until either the edge terminates, a tile boundary is crossed or a junction between several edges is reached.

Since HyperRes assumes that edges are only split, where they intersect with other edges (i.e., in regions of high detail), we identify closely spaced pairs of edge endpoints and merge these edges into a single longer edge in the MapMerge stage. Edges are typically split by image artifacts, noise and tile boundaries. Our implementation uses a geometric hash table-based algorithm to quickly locate adjacent endpoints.

2.3. Transform

The central principle of operation in HyperRes revolves around the idea of capturing detail on and around image edges. Therefore, we must establish a bidirectional mapping between two coordinate spaces: the orthogonal space in which the image pixels reside along \(x, y\) coordinates, and the parametric space, a non-linearly transformed space relative to the curvature of each edge. Each edge has its own local parametric space in the form of a warped rectangle (as shown in Fig. 3), with the \(u\) coordinate running parallel to the edge and the \(v\) coordinate running out along the edge’s normal as evaluated at \(u\). The \(u\) coordinate ranges from 0 to the length of the original edge in image pixels. This length is found via chord length parameterization [26] of the original edge point set.

A mathematically continuous representation of each edge is critical to this mapping. Any uneven spacing between points or other discontinuities in an edge will result in gaps in the domain of the parametric space. Unfortunately, edge gaps and other artifacts are fairly common due to the earlier merging of edges which may be separated by a few pixels. Therefore, we apply a
localized cubic spline-based algorithm to each list of raw pixel coordinates to obtain coefficients defining a parametric curve. This allows us to interpolate the \( x, y \) coordinates of any point in the spans between known points, as well as to easily find the normal vector to the edge at a given point (Fig. 2 shows one example of the eye edge image with superimposed parametric curves (i.e., splines)). A pre-processing stage is applied to reduce the number of points by replacing a set of several nearby points with the vector median of their coordinates; this also reduces noise and rounding effects. Thus, a smooth transform \( u, v \) parametric coordinates to \( x, y \) orthogonal coordinates is realized.

We restrict the \( u \) coordinate to the uniform range \([0, 1]\) such that it represents the normalized distance along the edge length we are examining; otherwise it would be impossible to compare similar edges on different scales later on. Therefore, we first use chord length parameterization [26] to associate the desired value of \( u \) with each edge point. The total length of the curve is first estimated via the \( N \) known points \( E_n \) as

\[
L = \sum_{n=0}^{N-1} |E_{n-1} - E_n|.
\]  
(1)

From this, we assign

\[
U_n = U_{n-1} + \frac{1}{L} |E_{n-1} - E_n|,
\]  
(2)

where \( U_0 = 0 \) and \( U_{N-1} = 1 \). Therefore, given an arbitrary value of \( u \) in the range \([0, 1]\), a simple binary search may be used to obtain the point number \( n \) at the start of the surrounding span and the base \( U_n \) of that span.

We chose to use a Gaussian cubic spline interpolation matrix instead of the common Catmull–Rom [3] matrix so as to minimize the effects of noise on edge formation. This approach allows us to derive the coordinates of an arbitrary point \( \hat{P}(u) \) at a parametric coordinate \( u \) in the span between any two known edge points \( \hat{E}_n \) and \( \hat{E}_{n+1} \) (which can be found by a binary search on the \( u \) coordinate). In Eq. (3), this span is defined as starting at \( u = U_n \) with a span length of \( S_n = |\hat{E}_{n+1} - \hat{E}_n| \). The complete transform is thus

\[
\hat{P}(u) = \begin{bmatrix}
\frac{1}{6} & \frac{1}{3} & \frac{1}{2} & 0 \\
\frac{1}{6} & \frac{1}{2} & 0 & 0 \\
-1 & -2 & 3 & 0 \\
\end{bmatrix}
\begin{bmatrix}
\frac{\hat{E}_{n-1}}{S_n} \\
\frac{\hat{E}_n}{S_n} \\
\frac{\hat{E}_{n+1}}{S_n} \\
\frac{\hat{E}_{n+2}}{S_n} \\
\end{bmatrix}.
\]  
(3)

To completely define the parametric space, we must also find the \( v \) coordinate, which runs parallel to the curve’s normal. The complete transform from a parametric coordinate \( u, v \) to an orthogonal coordinate \( \hat{Q}(u, v) \) is thus

\[
\hat{Q}(u, v) = \hat{P}(u) + v \frac{\partial \hat{P}(u)}{\partial u}.
\]  
(4)

where \(-v_s \leq v \leq +v_s\) (\( v_s \) is set as 8 pixels). \( \hat{Q}(u, v) = [Q_x(u, v), Q_y(u, v)] \) is the final coordinate in the orthogonal space, \( \hat{P}(u) = [P_x(u), P_y(u)] \) and

\[
\frac{\partial \hat{P}(u)}{\partial u} = \left[ \begin{array}{c}
\frac{\partial P_x(u)}{\partial u} \\
\frac{\partial P_y(u)}{\partial u}
\end{array} \right].
\]

\[
\frac{\partial \hat{P}(u) / \partial u \cdot \partial \hat{P}(u) / \partial v}{\partial \hat{P}(u) / \partial u} \] is the tangent vector at \( u \), rotating the tangent vector by matrix \( \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \) derives the normal vector.

2.4. Parametric mapping

Based on the mapping between parametric \( u, v \) coordinates and orthogonal \( x, y \) coordinates established for each edge, we construct a rectangular parametric

![Fig. 2. Example: Eye edges with spline fitting: (a) eye image; (b) edge splines; (c) eye image with edge splines.](image)
single searchable entity, then recursively split the edge into two segments. We place the split point according to where the vertical integral projection $G$ is maximized along the $u$ direction [27]:

$$G = \max \left( \frac{\partial}{\partial u} \int_{v}^{v_0} e^{-v^2/2\sigma^2} P(u, v) \, dv \right),$$

(5)

where $P(u, v)$ is the highpass paramap value at parametric coordinates $(u, v)$ and $u_L \leq u \leq u_R$, where $u_L$ and $u_R$ are the leftmost and rightmost values of $u$ defining the edge segment to be subdivided (see Fig. 3). Eq. (5) essentially divides the paramap where its average gradient in the $u$ direction is maximized. Note that the values are Gaussian center weighted to ensure that remote features near the periphery of the paramap have less influence on the subdivision than the center features do. $(v_0$ defines the variance of the Gaussian curve.) Considering the real implementation in the discrete domain, the vertical integral projection is conducted within a small range of incremental step $\Delta u$, our edge statistics show that inflection points or intersection points of edges have local maximum $G$ values, therefore, Eq. (5) naturally subdivides edges at inflection points or where other edges intersect.

This recursive subdivision process continues until either no more inflection points exist or the segments reach a minimum length (e.g., 8 pixels long). We retain the keys at all levels of this subdivision hierarchy in both the source and target; therefore, we can always attempt to match the longest possible edge first, then subdivide as needed to optimize for accuracy. It should be noted that this strict binary subdivision algorithm creates at most $2n - 1$ total searchable entities from an edge with $n$ total segments; this is a great improvement over simply entering every possible combination of edge segments into the database.

Our subdivision algorithm is also illustrated in Fig. 3. In this case, the paramap is split into a total of eight segments. Subdivision starts by splitting the complete edge in two at the strongest inflection point (i.e., at the point corresponding to the center dip in the edge of the letter “M”). Each of these two segments are then split again at the next strongest inflection point in each, and the process repeats for three levels to yield $2^3 = 8$ segments. Any of the $2^3+1 - 1 = 15$ combinations can potentially be matched.

2.6. Key generation

A primary requirement of HyperRes is to be able to efficiently match edge details (i.e., the contents of paramaps) between the source and target images. However, it is inefficient to do this directly on the paramaps when searching a huge database of edge details, since the performance depends on the size of
each paramap (which may be large for very long edge segments). Therefore, to facilitate efficient operation, we use two techniques: key generation and multi-resolution decomposition.

Key generation refers to the construction of fixed size keys for each edge segment based on its paramap; these keys are then entered into the database and searched much faster than the paramaps themselves. To construct each key in our implementation, we take the corresponding paramap section (which is typically \(16 \times L\) pixels in size for a segment of length \(L\)) and warp it into a \(16 \times 16\) region through bicubic interpolation. Since fixed size keys may discard important image details in long edges, we rely on subdivision (as described in Section 2.5, Fig. 3) to ensure that any details lost in making a key for a long edge will still show up in the corresponding key for a subdivided segment of that edge. In addition, our implementation further avoids this information loss by using three distinct key sizes: edge segments from 8 to 23 pixels are normalized into \(16 \times 16\) pixel keys (denoted as \(K_{\text{short}}\)), segments from 24 to 63 pixels are normalized into \(16 \times 32\) pixel keys (denoted as \(K_{\text{medium}}\)), while all longer edges are placed into \(16 \times 64\) pixel keys (denoted as \(K_{\text{long}}\)).

2.7. Reconstruction and rendering

2.7.1. Search and refinement

During reconstruction of a high-resolution image, we take a strategy (so-called global-to-local search) to find the optimal match between target paramaps and source paramaps. The search-stage algorithm first attempts to find the longest possible source edge matching a given target edge. If the match is not adequate at some points on the edge, the refine stage subdivides the target edge and recursively repeats the search for a shorter source edge to better match each subdivision. The match is based on the similarity between the target keys and the source keys, which is measured by mean square error \(\delta\) (or called score of difference (SoD)). The refine stage is analogous to the subdivide stage used prior to database construction: it is present to prevent the search stage from becoming desperate and returning something having little similarity to a given target edge segment. It takes the results of a key exiting the search stage and determines if its final value of \(\delta\) exceeds some threshold \(\delta_s\). If this threshold test passes (i.e., \(\delta < \delta_s\)), execution proceeds to the next stage. Otherwise, HyperRes locates the failed target key’s two subdivided keys on the next level of the subdivision hierarchy, and recursively sends them back to the search stage to look for a shorter but better match.

In order to accelerate the matching process, we use a coarse-to-fine multi-resolution searching scheme to find the SoD (\(\delta\)) between target keys and source keys. For each \(M \times N\) pixel key (e.g., \(16 \times 16\) pixels), we build a series of \(k = \log_2 M\) images, with each image half as large as the previous one. This process continues until \(4 \times 4\) pixel \(K_0\) (coarse level 0) keys are generated. These \(K_0\) keys represent only the critical large-scale spatial details found in each paramap; two very similar \(K_0\) keys are also likely to be similar when compared at higher fine levels.

The actual search algorithm as implemented in the search stage first compares the \(K_0\) version of the target key to each \(K_0\) key in a subset of the database, keeping track of which keys are similar enough to the target (i.e., with mean square error \(\delta\)) to pass a given threshold \(\delta_0\). Only the passing keys are promoted to the \(C_1\) search level, where keys with twice the resolution are compared to the \(C_1\) version of the target key. This process continues until either only one possible key remains or the top of the hierarchy (i.e., with full \(K_4\) keys) is reached. In either case, the best full resolution match is selected and returned. In our implementation, the search space is normally reduced by a factor of 30 after each successive level.

Additional performance gains are achieved through clustering: sets of very similar \(K_0\) keys are linked to a single averaged \(K_0\) key in the database; this reduces the number of keys substantially. The clustering threshold \(\delta_c\) can be adjusted to limit the total \(K_0\) database size to a few thousand keys, thereby making it fit entirely into the data cache of a modern microprocessor.

Note that to ensure accurate matching, we must “degrade” the keys constructed from source paramaps to approximate the degradation induced by the natural undersampling of the target. This concept, introduced in [12], involves Gaussian blurring the source image to approximate the target’s lack of high-frequency spatial detail. In Fig. 4, we show a graphical representation of the matches found between source and target edge segments and their corresponding paramaps. Note the clear visual correspondence between the source and target paramaps, even though their corresponding edges may have entirely different curvatures.

Fig. 4(e) shows HyperRes matching at work: source edge segment S3 is really the left subdivision of S2; splitting the edge has allowed the search algorithm to find a nearly perfect match for target T4. Also note that two of the three source segments have been applied multiple times to the five target edges. HyperRes places no restrictions on such reuse, assuming it results in the best match. However, each target edge segment can only accept data from one source.

2.7.2. Merging matched high-resolution data

The final results of the search and refine stages are in the form of key pairs: a target key and its matching source key, either of which may represent a small segment of an original edge. In the Merge stage, we map the source key back to the corresponding subsection of the paramap owning it, and merge that parametric pixel data back into the correct place in the target edge’s paramap. This is a
simple matter of cut and paste: the high-resolution source
paramap section is pasted directly over the corresponding
low-resolution data in the target paramap. The source may
require only bicubic interpolation to make it fit into the
target, since our search algorithm guarantees that the
source and target segments are roughly the same length.
By virtue of our subdivision algorithm (which splits edges
only at their natural inflection points), we are also
guaranteed that the textures near the ends of the selected
source paramap are compatible with those of the target;
therefore, no extra blending is needed to make the high-
resolution details fit in visually.

2.7.3. Rendering the final image
After all target paramaps have been updated with
high-resolution information, we must render them back
into the image in the OrthoMap (orthogonal mapping)
stage by effectively inverting the parametric mapping
done earlier. The OrthoMap stage takes each point in
each target edge’s paramap, projects its $u, v$ coordinates
back into orthogonal $x, y$ space, and renders that pixel
into a temporary buffer. To ensure visual continuity, we
use anti-aliased merging to plot each point with subpixel
accuracy. In addition, we use alpha blending techniques
to ensure that high-resolution data are only rendered
around where the original target image contained edges;
this guarantees that any artifacts picked up in the search
stage do not extend out into regions expected to be
visually smooth. Since the final temporary buffer
contains only the reconstructed high-frequency compo-
nents, and the target image effectively contains only low-
frequency data; adding the two together will complete
the final high-resolution image and hence the HyperRes
algorithm itself.

3. Experimental results
Numerous experiments were conducted on different
types of images to evaluate the quality of results
generated by our algorithm; example images are shown
in Figs. 6–8 for both controlled as well as qualitative
evaluations. In each case, we show the results of
doubling (or four times) the resolution of a target image
using bicubic interpolation, bicubic interpolation with
unsharp masking and finally HyperRes processing. Note
that HyperRes processing was applied only to the
luminance channel of all images, since most of an
image’s visual information content is in its luminance.
The chrominance channels were enlarged with tradi-
tional bicubic interpolation. We also evaluate the
HyperRes through its application to face feature
detection and identification.

In our experiment, 12 representative images are used
as source images (as shown in Fig. 5). There are total
23 476 keys being extracted from these images, in which
there are 12 056 short keys ($K_{short}$), 7548 medium keys
($K_{medium}$), and 3872 long keys ($K_{long}$).

3.1. Controlled evaluations

In our first experiment, we use HyperRes on the
images in Fig. 6 to double the resolution of a 128 \times 128
pixel target image using a single 256 \times 256 source image
(b). Both the target image and source image are non-
overlapping crops of (a); this is to demonstrate the
performance of our algorithm under uniform lighting
conditions. To ensure an accurate evaluation, we would
like to compare the output of HyperRes to an optically

Fig. 4. HyperRes in action (matching and subdivision): edge
segments from the source image (a, c) are matched to the target
image (b, d). The edge segments are fitted by parametric curves.
$T_n$ represents the target key patch of the edges; $S_n$ represents
the source key patch. In (e), multiple source edge segments (top
row) contribute high-resolution detail to similar target edge
segments (bottom row). These images are taken from Fig. 6.
enlarged “control” photograph of the same target subject. Our control is a single $256 \times 256$ image of the target shown in (c). To form the true target image processed by HyperRes, the control image was Gaussian blurred and then downsampled to $128 \times 128$ pixels; this accurately simulates originally capturing the image at half the optical resolution we desire. Fig. 6(d) shows the result of bicubic interpolating this target back up to $256 \times 256$ pixels in preparation for HyperRes processing as discussed in Section 2.

The results of HyperRes processing are shown in (f). Figs. 6(g)–(j) give enlarged $64 \times 64$ crops from (c)–(f) to demonstrate results at the pixel level. Figs. 6(d) and (g) are the result of traditional bicubic interpolation. To provide a fair comparison with traditional interpolation techniques, strong unsharp masking has been applied to the bicubic interpolation in (e) and (h). Perceived quality is improved by this sharpening, but upon closer inspection, it is clear that edges and fine detail are no better resolved. HyperRes is applied in (f) and (i), and an equivalent crop of the control image before degradation is given in (c) and (j). In comparing the control image to the results of HyperRes processing, it is clear that our algorithm has accurately reconstructed most edge and texture detail by using the corresponding structures in the source image (b) (see Fig. 4 for an example of this matching accuracy). In this case, HyperRes has produced an image with subjectively sharper details than even the original optically captured control image; in addition, it lacks most of the jagged edge artifacts induced by unsharp masking the interpolated image. Note that we have omitted specific correlation measurements (such as mean square error), as Freeman et al. [12] have shown that such values are not representative of whether a given super-resolved image is visually “closer” to an ideal control image than an interpolated version of the image is.

Another more complex image showing a nature scene is given in Fig. 7. In this case, we start with the $256 \times 256$ target image, which is a crop of photograph (a). HyperRes is then used to reconstruct this image as the $512 \times 512$ output image (e). As in Fig. 6, we have formed (c) by downsampling the $512 \times 512$ control image (b), then bicubic interpolating back to...
512 \times 512$. This time, we use the 12 source images shown in Fig. 5 to illustrate how HyperRes can utilize high-resolution data from more sources with different lighting conditions.

Figs. 7(f)-(i) give $128 \times 128$ crops of (b)-(e) to illustrate operations at the pixel level. The traditional bicubic interpolation results in (c) and (f) show a lack of sharp edges around the rocks, poor reconstruction of the individual stems of the brown moss, and blurred textures on the rock faces. Applying unsharp masking in (d) and (g) only increases jagged edge artifacts without adding more detail. In contrast, the HyperRes image of (e) and (h) accurately corrects all of these problems to yield an image that is nearly as detailed as the control image in (b) and (i). In some cases, HyperRes has been able to deliver even sharper edge details than the control itself: the algorithm has used its powerful edge model to approximate and reconstruct the subpixel “width” of each crack between the rocks. Note that some edges are brightened due to the fact of many-to-one match in the key-matching stage. The statistics in our case study show that among the matched high-resolution edges, there are 26% bright edges are selected more than twice.

3.2. Qualitative evaluation

To examine the qualitative performance of HyperRes when a control image is not available (as is the case in real world usage), we have applied our algorithm to the
64 × 64 pixel target image of the human eye (see the crop of Fig. 8(a)). The right side image of Fig. 8(a) shows the magnified image (256 × 256) by using our HyperRes method. (As before, 12 source images are used.) Fig. 8(b) and (c) show the results by methods of bicubic interpolation and unsharp masking for comparison. Clearly, the HyperRes reconstruction in (a) has leveraged the available data to dramatically enhance the result compared with both (b) and (c). Notice the sharp and accurate reconstruction of facial hair, skin textures, and iris details with minimal artifacts. This is made possible by our algorithm’s unique edge structured matching approach: intersections between edges often substantially improve matching accuracy.

3.3. Evaluation through the application to facial feature detection and face identification

In some applications, such as capturing face images in a distance by a digital camera or multi-frames face images in surveillance video, low image quality limits the ability for the subject identification. Due to the uncooperative imaging circumstance, the valid face region captured by a camera could be as small as 20 × 20 pixels. The small-size images not only make the face recognition task difficult, but also affect the accuracy of face detection and facial feature extraction. One solution to improve the image quality is to increase the image resolution.
We applied HyperRes technique to meet this challenge. The HyperRes algorithm is tested on 60 frontal-view face images. Fig. 9 shows one example of the frontal view face. We still use 12 images in Fig. 5 as the source images. The example of resolution-enhanced eye images (magnified by 4×) and the reconstructed highpass edge details by HyperRes algorithm are shown in Fig. 10, the further increased resolution on the local eye-corner areas can be more clearly seen in Fig. 11(b) than the image produced by interpolation method in Fig. 11(a).

The image quality of HyperRes is assessed by the accuracy of eye feature detection. Based on our existing work [28] we have developed an energy-oriented mesh [29] for adapting a mesh model to the eye features. However, the accuracy and robustness of the mesh motion heavily rely on the image resolution because the mesh nodes move in the subpixel domain. HyperRes can greatly improve the adaptation performance. The matching accuracy has been improved by averagely 3.6 pixels on eye contours of 60 tested face images as compared to the case without applying HyperRes.

The HyperRes algorithm is also evaluated through our developed 3D face recognition system, which consists of 3D face modeling and 3D face structure similarity measurement [30]. HyperRes has been applied to this system as a pre-processing stage in order to increase the quality of the input image and video. The increased image resolution facilitates the feature detection, and thereby helps for the generation of high-resolution facial models. HyperRes
is a critical step to improve the system performance. The effectiveness of the image resolution enhancement has been verified by comparing the correct recognition rate of face images with and without applying the HyperRes algorithm. Sixty subjects’ facial images (two views per person: front and profile) are tested for recognition. Experimental results show that the HyperRes algorithm improves the recognition performance by a 16.3% when compare to the case without applying HyperRes algorithm (see [30] for detail).

4. Discussion

Our experimental results have shown that the HyperRes model has a number of advantages over conventional techniques including bicubic interpolation and unsharp masking. Bicubic interpolation results in blurred images lacking distinct edges and fine textures. Unsharp masking helps to restore image contrast, but introduces artifacts and still cannot bring back detail lost during the interpolation process. We conducted the subjective assessment through a “blind” visual preference experiment by multiple observers. The subjective rating scale is designed as in Table 1.

The experiment was performed by 10 volunteers including five graduate students, two psychologists and three faculties in SUNY at Binghamton. Hundred images (including human faces, nature scene, etc.) are used as original control images, each image is downsampled to a quarter size, then magnified by four times by three algorithms: HyperRes, bicubic interpolation, and unsharp masking individually. Results in Table 2 show that HyperRes algorithm in visual preference test has over 87% ratings at the category 4 and above, which outperforms the peer algorithms at the rating category 4 and 5 by over 40% preference.

Due to the software of existing algorithms [12] not available for public, we are not able to conduct the direct subjective comparison. However, in the following sections, we will discuss the advantage and limitations of our algorithm against those algorithms.

4.1. Computation efficiency and parallelism

Because our edge-based representation is used only where it is important (on high-frequency edge detail), not on low-frequency areas where increased resolution is
irrelevant, HyperRes can distill images into much smaller datasets than patch-based approaches. For instance, patch-based algorithms [12] decomposes several source images into 100,000 small patches, each of which must be searched in the database. In contrast, HyperRes only generated around 20,000 searchable edge keys for an image of comparable size and detail.

Furthermore, HyperRes is capable of massively parallel and pipelined implementation, as shown in Fig. 1. To ensure visual continuity, existing algorithms must consult neighboring areas after each image patch is super-resolved; this dependency precludes parallel implementation in most cases. In contrast, by pre-processing an image into independent edges and working on each edge's parametric space in parallel, HyperRes can sustain far greater performance, making it appropriate for real-time processing in both software and hardware.

For instance, it is noted in [12] that “typical processing times for the single-pass algorithm are 2 s to enlarge a 100×100 image up to 200×200 pixels.” and 20–100 s on a Pentium-IV 1.7G PC are needed by the similar algorithm in [19]. In contrast, on a 1.7 GHz Pentium-IV-based computer system, our optimized implementation of HyperRes can double a 256×256 image to 512×512 in just under 1.2 s.

### 4.2. Working on noisy images

When the signal-to-noise ratio of an image is low, the use of small patches without regard to higher-order structures (i.e., edges) can generate useless data. We attack the noise problem on two levels: the identification of edges themselves and the accurate matching of parameters during image reconstruction.

HyperRes incorporates three techniques to prevent noise from interfering with edge construction itself. First, after mapping an image’s edges into sorted lists of points (Section 2.2), the algorithm identifies any short edges interrupted by noise pixels and merges their adjacent endpoints to form a single longer edge. Second, we discard any merged edges less than 8 pixel units in length or with very tight curvatures; this avoids degenerate cases where a cluster of adjacent noise pixels form a pseudo-edge. Finally, during the parametric curve fitting stage (Section 2.3), we never fit the curve to each and every point, as the point coordinates are rarely noise free due mainly to rounding issues. Instead, small groups of points are reduced to a single noise-free point by taking the vector median of the surrounding edge point coordinates; this not only eliminates the potential for rogue points to disrupt the smooth curve but also...
reduces the number of spline coefficients required to represent long edges.

Note that the patch-based approaches utilized Markov network to enforce spatial continuity, instead, the HyperRes algorithm uses arbitrary length edges, which make the resulting images maintain the same spatial continuity while even more efficient and faster.

4.3. Affine transform independence

Pixel patch-based super-resolution algorithms have difficulty accurately matching visually similar regions after more complex affine transforms (rotations and perspective changes, in particular) have been applied to the camera coordinate system. In contrast, a parametric representation of edges allows us to match image details under any conditions, even if the transform is completely nonlinear: the algorithm naturally follows the curvature of edges, regardless of how that curvature may have changed. As a result, HyperRes requires fewer high-resolution source images to achieve high-quality reconstruction than [12]. In addition, our use of edges allows HyperRes to construct a “model” of common image shapes and their associated textures without requiring this knowledge ahead of time, as model-based algorithms [14–16] require.

4.4. Limitations of the HyperRes model

In image regions where the local gradients are too weak to qualify as true edges, our algorithm may fail to reconstruct fine details without edges to structure the high-resolution pixels around. As a result, HyperRes may not achieve optimal quality when the input images are so blurred that accurate edge detection is impossible. In this case, other model-based techniques [14–16,19] have achieved much better results. However, if sufficient baseline resolution is available, our algorithm is more scalable to higher resolutions. Sometimes due to the unbalanced (or unnormalized) brightness of source–target edges and many-to-one match of key-edges, some bright edges may be synthesized with an unnatural appearance. The possible way to alleviate this problem is to apply a consistency check of reconstructed edges or to compensate illumination difference between source–target edges in order to reduce the large brightness variations of adjacent edges in the local area. Also of concern is optical aliasing (moire patterns) and its ability to create false edges. Our initial experiments did not involve these scenarios, so future investigation is needed in these areas.

5. Conclusion and future work

In this paper, we have proposed **Hyper-Resolution** (HyperRes), a new algorithm for increasing image resolution. The foundation of our algorithm lies in the restructuring of image data around edges. Increased resolution is obtained by augmenting low-resolution textures in the target image with high-resolution edge structures taken from a database of source images. Compared to existing methods, HyperRes delivers higher quality, increased flexibility and substantially faster performance. However, HyperRes may have some limitations in applications involving many weak edges or false edges.

In the future, we will extend the edge structured framework provided by HyperRes to incorporate a variant of registration-based super-resolution into the present algorithm to further improve its flexibility and accuracy. By matching the shapes of edges in addition to the textures surrounding them, we can assume that if certain edges are identical in both shape and appearance across multiple source images, these edges represent the same object but with different affine transforms applied to the camera coordinate system. Since the HyperRes edge model is affine transform invariant, we can apply traditional registration-based super-resolution techniques to the set of paramaps believed to refer to the same edge across multiple images, thereby obtaining a single paramap of even higher resolution than the present algorithm can provide. In addition, the use of chrominance information may provide increased matching accuracy. Our future work will focus on developing a technique by combining the HyperRes, registration-based super-resolution and Bayesian approach to tackle the very challenging issues in face recognition, for example, with extremely low-resolution face image (e.g., under 20 × 20 pixels).

**Acknowledgments**

This work is supported by the National Science Foundation under Grant no. IIS-0414029. We also thank the SUNY Upstate Medical Center for the partial support and Dr. Anup Basu for the discussion and proof-reading of the paper.

**References**


