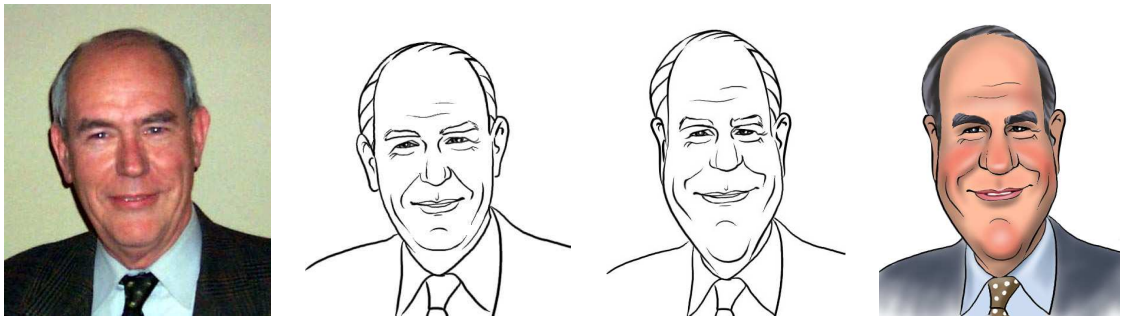
# Sample Based Face Caricature Generation

Heung-Yeung Shum[†] Ying-Qing Xu[†] Michael F. Cohen[‡] Hua Zhong[§]

[†]Microsoft Research Asia
{hshum,yqxu}@microsoft.com

[‡]Microsoft Research
mcohen@microsoft.com

[§]Carnegie Mellon University
zhonghh@cs.cmu.edu



(a) (b) (c) (d)

Figure 1: Turing Award winner Ivan Sutherland's caricature generated by our system. (a) is the input frontal face image. (b) is the unexaggerated line-drawing. (c) is the caricature generated by our system. (d) is a shaded caricature based on our system's result.

## Abstract

In this paper we present a system to automatically create a caricature drawing from a frontal face photograph. The system learns how to exaggerate features based on training examples from a particular caricaturist.

The input to the system is a frontal face image along with its vector based line-drawing. The system creates a face shape model of the given face with a minimum of user interaction. The face shape is then converted to a set of semantic face features. A Kernel Regression (KR) algorithm determines how much each feature should be exaggerated based on the knowledge learned from a particular artist. The exaggerated features are then projected into an exaggerated face shape using a maximum likelihood estimation (MLE) algorithm. This MLE algorithm also maintains face constraints to generate reasonable caricatures. Finally, the exaggerated face shape together with the unexaggerated face shape are used to morph the input line-drawing to create a caricature. A user interface is provided to indicate the extent of overall exaggeration. We have trained the system with two separate artists and show results of passing several photographs and line drawings through the system.

## 1 Introduction

**Car • i • ca • ture:** *A representation, especially pictorial or literary, in which the subject's distinctive features or peculiarities are deliberately exaggerated to produce a comic or grotesque effect.*[1]

Caricature can also be defined as "an exaggerated likeness of a person made by emphasizing all of the features that make the person different from everyone else."[14] An experienced caricaturist is trained to spot the most distinguishing features of a particular person and exaggerate them in such a way that the caricature will look much more interesting than a simple likeness. This skill needs to be developed through years of experiences and is very difficult to master quickly. Our goal is to create a caricature system to allow non-artists to create caricatures easily with minimum interaction. We take a learning approach by providing the system a set of training examples from a particular caricaturist. The system learns the mapping from an input face to a caricature of that person. It then automatically generates a caricature with an input of any new face.

Drawing caricatures is a highly artistic process. A single caricature cannot be objectively judged as correct or wrong. Different artists develop different styles for caricatures. We thus aim to automatically generate results from a training database created by an artist so that the caricatures all share the same style, but without judging whether this particular style is the best. Since our system learns the style from database of examples, changing the style involves retraining the system from a new set of examples from a different artist. We will show results from two artists' styles.

It should be noted that when artists draw caricatures from a live model, they are trained to observe the subject from all views and then find the best way to exaggerate the most distinctive features of the face. In our case, the system is designed to aid users to create caricatures from a single frontal face drawing or photograph. It is very easy to capture a picture of the face with a digital camera for input. We thus confine our problem to the 2D space of frontal faces realizing this may limit the quality of our system's results. We will show that most facial features when measured from a 2D frontal

face image provide enough information to create a successful caricature. These features include the width of the face, size of the nose, eyes, and so on.

There are many techniques used by a caricaturist. Changing the geometric shape and positioning of facial parts is one of the most common and useful techniques. Other techniques include using the strokes to emphasize important parts of the face or simplify other parts of the face. In this paper we focus only on techniques that exaggerate the size and positioning of facial features.

Our goal is to build a system which can generate caricatures of 2D frontal face images by exaggerating the geometric face shape based on caricature examples provided by an artist. We wish to do this with a minimum of required user interaction while still giving users some overall control of the output. To achieve this goal, we solve the following problems:

- How much should each feature be exaggerated or understated? (We will use the term exaggerate generically even if the modification of a feature to diminish its size.) We define a set of semantic features and learn what features should be exaggerated and the magnitudes of exaggeration from a training data set. We assume that there exists a nonlinear mapping between corresponding unexaggerated and exaggerated features. The complex nonlinear mapping is learned through kernel regression. The kernels re-weigh the training data before doing a linear regression.

- How can we encourage reasonable constraints on the face shape while exaggerating? After exaggerating the face features, we must still create images recognizable as faces. For instance, the eyes should always be located above the mouth, the mouth cannot be wider than the face. Rather than enforcing explicit constraints such as these, we achieve our goal by constructing a lower-dimensional subspace of all the example caricatures. The face with the newly determined exaggerated features is then projected onto this space. The result is then pulled towards the set of artist drawn caricatures in this space through an optimization process. The intuition is that since the hand drawn caricatures are all valid, then the sub-space spanned by them will also contain only valid face shapes.

An important observation which has been largely ignored by previous systems is that the exaggeration of any feature cannot be done out of context, but has to be done in relation to its adjacent features. For example, a face may have a nose of normal size, but very small mouth and eyes. A caricaturist may decide in this case to make the nose larger since its *relative* size is large compared to the nearby features. The difference between a feature and its own mean is called a first order relationship, while the difference between the relative size of one feature with other features and its mean is the second order relationship. These two relationships are described in [14] as a basis for performing caricature exaggerations. We show that the generality of the kernel regression is able to fully exploit both the first and the second order relationships when exaggerating a face.

### 1.1 Related work

Recently, a number of non-photorealistic rendering algorithms have been proposed to automatically "draw" sketches [5], engravings [13], oil paintings [10], and line-drawings with different styles [8; 11]. These systems focus on the painting style or stroke style of the image but not on the higher level aspects of the drawings such as the relationship among facial features as in facial caricatures. Some tools have been built to aid users to create caricatures [9; 2], but they do not instruct the users how to exaggerate. While experienced caricaturists may be able to generate interesting caricatures with these tools, the level of automation does not provide ordinary users with sufficient guidance.

Some automatic caricature generators have been constructed. The most common idea is to find the difference of a particular face to an *average* face and then enlarge this difference. This approach has been employed in [4; 15; 11]. These systems provide some information about how to exaggerate and the magnitude of the exaggeration can be easily adjusted. But they only consider the first order relationship and when users adjust the magnitude of exaggeration, these systems have no guarantee that the constraints of human facial integrity will be kept. Sometimes, if the distance between two eyes has been enlarged too much, the eyes may extend beyond the face contour. In [6], a set of pre-designed exaggeration templates are used to generate caricatures. In this way facial constraints are maintained but users are left with only a limited choice of templates to be used for exaggeration and the magnitude of exaggeration is fixed.

In [12], a caricature training database is used and some exaggeration prototypes are learned from the database using PCA. When a new face image is presented to the system, a particular prototype is chosen automatically and the magnitude of the exaggeration is discovered through linear regression. Two obvious drawbacks of this system are that there is no user interaction function and no face constraints have been enforced. These two drawbacks are intricately tied to the algorithm so it cannot be easily fixed in their system.

## 2 System Framework

Figure 2 outlines the framework of our caricature system. A face is represented in three different ways as it passes through the layers of the system.

The topmost layer is the *Image Layer* in which the face is represented as an image. The input can be a photograph or a vector based line-drawing of a face drawn with Adobe Illustrator. The output is a caricature image generated by the system. In the second *Shape Layer*, the face is represented by its shape. More specifically, we define a ***face shape model*** with 92 key points as shown in Figure 3. Given a frontal face image, the face shape can be semi-automatically located with our tool [7]. The shape representation of the face is then projected onto the *Feature Layer* where a set of high level semantic features of the face are determined based on the shape model. Then we use a *Kernel Regression* (KR) algorithm to map unexaggerated face features to exaggerated face features with the help of our training data. After we compute the exaggerated face features, a *Maximum Likelihood Estimation* (MLE) algorithm is used to find an exaggerated face shape which best fits the exaggerated features while maintaining human face constraints.

In the synthesis stage, we use exaggerated and unexaggerated face shapes to morph the vector based line-drawing and generate the output caricature. The exaggerated and unexaggerated face shapes serve as source and destination morphing features in this process.

After our system has automatically generated a caricature, the user can adjust the result both globally (changing the overall exaggeration magnitude) and locally (adjusting the size of individual semantic facial feature) with the user interface of our system. This gives our system maximum flexibility to help both untrained users to generate more customized caricatures and aid experienced users to generate more creative caricatures.

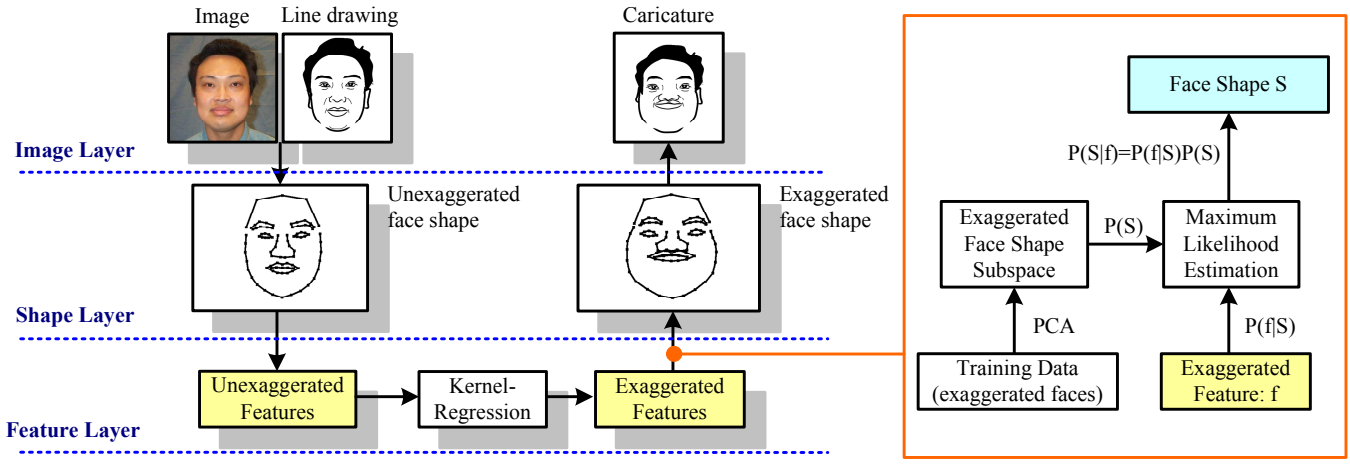
Our system has the following three components:

1. The definition of the semantic facial features.

2. The KR algorithm to map unexaggerated features to exaggerated features exploiting the information from training database.

3. The MLE to compute a face shape to best fit exaggerated features and enforce human face constraints.

We will discuss these in the following Sections 3, 4 and 5.




(a) System Framework (b) MLE

Figure 2: System Framework Diagram. (a) is the 3-layer system framework. A given face image is first converted to face shape, and then the features. The exaggeration is done in feature layer with a Kernel Regression algorithm. And from exaggerated feature to exaggerated face shape, we use a MLE algorithm which is shown in (b). The MLE algorithm balances both the face constrains enforced by the exaggerated face subspace and the similarity to the given exaggerated face features.

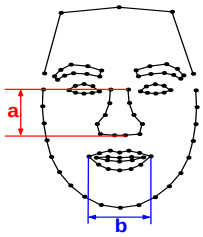



Figure 3: Face shape and definition of semantic features. The Face Shape has 92 key points. And the semantic face features are defined as the distance between two key points along Y direction (a) or along X direction (b).

### 2.1 Training Data

Our training data consists of 210 facial images. We asked two artists to draw a caricature for each face. Figure 4 shows four sample faces in our training data set. We can notice the different styles of the two artists. Because the learned exaggeration style of our system is consistent with the style in the training database. By creating two databases, we enable users of our system to exaggerate a face in two styles. Then we manually labelled face shapes for each face image and caricature. Finally all unexaggerated face shapes and exaggerated face shapes were resized, translated and rotated to be aligned to a generic face shape.

## 3 Face Features

We use a set of semantic face features in our system. Both the KR algorithm and user interaction operate on these features. These semantic features are computed from the face shape model.

### 3.1 Face Shape Model

In our system, we use a face shape model which has 92 key points as shown in Figure 3: points 0-7 are for the left eye, points 8-15 are for the right eye, points 16-25 are for the left brow, points 26-35 are for the right brow, points 36-47 are for the nose, points 48-67 are for the mouth, points 68-85 are for the face contour, and points 87-91 are for the forehead. This face shape model can be located semi-automatically [7] with minimum human interaction (usually 3-4 clicks and drags). Sometimes, the 5 key points on forehead need to be manually labelled due to the high variance of hair styles. This face shape is then scaled, translated and rotated to align with a generic face shape.

### 3.2 Semantic Face Features

The 92 key points of the face shape model provide the information to create a set of higher level semantic features for the feature layer representation. These features indicate various measurements of the face, for example, “width of left eye”, “distance between the eyes”, “height of the nose”, “width of the mouth”, “height of the mouth”, “thickness of upper lip”, etc. Later our KR learning algorithm runs on these features.

More specifically, these measurements are made as either the horizontal or vertical distance between a specific pair of the 92 points, in other words between either the $X$ coordinates or the $Y$ coordinates of two points.

If we pack the 92 face shape points into a face shape vector:

$$S = (x_1, y_1, x_2, y_2, ..., x_{92}, y_{92})$$

then a specific feature can also be represented as a vector, $f$, with all zero elements except one 1 and one $-1$. For example if a feature is the difference between the X coordinates of key point 1 and key point 2, $x_1 - x_2$, then the feature vector for this feature can be written as: $f = (1, 0, -1, 0, ..., 0)$. The inner product of $S$ and $f$ is the value of the feature for the face shape: $Sf = x_1 - x_2 = a$. For each feature, there is a corresponding feature vector. Putting all $n$ features together creates a *feature matrix* $F = (f_1^T, f_2^T, ..., f_n^T)$. The features of a given face shape are then given by

$$f_s = SF$$

where $f_s$ is a 1 by $n$ vector where each element is the value of a feature. An important property of the feature matrix is that if there are no redundant features, $F$ has a pseudo-inverse. This becomes important in the later stage for finding a face shape to best fit a given feature vector $f_s$.

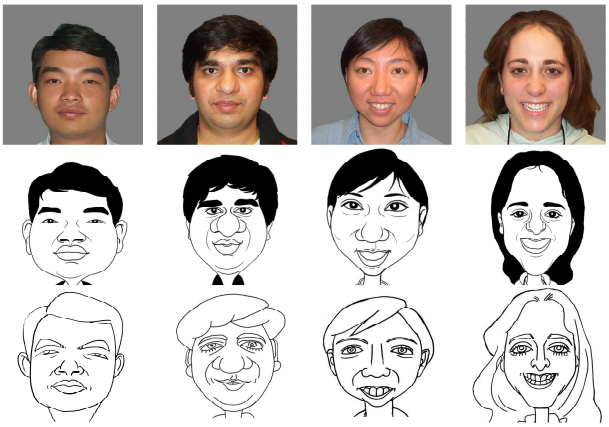


Figure 4: Examples of training data. First row are the original pictures. Second and third rows are caricatures drawn by two artists.

### 3.3 Discussion of Face Features

Why do we need to define semantic features? The first reason is that when artists draw a caricature, they pay attention to semantic facial features and the first and second order relationships mentioned in Section 1 relate to these semantic facial features. The results that we show are automatically generated. However, we also want to allow users of the system to easily modify these automatically generated results. Instead of forcing users to interact with our system in a space which is unfamiliar with them but easy to understand for machines (e.g., eigen vectors found by PCA of the face shape space [6]), users interact in a language which they can easily understand (i.e., semantic facial features in our system).

The feature definition of our system comes from two sources. First is the prior knowledge from discussions with artists. We include features to which they pay attention when exaggerating a face. The second source is from our experiments. As we will see later, these features serve as constraints for computing an exaggerated face shape. So we need a feature set that can exert enough constraints to ensure good caricature results. This was done in an interactive and iterative way: First we generate some caricatures for faces in the training database and compare them with the caricatures done by the artist. We try to find the most salient and consistent difference between these two sets of caricatures. If we find one, it usually means we should add a new feature here to provide extra constraints so our results can capture the exaggeration details in the training data. We repeat this process until we can consistently generate satisfying results.

An example is the "angle of mouth" feature. It is defined as the vertical distance of the two face shape key points on the left and right corners of mouth. Initially, this feature was not singled out as one of our semantic facial features. It appears that the two corners of a mouth on an ordinary face should always have the same height. However, we found out from our experiments that one of the two corners of a mouth can be much lower than the other in some faces and the artist invariably exaggerates this characteristic when it is present. Unfortunately, this error-finding process would be difficult to automate. For example, artists pay more attention to tiny changes of the eyes while the same amount of change for the width of the face may be thought of as trivial. For machines, it is very difficult to detect this difference. In our experiments, we compiled a feature set of 47 semantic features to ensure satisfying results.

Adding a new feature to our system is very easy. One simply adds a column in the feature matrix $F$.

## 4 Kernel Regression on Feature Space

Our goal is to learn from the training data a function, $g$, that maps an unexaggerated feature vector $f_{in}$ to an exaggerated feature vector $f_{out}$: $f_{out} = g(f_{in}, training\_data)$. We use Kernel Regression (KR) to learn $g()$. KR is an extension of linear regression that includes non-linearities into the learning process.

### 4.1 Linear Regression

Linear Regression assumes that there is a linear relationship between input $f_{in}$ and output $f_{out}$, and determines linear coefficients $A$ from the training data. Suppose we have $n$ training pairs, $n > 47, i = 1,...,n, j = 1,...47$,

$$E = \begin{pmatrix} e_{1,1} & \dots & e_{1,47} \\ e_{2,1} & \dots & e_{2,47} \\ \vdots & \ddots & \vdots \\ e_{n,1} & \dots & e_{n,47} \end{pmatrix}, U = \begin{pmatrix} u_{1,1} & \dots & u_{1,47} \\ u_{2,1} & \dots & u_{2,47} \\ \vdots & \ddots & \vdots \\ u_{n,1} & \dots & u_{n,47} \end{pmatrix}$$

and we have:

$$E = UA \tag{1}$$

where each $e_{i,j}$ is the $j^{th}$ exaggerated feature in the $i^{th}$ training data pair, and $u_{i,j}$ is the $j^{th}$ unexaggerated feature in $i^{th}$ training data. The coefficient matrix $A_{47\times 47}$ solved by Maximum Likelihood algorithm is given by:

$$A = (U^T U)^{-1}(U^T E)$$

### 4.2 Kernel Regression

Kernel regression captures non-linearity by allowing each input feature vector, $f_{in}$, to independently re-weigh the training data before running a linear regression. A kernel function, typically a radial basis function [3] centered at the input vector re-weighs the training data. When an input unexaggerated feature vector $f_{in}$ is given, the kernel function, $W(\| f_{in} - f_{train} \|)$, gives those training data points, which are closer to $f_{in}$ in the feature space more weight and those further away from $f_{in}$ less weight. Then conventional linear regression is applied to the re-weighted training data.

Let $W$ be a diagonal re-weighting matrix calculated by the kernel function, then Equation (1) becomes:

$$WE = WUA \tag{2}$$

The maximum likelihood solution of $A$ is given by:

$$A = (U^T W^T W U)^{-1}(U^T W^T W E)$$

Therefore, given an input feature vector $f_{in}$, the output feature $f_{out}$ is computed with $g() : f_{out} = f_{in}A$.

### 4.3 The Kernel Function

We use a kernel function with the form:

$$w_i = e^{\frac{1}{(\| f_{in} - f_i \| + a)^b}}$$

where $f_{in}$ is the input data and $f_i$ the $i^{th}$ data point in the training database, and $a$ and $b$ are constants. $\| f_{in} - f_i \|$ is the distance between two data points $f_{in}$ and $f_i$. The definition of this distance function is important to kernel regression because it determines the weights given to each of the training data. The Mahalanobis distance is commonly used in high dimensional space to measure distance. Training data points with larger distances to the input $f_{in}$ will get lower weights, and vice versa.

One observation is that each output feature may be highly correlated to only a few input dimensions. A simple distance in the

feature space, can thus cause unrelated input dimensions to have undue influence. To avoid this, we first determine an overall correlation between the output feature dimension and the input feature dimensions. A modified Mahalanobis distance between $f_{in}$ and $f_i$ is then found by re-weighing each dimension by its correlation coefficient to the output dimension.

Note that for each output feature, a particular input dimension will now have 47 different correlation coefficients, thus the weight matrix $W$ changes for each output feature. Therefore instead of learning a unique $A$, we learn 47 $A_j$, leading to a $1 \times 47$ vector. Equation (2) becomes:

$$W_j \begin{pmatrix} e_{1,j} \\ e_{2,j} \\ \vdots \\ e_{n,j} \end{pmatrix} = W_j \begin{pmatrix} u_{1,1} & \dots & u_{1,47} \\ u_{2,1} & \dots & u_{2,47} \\ \vdots & \ddots & \vdots \\ u_{n,1} & \dots & u_{n,47} \end{pmatrix} A_j \qquad (3)$$

This is the formula for the $j^{th}$ output feature. The solution is given by

$$A_j = (U^T W_j^T W_j U)^{-1} (U^T W_j^T W_j E_j)$$

To ensure a stable solution for the $A_j$'s, instead of running the KR directly on the original 47 input features, we first use a PCA to find a set of linearly independent bases for the features. We then run the KR on this reduced basis set.

Since our goal is to reduce the linear dependency among different features during the kernel regression rather than compression, we do not permanently discard bases with small Eigen values. We convert the results back into the original feature basis since these have semantic value. This provides a meaningful final user interaction discussed later.

### 4.4 Discussion of Kernel Regression

The parameters $a$ and $b$ in our kernel function are determined by cross-validation. A sequence of values for $a$ and $b$ are evaluated by testing 10 percent of the training data as test input with a training set consisting of the remaining 90 percent of the training data. $a$ and $b$ are selected as the ones that best return the original caricatures of the 10 percent used for the test.

As we can see from Equation (3), the output of one feature is not a function of itself only, but a function of all input features. This means that even if two faces have exactly the same value for the $j^{th}$ feature, the difference of other features will still cause the system to exaggerate the $j^{th}$ feature in these two faces differently.

The framework of our system thus naturally includes second order relationships in a straightforward way. If we only consider the first order relationship, Equation (3) would look like

$$W_j \begin{pmatrix} e_{1,j} \\ e_{2,j} \\ \vdots \\ e_{n,j} \end{pmatrix} = W_j \begin{pmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{n,j} \end{pmatrix} A_j \qquad (4)$$

## 5 Finding the Best Face Shape

Once we have obtained the exaggerated features, we need to find an exaggerated face shape that best fits the features without violating the basic constraints of a face. The inverse problem was described earlier; given a face shape $S$, the face features can be computed by multiplying by the feature matrix $F$: $f = SF$. The fit of a shape to a set of features is then defined by the difference, $\|SF - f\|$, a measure of the distance between face shape $S$'s features and the feature vector $f$. We maintain basic face constraints by finding a shape that balances lying within the subspace of shapes spanned by the artist's caricatures and having a best fit to the exaggerated features determined by the kernel regression.

### 5.1 Exaggerated Face Shape Subspace and Prior

We first construct a generative model of the *exaggerated face shape subspace*. From all the *exaggerated* face shapes in our training data, we use principal component analysis (PCA) to find a set of basis vectors that span the exaggerated face shape subspace. Then any exaggerated face shape $S$ can be written as the linear combination of these basis vectors:

$$S = XB + \mu$$

$$B = [b_1, b_2, ..., b_k]^T, X = [x_1, x_2, ..., x_k]$$

where the $b_i$'s are first $k$ Eigenvectors (92 by 1) from the PCA and $x_i$ are the corresponding coefficients for $S$. $X$ is the projection of $S$ in the space spanned by the $k$ Eigenvectors. $\mu$ is the mean exaggerated face shape. Here we take $k = 35$ which covers 96.3% energy of all the Eigen values. In our experiments we have found larger $k's$ do not generate saliently visually better results.

We assume that the probability distribution of exaggerated face shapes is a Gaussian. The probability of any given face shape $S$ projected to the subspace, $X$, becomes:

$$p(S) = p(X) = \frac{1}{D} exp(-X\Lambda^{-1}X^T) \qquad (5)$$

where $D$ is a normalization constant, and $\Lambda$ the covariance matrix where the diagonal matrix is $\Lambda = diag(\sigma_1, \sigma_2, ..., \sigma_k)$. $\sigma_i$ is the variance of $i^{th}$ dimension of the subspace.

### 5.2 Likelihood and MLE

Given a feature vector $f$, the likelihood of a face shape $S$ can be defined as follows:

$$p(S \mid f) = p(f \mid S)p(S) = \frac{1}{D'} exp(-\frac{\|SF - f\|^2}{\lambda}) p(S) \qquad (6)$$

$\|SF - f\|$ is the distance between the feature of the face shape $S$ and the given feature vector $f$. $F$ is the feature matrix. $\lambda$ and $D'$ are constants. $p(S)$ can be replaced by Equation (5).

So the problem now becomes: given an exaggerated feature vector $f$, we want to find an $S$ from the exaggerated face shape subspace to maximize the likelihood in Equation (6).

First we replace $S$ in Equation (6) with $XB + \mu$:

$$p(S \mid f) = \frac{1}{D'} exp(-\frac{\|(XB + \mu)F - f\|^2}{\lambda}) p(S)$$

Then we insert equation (5) into it:

$$p(S \mid f) = \frac{1}{D'D} exp(\frac{-\|(XB + \mu)F - f\|^2 - \lambda X\Lambda^{-1}X^T}{\lambda})$$

The $X$ that maximizes the likelihood is:

$$X_{max} = \arg\min(\|(XB + \mu)F - f\|^2 + \lambda X\Lambda^{-1}X^T) \qquad (7)$$

And we have:

$$X_{max} = (f - \mu F)F^T B^T (BFF^T B^T + \lambda\Lambda^{-1})^{-1} \qquad (8)$$

Recall that the feature matrix $F$ is pseudo-invertible (Section 3). Because $B$ contains only the basis vectors of the subspace, $B$ is also pseudo-invertible. $\Lambda$ is a diagonal matrix with all positive diagonal elements (since they are the variance of the database) so it is invertible. Therefore in Equation (8), $X_{max}$ can always be computed reliably in close form.

Equation 7 formalizes the objective to find an $X$ that minimizes a weighted sum of two terms. The first term $\|(XB + \mu)F - f\|^2$ is the distance between the face feature vector of face shape $S$ and the given feature vector $f$, the second term $\lambda X\Lambda^{-1}X^T$ is the distance

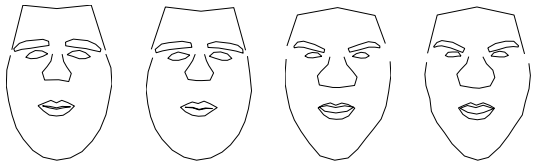



Figure 5: Some Test Results of Finding the Best Face Shape. (1-a) and (2-a) are ground truth, (1-b) and (2-b) are face shapes computed by our system.

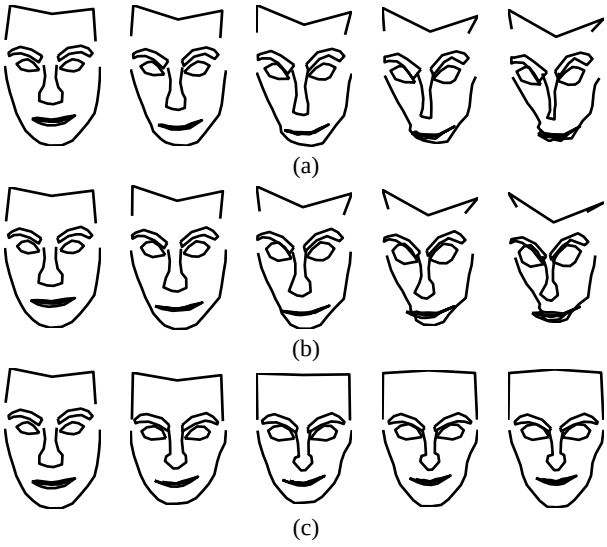



Figure 6: Constraints are automatically held in our system as shown in (c), but not for simple exaggerators as shown in (a) and (b). For each row, from left to right, the magnitude of exaggeration is larger and larger.

from $S$ to the mean face shape $\mu$ in the exaggerated face shape subspace weighted by a constant $\lambda$. An optimal $X$ thus adheres to the features predicted by the kernel regression, but provides assurance that the exaggerated face shape will not stray too far from the mean shape. Although the second term provides only a soft constraint on the final caricature, we have found that it invariably produces reasonable results.

### 5.3 Why MLE?

Figure 5 shows two face shapes created using our system to best fit the given features. Figure 5 (1-a) is the exaggerated face shape in our training database, while Figure 5 (1-b) shows the best fitted shape from the subspace. Another example is given in Figure 5 (2-a) and (2-b). Note that the fitted face shapes are nearly the same as the ground truth. As a test, we compare two simple exaggerators with our algorithm:

1. The first simple exaggerator computes the difference of key point positions of a given face and those of the mean face shape. The difference is then exaggerated (i.e., multiplied) by a factor $k$ and added back to the mean shape.

2. The second simple exaggerator first computes the eigenvectors of face shapes in the training database, and projects each face shape onto the space spanned by these vectors. Then, as in our first method, the exaggerator finds the difference between a projected face shape and the projection of the mean face shape. The differences are scaled by $k$ and added back to the mean and finally re-projected back to the original space.

3. In contrast, our algorithm first computes an exaggerated face feature vector. Then, the difference from this exaggerated face feature vector to the unexaggerated face feature vector is calculated. We multiply the difference by $k$ and add this to the original feature vector. Finally the MLE method is used to determine the exaggerated face shape.

A comparison of the results obtained by these two simple exaggerators and our algorithm is shown in Figure 6, with $k$ being 0.0, 0.5, 1.0, 1.5 and 2.0 from left to right. We can see from our result in Figure 6 that our algorithm successfully maintains the constraints of a human face while the simple exaggerators fail to do so.

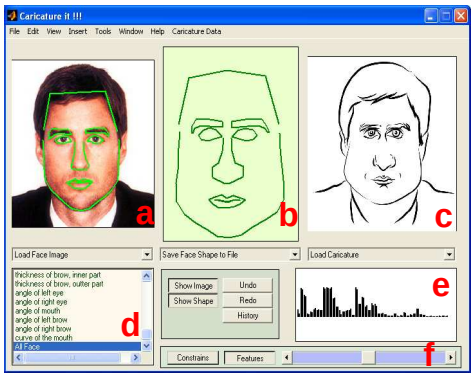



Figure 7: User interface. (a) The unexaggerated face, (b) exaggerated face shape generated by system, (c) caricature generated by system, (d) semantic feature lists and (e) visualization of the selected feature sizes (f) slider bar that can be used to adjust each feature's size or the overall exaggeration magnitude.

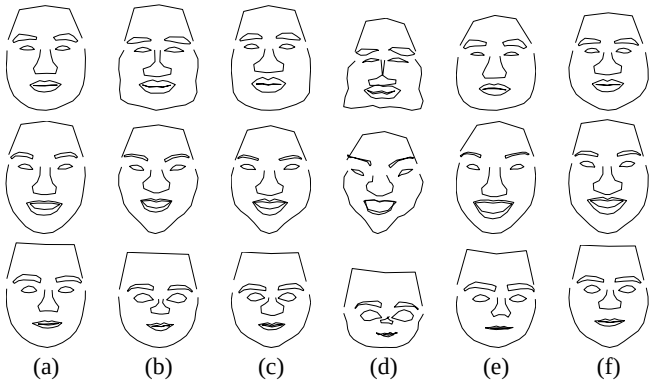
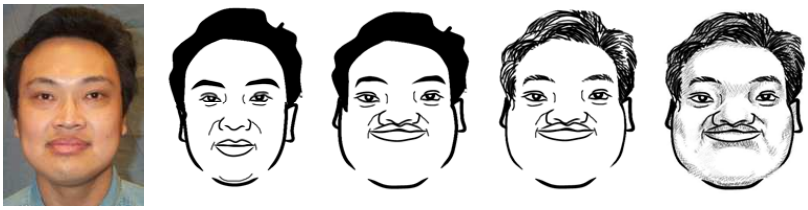



Figure 8: Some Leave-one-out test result. (a) is the input face. (b) is the ground truth. (c) is our result. (d) and (e) are results of the two simple exaggerators. (f) are the results that involve only the first order relationship.

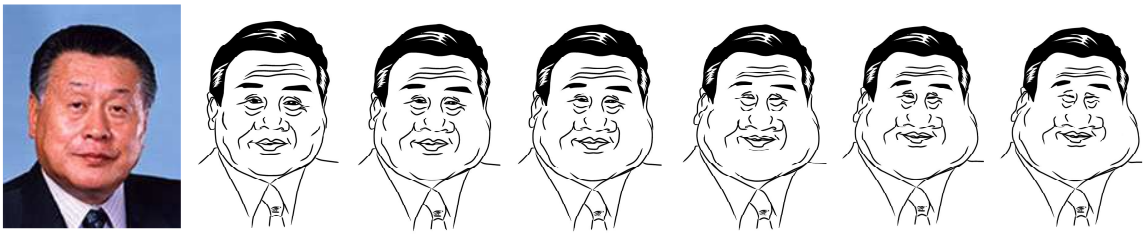
## 6 User Interaction

All the results shown in the paper were generated automatically, modifying only the overall exaggeration in some figures as indicated. In other words, individual features were not modified by hand. That said, the use of semantic features provides the basis for



(a) (b) (c) (d) (e)

Figure 9: From left to right: an input image, a line drawing, an exaggerated caricature generated by our system, the caricature with a different stroke style applied by the user, and with shading added by the user.



(a) (b) (c) (d) (e) (f) (g)

Figure 10: Caricature with different exaggeration magnitude. For the input image shown in (a), our system generated the caricatures with less exaggerating magnitude as in (a) to more exaggerating as in (g).

an intuitive user interface to provide more detailed control to the user if desired.

In the UI shown in Figure 7, users can directly adjust each feature's size individually as well as the overall exaggeration magnitude with "feature list" control and slider bar control. Since every feature in our system has semantic meaning, it is very easy for users to achieve a desired effect. For example, if they want to make the nose bigger, they can just increase the size of "width of the bottom of the nose" and "height of the nose". Our system then will automatically generate a new exaggerated face shape for the adjusted features through the MLE projection. The individual feature modifications can be seen in real time, and when the users are satisfied with the edited features, they can select "generate caricature" in the drop-down list under the caricature window to generate the final caricature.

## 7 Results

To test our learning algorithm, we have run some leave-one-out tests. First we exclude a randomly selected face from the training data and use the remaining faces as the new training data. Then we compute the exaggerated face shape for the excluded face and compare the result with the ground truth. We compare our results with the two simple exaggerators mentioned in Section 5.3. We generated results with the same algorithm but only using the first order relationship as mentioned in Section 4.4. Figure 8 shows the leave-one-out results. Figure 8 (a) are the original unexaggerated face shapes, (b) are the ground truth, (c) are our results, (d) and (e) are results of the two simple exaggerators mentioned above and (f) are the results that only involve the first order relationship. Our system creates a better match to the artist's face shape than the shapes created by the two simple exaggerators. Compared to the one using only first order relationships, our system retains more subtle details of exaggeration.

Figure 9 shows the resulting caricature generated by our system. Next to the original image (a), the unexaggerated line drawing (b) is automatically generated by the sketch generation system [5]. This unexaggerated line drawing is then exaggerated by our system to generate the caricature (c). With the caricature result, users can change the stroke styles and shade it as shown in (d) and (e).

Our system can generate a range of plausible caricatures for a single person. In Figure 10, the caricatures of a single person with different exaggeration magnitudes are shown. (a) is the input face image (b) is the unexaggerated input line-drawing image. (d) is the result generated by our system. (c) and (e) are results with less and more exaggeration adjusted by the UI of our system.

Because our system has two training databases created by two different artists, it can generate caricatures of two different styles. We show the results of two different styles in Figure 11. Users can use photoshop or other painting software to shade the caricature results to generate colored caricatures. Please note that all the results here shown in Figure 11 are automatically generated, no human interaction has been involved.

Two caricatures shown in the first row (columns c and d) of Figure 11 exemplify that our system learns distinctive features from different training databases. For the first style (in column c), the most important features include the height of nose, the height of forehead, and the width of face (across lower chin). For the second style (in column d), our system has found that the width of face (across mouth), the width of face (across lower chin) and thickness of lower lip are among the most important features. A close inspection of the original photo convinces us that all these features are good and reasonable. But it is up to individual artists to define their own styles by choosing what features to exaggerate. What our system does is to learn a particular style of caricaturing from the training data.

## 8 Conclusion

As we have seen, our example based caricature system can automatically generate caricatures from a training database with the same exaggeration style as the training database. With two sets of training data, we can thus automatically generate caricatures with two different styles.

The kernel regression operating on a set of semantic features naturally includes second order relationships. KR learns the complex non-linear mapping from unexaggerated features to exaggerated features. We have shown that reasonable face constraints are maintained through projection using a maximum likelihood estimation.

Our system only focuses on the exaggeration of the 2D geometric shape of frontal faces. No texture or 3D information is included. The feature definition was created by hand through a combination of prior knowledge and the experiments with our training database. In a new database, the current selected 47 features may not be enough. But our feature definition is easily extended. A simple user interface has also been discussed to allow users to modify results to achieve specific results.

In the future, we hope to extend our system to a 3D analysis of human faces to generate caricatures of a full 3D model. We also want to add the face texture to the analysis so the system can exaggerate not only the geometric shape of the face but also the subtle shading of the face.

# References


[1] American heritage dictionary of the english language, fourth edition. Technical report. As reported on *www.dictionary.com*.

[2] E. Akleman, J. Palmer, and R. Logan. Making extreme caricatures with a new interactive 2d deformation technique with simplicial complexes. In *Proceedings of Visual 2001*, 2000.

[3] N. Arad, N. Dyn, D. Reisfeld, and Y. Yeshurun. Image warping by radial basis functions: Applications to facial expressions. *Computer Vision, Graphics, and Image Processing*, 56(2):161–172, March 1994.

[4] S. E. Brennan. Caricature generator: The dynamic exaggeration of faces by computer. *Leonardo*, 18(3):170–178, 1985.

[5] H. Chen, Y. Q. Xu, H. Y. Shum, S. C. Zhu, and N. N. Zheng. Example-based facial sketch generation with non-parametric sampling. In *ICCV 2001*, 2001.

[6] H. Chen, N. N. Zheng, L. Liang, Y. Li, Y. Q. Xu, and H. Y. Shum. Pictoon: A personalized image-based cartoon system. In *ACM Multimedia 2002*, 2002.

[7] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance model. In *Proceedings of the 5th European Conference on Computer Vision*, 1998.

[8] W. T. Freeman, J. B. Tenenbaum, and E. Pasztor. An example-based approach to style translation for line drawings. Technical Report TR99-11, 1999.

[9] B. Gooch, E. Reinhard, and A. Gooch. Perception-driven black-and-white drawings and caricatures. Technical Report UUCS-02-002, 2002.

[10] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series. ACM, ACM Press / ACM SIGGRAPH, 2001.

[11] H. Koshimizu et al. On kansei facial processing for computerized facial caricaturing system picasso. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 6, 1999.

[12] L. Liang, H. Chen, Y. Q. Xu, and H. Y. Shum. Example-based caricature generation with exaggeration. In *IEEE Pacific Graphics 2002*, 2002.

[13] V. Ostromoukhov. Digital facial engraving. In *Proceedings of SIGGRAPH 1999*, 1999.

[14] L. Redman. *How To Draw Caricature*. Contemporary Books, 1984.

[15] T. Valentine. A unified account of the effects of distinctiveness, inversion and race in face recognition. *The Quarterly Journal of Experimental Psychology*.

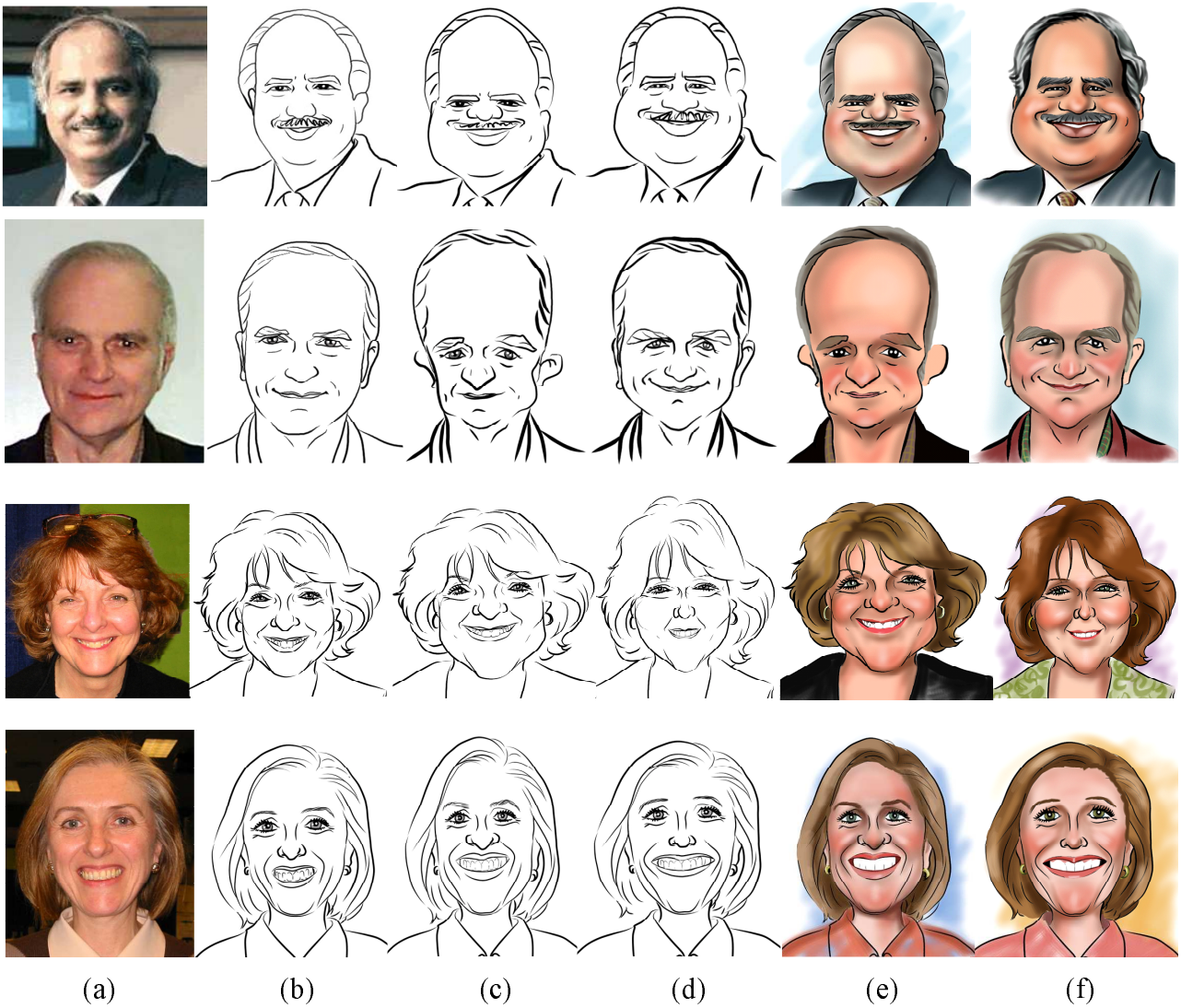



Figure 11: Caricatures with two styles. (a) are input face images, (b) are line-drawings, (c) are caricatures of the first style, (d) are caricatures of the second style, (e) are shaded caricatures of the first style and (f) are shaded caricatures of the second style.