b) If $P_i$ is a place-invariant of PTS, $i = 1, 2$, then $P_1 \cup P_2$ is a place-invariant of PTS'.

Proof: Similar to Theorem V.1.

VI. CONCLUSION

In this paper, we have considered five classes of very general but basic transformations on PTS's. They insert, eliminate or replace a set of places or transitions of a PTS, combine two PTS's into one, or split one PTS into two. Conditions for these transformations to preserve place-invariants or transition-invariants are expressed in the form of simple equalities among the new (or eliminated) places/transitions and affected places/transitions. This may be considered as an advantage of our approach when compared with those methods which require lengthy computation for deciding whether a transformation satisfies a certain pattern or not.

Our approach may be extended in at least three directions. One is to extend to other important properties. As mentioned in [1], [2], invariants of PTS's can be used to verify other properties such as liveness, boundedness, etc. However, a more direct approach is to find “patternless” transformations specifically for handling these properties. The second extension is to find transformations for other place/transition models. For instance, generalization of our results to colored Petri nets is being investigated recently [21]. Another example is predicate/action nets. While transforming a net's structure, its predicates and actions may also be changed. The specified properties have to be preserved under the new predicates. The third extension is to find transformations which can preserve two or more properties simultaneously. Though this requirement is more realistic in real-life problems, it is obviously much more difficult, especially when the properties are of very different nature.

Other important issues include how to algorithmically generate these transformations and to apply them to find invariants. This has been done in the case of eliminations recently [20].

REFERENCES


Comparison of Fuzzy Forecaster to a Statistically Motivated Forecaster

Tom Burr

Abstract—Recently a fuzzy forecaster (also called a fuzzy controller) was proposed as one method for forecasting an autoregressive time series. The approach in the fuzzy forecaster is similar to the approach in statistically motivated curve smoothers. However, the curve smoothers perform a beneficial type of data averaging that the current fuzzy forecasters do not employ. Also, the curve smoothers have a mature methodology for choosing the degree of smoothing. Therefore, in this paper we develop an enhanced fuzzy forecaster that uses some of the curve-smoother methodology and we compare the performance of the improved fuzzy forecaster to one particular curve smoother (loess) on five real and five simulated data sets. The performance criterion is the one-step-ahead forecast error variance, and the loess method outperforms the fuzzy forecaster on all five simulated data sets, and four of the five real data sets.

I. INTRODUCTION

This paper compares a recently introduced [1] fuzzy-logic-based method for forecasting an autoregressive time series to a statistically motivated forecasting method. The statistically motivated method applies the loess curve smoother [2]. We implemented an improved version of the fuzzy forecaster that was presented in [1] and compared its performance to the performance of loess. We applied both methods

Manuscript received December 22, 1995; revised August 5, 1997.

The author is with the Los Alamos National Laboratory, Los Alamos, NM 87545 USA (e-mail: tburr@lanl.gov).

Publisher Item Identifier S 1083-4427(98)00131-3.
to a randomly selected set of five real data sets, and to five simulated data sets. We knew that for the kinds of simulated data we studied, the statistically motivated method would be superior, because it was designed to be superior for that type of data. However, real data never follows any model exactly, so we did not know how the two methods would compare to real data. The conclusion here is that the loess method is also superior on most of the real, numeric data sets that we considered. In defense of the fuzzy-logic-motivated methods, we point out that originally, fuzzy-logic methods were not intended to compete with statistical methods for numeric data. Fuzzy logic was developed to handle real-world situations where either numeric data was not available or was supplemented with either expert opinion or some sort of linguistic information. Recent efforts ([1], for example) have extended the applicability of fuzzy logic to purely numeric data, and it seems that those efforts would benefit from the maturity of statistically motivated methods for numeric data, as will be shown.

This paper is organized as follows. In Section II, we review modeling and forecasting methods for scalar or vector time series, with emphasis on a fuzzy forecaster and loess. Loess is a statistically motivated forecaster that is reasonable to compare to the fuzzy forecaster for reasons we explain below. In Section III, we compare the performances of the fuzzy forecaster and loess to five simulated data sets, and in Section IV we compare the performances of the fuzzy forecaster and loess to five real data sets. Section V is a summary of the time series models considered, and Section VI is a summary of the performances of the two methods.

II. ANALYSIS OF SCALAR OR VECTOR TIME SERIES

In this section, we review vector time series modeling. In all cases, we require that the time series be stationary. Formal definitions of the stationarity of a time series can be given, but for our purposes, we simply need the mean and variance to remain the same over time. Many real-time series exhibit a trend, so are therefore not stationary. Fortunately, it is simple to remove trends that are approximately polynomial in time, such as a linear or quadratic increase in the average value of the times series over time. The simplest way to remove a linear trend in a series \( Y_t \) is to compute a new time series, say \( X_t \) that is defined by \( X_t = Y_t - Y_{t-1} \). The new series \( X_t \) would then have a constant average value. For more detail, see [3].

Note that the time series has been detrended by differencing or, equivalently, by fitting polynomial functions of time. The detrended (stationary) series will be denoted \( X_t \).

A. ARMA Time Series Models

An ARMA\((p, q)\) (autoregressive, moving average) model for the scalar, detrended series \( X_t \), can be written as

\[
X_t = a_0 + \sum_{j=1}^{p} a_j X_{t-j} + \sum_{j=0}^{q} b_j e_{t-j}
\]

where the \( a_j \) and \( b_j \) are real constants, the \( e_{t-j} \) are independently and identically distributed (iid) random variables, and \( t \in \{1, 2, \ldots, n\} \). It is common to refer to \( e_{t-j} \) as the shock at time \( t - j \). Usually, ARMA models further specify that the shock, \( e_t \), follows a Gaussian distribution with mean zero and variance, \( \sigma^2 \), denoted \( N(0, \sigma) \). In addition, to ensure that the time series is stationary (constant mean, variance, and covariances) and invertible (representable as an AR model) there are conditions imposed on the values of all the \( a_j \) and \( b_j \). See [3] for further details.

Note that \( X_t \) is a linear combination of the past \( p \) values of the series (autoregressive) and of the past \( q \) shocks (moving average).

B. General AR Models

In this section, we present a quite general AR time-series model. We will only consider AR models in this paper because of difficulties in treating nonlinear MA or ARMA models. Some of those difficulties are discussed in [4] where a practical suggestion is made for treating nonlinear MA or ARMA models. The difficulties arise because we do not observe the underlying shocks. The shocks must be estimated and there is no closed-form estimation scheme to do so. A general autoregressive (AR) time series can be written as

\[
X_t = f(X_{t-1}, X_{t-2}, \ldots, X_{t-\omega}) + e_t.
\]

The error \( e_t \) is usually assumed to be from some convenient distribution such as the Gaussian, but need not be. Nearly always the distribution of the \( e_t \) is at least assumed to be the same for all \( t \).

We will assume that the errors have the same distribution, \( F \), which we write as \( e_t \sim F(\cdot) \). Not all functions \( f \), combined with error distribution \( F \), lead to a stationary time series. We do not attempt a formal treatment of this issue, but rather accept (2) as a basis for trying our two modeling approaches. We also easily generalize (2) to include other time series, say \( Y \) and \( Z \), as

\[
X_t = f(X_{t-1}, \ldots, X_{t-\omega}, Y_{t-1}, \ldots, Y_{t-\omega}, Z_{t-1}, \ldots, Z_{t-\omega}) + e_t.
\]

Note that present values of both \( Y \) and \( Z \) are allowed as the goal is to forecast \( X \).

Equation (3) can be treated exactly like an ordinary regression model for which there are many techniques, loosely described by the degree of assumptions placed on the functional form for \( f \). We will discuss the two models shortly, but first we point out two advantages to using restrictive models such as models that assume \( f \) is linear.

1) Often there is not enough data to estimate complicated models for completely unrestricted \( f \). In fact, the old standby, linear regression, will be with us forever for this reason and represents an extreme example of combining information, in that data at one end of the range is assumed to have the same functional form as data at another end of the range. Because we are insisting on working with stationary series, we often must restrict the time window so that the series can be considered stationary over that window, thereby reducing the effective size of the data set and essentially forcing us to use only the simplest models. More complicated models suffer from the curse of dimensionality, which we will explain below.

2) For years, the regression literature has been filled with informal confirmation of the parsimony principle, which states that simplest models are preferred when possible because no data follows any model exactly, and model departures can be more severe when the training data is overfit by using an overly complex model. Fortunately, we have a straightforward way to guide us toward the proper degree of model complexity: use the model to forecast a held-out (not used for training) testing set, and accept the model that performs best on the testing set. A complete treatment of this issue uses what is known as cross-validation to repeatedly divide the data into training and testing sets. Cross-validation is very important when the data sets are small. In our case, we simply did a one-time division into training and testing sets because the data sets were reasonably large.

For notational convenience, we will drop the distinction between, for example, the time series \( X \), \( Y \), and \( Z \), and will lump all candidate predictors together to form \( p \) predictors as follows:

\[
x_t = f(x_{1t}, x_{2t}, \ldots, x_{pt}) + e_t.
\]
The main reason for writing (4) without the explicit reference to lagged predictors is so that our data sets need not be restricted to AR time series. Results of our methods in the ordinary regression setting will also be relevant for the AR setting [5]. In ordinary regression, we have \( n \) observations of the predictor, response pairs: \((x, y)\). Demonstration of properties of estimators such as convergence to the theoretical optimum value is not quite the same in the AR time series context as it is in the ordinary regression context. Also, small sample results are not necessarily the same in the two contexts. Very little has been published about small sample properties of estimators such as loess that were intended for the regression context but that can also be applied in the AR time series context. Also, in the context of time series, an important issue is how to choose the lag for each of the time series. We always use a trial-and-error approach starting with lag 1 only for the \(X\) series (the series to be predicted) and starting with lag 0 for the other series. The winning method is the one that minimizes the sum of squared errors on a held-out testing set.

In our view, using a curve smoother algorithm is usually feasible in one dimension or perhaps in a few dimensions. Occasionally, if there is enough data following (2), then a multidimensional curve smoother could be feasible. But, because many real data sets exhibit only quasistationarity, there is often not as much data following a model like (2) as one might first think. This is an important practical issue, and one that we plan to treat in a later report. In this report, we restrict attention to data sets where it is feasible to try some kind of curve smoother method (fuzzy-logic motivated or statistically motivated) because there is enough data for the problem’s dimensionality.

In Fig. 1, we present one simple idea that pervades most of this report. Consider the model \( y = f(x) + \epsilon \) in one dimension. In Fig. 1, we have \( f(x) = 4x(1 - x) \), which is the well-studied logistic map. The logistic map has properties that are interesting in the study of chaotic time series, but our concern is simply to estimate \( f \) as well as possible so that future values of \(x\) can be used to forecast future values of \(y\). Now that we know the true functional form, we know that the estimate, \( \hat{f} \), shown, is close to the true \( f \), but exhibits a slight overfit of the data because it is more bumpy than the true \( f \).

Our estimate \( \hat{f} \) was based on 200 observations of \((x, y)\) pairs. For our purposes here, we can also think of the data as being the AR time series \(x_t = 4x_{t-1}(1 - x_{t-1}) + \epsilon_t\).

The estimate \( \hat{f} \) in Fig. 1 was obtained by the loess algorithm available for example in the statistical programming language S-Plus.

1) Local Regression (Loess): Assume there are \( n \) data pairs \((x_i, y_i)\). In the context of a time series, replace \((x_i, y_i)\) with \((x_{i-1}, x_t)\) because the previous value could be used in a lag-one model to predict the present value. The loess estimate at any point \(x\) uses the \( q \) of \( n \) data pairs that are closest to the value \(x\). The closest \( q \) data pairs are used in a weighted linear or quadratic regression with weights according to the distance from \(x\). Because loess must perform the regression at each point \(x\) where the estimate is desired, it is important to use efficient algorithms. See [6] for computational details. The same idea carries over to multiple dimensions, but the current implementation of loess is limited to up to 15 predictors for local linear fits, or up to four predictors for local quadratic fits. To complete the discussion of Fig. 1, we mention that we also use a nonparametric smoother (no local linear or quadratic assumption) that we will describe in Section II-B2.

2) Nonparametric Curve Smoother: We provide here a brief description of a one-dimensional curve smoother that is a competitor to loess. The higher-dimensional smoother is a natural extension. Consider an estimate of \( f \) at a point \(x\)

\[
\hat{f}(x) = \left[ \frac{1}{(n - 1)} \sum_{j=1}^{n} w_j \left( \frac{x_j - X}{h} \right) \right] \left/ \left[ \frac{1}{n} \sum_{j=1}^{n} w_j \left( \frac{x - X_j}{h} \right) \right] \right.,
\]

(5)
Equation (5) provides an intuitive way to estimate $f$ at the point $x$. That is, all data contributes to the estimated value via a weighted average, with weight given by the data points distance from $x$. A data point, say $X_i$, that is far from $x$, simply will not contribute much to the estimate at $x$, provided we choose the smoothing parameter $h$ and the weight function $w$ so that $w((x - X_i)/h)$, which is the weighting term, is small when $x - X_i$ is large. Consider Fig. 1 in the light of this description: for a given value $X = x$, the estimate for $Y$ is mostly determined by those $Y$ values that correspond to $X$ values near $x$. This is a simple idea, but selecting the smoothing degree remains somewhat of an art despite attempts to automate the choice of bandwidth $h$. However, by using held-out testing sets, it is possible to do a reasonable job of automating the choice of $h$. Experience and theory suggest that the choice of $h$ is more critical than the choice of $w$. Typically, simple smooth functions such as a Gaussian-shaped function are a good choice. Theoretically optimal weight functions that it is wasteful to follow step 5 because many data points will never contribute to the rule base. It would be better to at least compute an average rule in order to resolve conflicting rules such as the two given in step 5. That average rule could be a member of the final fuzzy rule base, so that for a given rule describing the values of the predictors, there would be only one rule for the response. However, that one rule would be an average of all relevant data pairs rules. Also, even for cases where there are no conflicting rules, so no averaging is needed, the rule base contains the center value of the responses fuzzi region rather than the actual value of the response. That is also a waste of information. Also, the statistically motivated curve smoother uses some type of trial-and-error to select a good bandwidth. Therefore, our implementation of the fuzzy forecaster tries a number of fuzzy regions ranging from two to six. Typically, we see improved performance (lower average squared error) as the number of regions (for each predictor and for the response) increases, up to a point, and then worsening results as additional regions are added. In all cases, we report the best performance we could find. We also apply both triangular- and Gaussian-shaped regions and report the best results. The fuzzy forecaster presented in [1] considered the model $y = f(\hat{x})$ rather than $y = f(\hat{x}) + e$. Under that model, the fuzzy forecaster was shown to be capable of estimating any continuous to arbitrary accuracy, as can, for example, the class of polynomial functions. Surely, a better model for most data sets is $y = f(\hat{x}) + e$ and, under that model, the fuzzy forecaster wastes data. In addition, statistically motivated forecasters such as loess pay attention to the fundamental tradeoff between bias (caused by too much smoothing) and variance (caused by too little smoothing) at any point $x$ of the estimate $\hat{f}(x)$ of $f(x)$.

In short, under the $y = f(\hat{x}) + e$ model, the fuzzy forecaster is failing to use all of the data, and is incompletely using the data that it does use because it disregards the actual value of the response. Therefore, we implemented both the standard (wasteful) fuzzy forecaster and an improved version. Results presented are the best results (over the range of candidate number of regions and trying both triangular- and Gaussian-shaped regions) for the improved version. In all cases considered, the improved version outperformed the standard version, but was still inferior to the loess version for five of five simulated cases and four of five real cases.

In the Sections III and IV, we present results of the fuzzy forecaster and for loess because loess was available in S-Plus and because it was simple to program the fuzzy forecaster in S-Plus. It is also simple to program the nonparametric curve smoother in S-Plus, but to date we only have implemented a Fortran version.

### III. COMPARISON OF FUZZY FORECASTER TO LOESS FOR FIVE SIMULATED DATA SETS

The protocol for the simulated data experiments was as follows. Generate 400 observations of the response and predictors. Randomly select 200 to train and 200 to test. Repeat ten times and report the average and standard deviation of the average squared forecast error. By repeating ten times, we can compute approximate confidence intervals for the true mean-squared forecast error on each data set for each method.

**Data set 1:** $y = x(1 - x) + e$, where $e \sim U(0,1) \times 0.1$, where $U(0,1)$ denotes a uniform distribution over (0,1).

**Data set 2:** $x \sim U(0,1),$ and $y = \sin(2\pi(1-x)^2) + x + e$, where $e \sim N(0,1)$ and $N(0,1)$ denotes the normal distribution with mean zero and variance 1.

**Data set 3:** $x_1 \sim U(0.1), x_2 \sim U(0.1), \text{ and } y = 2x - 3x_1 + 0.1x_2 + e$, where $e \sim N(0,1)$.

**Data set 4:** $x_1 \sim U(0.1), x_2 \sim U(0.1), \text{ and } y = (2x - 3x_1 + x_1^2 + x_2 + 0.1x_2) + e$, where $e \sim N(0,1)$.
Fig. 2. Visual comparison of result of best fitting fuzzy forecaster to result of loess: (a) result with best number of triangular fuzzy regions, (b) result with best number of Gaussian fuzzy regions, and (c) result with loess.

Data set 5: $x_1 \sim U(0, 1), x_2 \sim U(0, 1),$ and, $x_3 \sim U(0, 1)$, and
\[ y = (2 \times x_1 - 3 \times x_2 + x_1 \times x_2 + x_3 + x_2 \times x_3 + x_1 \times x_3) + 0.1 \times e, \]
where $e \sim N(0, 1)$.

A rescaled version of data set 1 was shown in Fig. 1. Still working with data set 1, in Fig. 2 we show the loess fit and the fuzzy forecaster fit using triangular regions and using Gaussian regions. Clearly, the loess fit is superior to both of the best fitting fuzzy forecaster methods.

Following are the approximate 95% confidence intervals for the mean squared forecast errors over the testing set for the five data sets.

<table>
<thead>
<tr>
<th></th>
<th>FuzzyForecaster</th>
<th>Loess</th>
<th>TheoreticalBest</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(.002, .004)</td>
<td>(.0013, .0014)</td>
<td>.00083</td>
</tr>
<tr>
<td>2</td>
<td>(.42, .69)</td>
<td>(.40, .49)</td>
<td>difficult to calculate</td>
</tr>
<tr>
<td>3</td>
<td>(.029, .033)</td>
<td>(.0096, .01)</td>
<td>.01</td>
</tr>
<tr>
<td>4</td>
<td>(.025, .033)</td>
<td>(.010, .011)</td>
<td>.01</td>
</tr>
<tr>
<td>5</td>
<td>(.057, .065)</td>
<td>(.011, .013)</td>
<td>.01</td>
</tr>
</tbody>
</table>

The loess method outperformed the best fuzzy forecaster method in all five data sets, and achieved quite close to the theoretical
best in the four cases for which we have calculated the theoretical best.

IV. COMPARISON OF FUZZY FORECASTER TO LOESS FOR FIVE REAL DATA SETS

The protocol for the real data sets was to use 50% to train, 50% to test, and to repeat ten times. As in Section III, repeating the experiment ten times allows us to report approximate confidence intervals for the true forecast error mean squared error.

Data set 1: This is the same data set that was analyzed in [7, Section V-A2]. The response is the gamma count at time \( t \). Predictors are gamma count at time \( t-1 \), gamma count at time \( t-2 \), electron count at time \( t \), and proton count at time \( t \). We also tried using higher lags for all predictor series, but there was no reduction in the average squared forecast error (four predictors, \( n = 878 \)).

Data set 2: Use NOx (nitric oxide and nitrogen dioxide combined) to predict the equivalence ratio at which the car engine was run (a measure of the richness of the air/ethanol mix) (1 predictor, \( n = 88 \)).

Data set 3: Use radiation, temperature, and wind to predict atmospheric ozone (three predictors, \( n = 111 \)).

Data set 4: Use previous days bond yield of type A and present days bond yield of type B to predict present days bond yield of type A (two predictors, \( n = 192 \)).

Data set 5: Use a cars weight and engine displacement to predict gallons/mile (two predictors, \( n = 60 \)).

Following are the approximate 95% confidence intervals for the mean-squared forecast errors over the testing set for the five data sets.

<table>
<thead>
<tr>
<th>Fuzzy Forecaster</th>
<th>Loess</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (55798, 60652)</td>
<td>(40584, 42810)</td>
</tr>
<tr>
<td>2. (.24, .36)</td>
<td>(.23, .32)</td>
</tr>
<tr>
<td>3. (.24, .29)</td>
<td>(.23, .27)</td>
</tr>
<tr>
<td>4. (711, 4921) \times 10^{-7}</td>
<td>(1.6, 2.0) \times 10^{-7}</td>
</tr>
<tr>
<td>5. (.13, .21)</td>
<td>(.14, .21)</td>
</tr>
</tbody>
</table>

So, loess is better on data sets 1–4 while the fuzzy forecaster is very slightly better on data set 5. Actually, there is little difference between the two on data sets 2, 3, and 4, but with data sets 1 and 4, loess does significantly better. Recall that we are presenting the best results we found for the fuzzy forecaster (an idea we have not found in any of the fuzzy logic literature), searching for the best number of regions and whether to use triangular or Gaussian regions.

V. SUMMARY OF THE VECTOR AR MODELS CONSIDERED

Recall that we are considering vector AR models in the context of time series so that forecasting time series can be viewed as an ordinary regression problem. Though it may be possible (see [4] for an ad hoc two-step procedure that is under investigation) to treat nonlinear MA models, we have not attempted that here. Once we restrict attention to AR models, we get access to a host of regression techniques that can be applied as if the data were in the usual regression setting: observe independent cases of data “pairs,” \( (x, y) \) where the \( x \) vector is a \( p \)-component predictor vector. The only difference in our setting is that successive cases are not independent because of the serial correlation. However, asymptotically (as the number of cases increases), this serial correlation can be ignored for the purpose of function estimation [4].

VI. SUMMARY OF THE PERFORMANCES OF FUZZY FORECASTER AND LOESS

First, recall that we have improved the fuzzy forecaster in three ways.

1) The fuzzy-rule base uses all of the training data.
2) The fuzzy-rule is calculated by averaging the actual value of the response rather than the center value of the fuzzy region to which the response has maximum membership. By only recording the center value, information is needlessly being thrown away with the fuzzy forecaster of [1].
3) We use trial-and-error to choose the number and type of fuzzy regions so we can explicitly optimize a particular criterion, which in this paper has been the average-squared forecast error.

When we undertook this study, we knew that the statistically motivated methods such as our curve smoother or loess would outperform the fuzzy forecaster on simulated data of the types presented here. That is obvious if for no other reason than the fuzzy forecaster does not use all of the training data. One reason that the fuzzy-forecaster does not use all of the data was that it was developed for models such as the one in (4), but without the stochastic component. That is, it was developed for deterministic models. We find that ironic because fuzzy logic proponents claim that fuzzy logic offers a new and complementary way to model uncertainty. See [8] for more of the ongoing debate on the topic. In the case of the fuzzy forecaster on simulated data, we conclude that fuzzy logic offers an inferior way to model uncertainty. That is, it simply ignores uncertainty. Perhaps sometimes there is so much training data that it is acceptable to ignore part of it, but that is almost never our experience. Nevertheless, we did not know how the fuzzy forecaster would compare to, for example, loess, on real data because real data never exactly follows any simulated model. The tentative conclusion here (based on five real data sets) is that loess or similar statistical methods are superior to the fuzzy forecaster in most situations, whether with real data or simulated data. Fuzzy-logic-based methods should be reserved for situations with nonnumeric data, and even in some of those cases, statistical methods are likely to be more mature, and therefore, at least for now, superior. We have not addressed that issue in this paper. Also, we remind the reader that the simpler methods such as global linear or global polynomial should always be applied first, and if the data is nearly linear or polynomial in form, then no other method will be as good as the linear or polynomial method.

REFERENCES

Digital Production of Color Mach Bands Using a Color Human Visual System Model

Curtis E. Martin, Steven K. Rogers, and Matthew Kabrisky

Abstract—A model of the human color vision system is used to construct constant brightness color Mach band stimuli along two chromatic axes for display on a standard workstation color monitor. Chromatic (not brightness) Mach bands appear in the displayed stimuli as over- and under-shoots along the axis of the varied component. The stimuli are displayed on several different but similar monitors without affecting the appearance of the Mach bands. Anomalous trichromats see no change in color when presented with a stimulus containing variations along only one of the two chromatic axes.

Index Terms—Color Mach bands, human visual system models.

I. INTRODUCTION

Since first being described in 1865, Mach bands have been one of several illusions used to show that perceived brightness measured by the human visual system (HVS) is not a simple function of incident intensity [1]. Mach bands are regions of increased or decreased perceived brightness which appear at locations where a luminance gradient meets a plateau, as illustrated in Fig. 1. Much effort has been expended to determine the causes of this illusion, in terms of both the stimulus conditions and the HVS mechanisms which bring it about [3]. While this work has been largely successful in outlining stimulus conditions which produce the Mach band illusion, as well as those which defeat it, there remains some disagreement as to what HVS mechanisms actually cause Mach bands to appear (see [4]–[7]).

Expanding consideration of the HVS to include color perception, it is reasonable to ask whether or not Mach-type phenomena can be produced using colored stimuli. Investigation of this question has caused something of a controversy: a number of researchers claim to have produced color Mach bands, despite a general conclusion that they do not exist. A summary of this controversy is given by Pease [8], who suggests that this disagreement is due at least in part to a poor specification of what is meant by the term “color Mach band,” and proposes a distinction between brightness, hue, and saturation effects [8]. Pease argues that such a distinction in perceptual terms is both appropriate, because Mach bands arise as a result of human perception, and necessary, for clarity in the discussion of the phenomenon. Based on his own review of the literature, he concludes that brightness and saturation Mach bands have been demonstrated, but that there is no evidence for the existence of hue Mach bands [8].

Keeping in mind the distinction suggested by Pease, this paper considers the question of color Mach bands within the context of a perceptual HVS model. It is commonly accepted that separate channels exist in the HVS for brightness and chromatic visual information [9]. Assuming that the brightness channel mediates the brightness Mach band illusion, it is reasonable to wonder if a Mach band illusion may be produced by constructing uniform brightness displays which only vary in the chromatic channels. Mach bands arising from such stimuli would be accurately termed “chromatic Mach bands,” and may be manifest as a combination of hue and saturation effects [8]. Just as brightness Mach bands appear as a perceived shift along a brightness axis, chromatic Mach bands should appear as a perceived shift in color along a chromatic axis.

This paper presents an approach to creating chromatic Mach bands on a standard computer workstation color display. Bands with the expected color shift are observed in the stimulus, and the appearance of these bands suggests that they could in fact be hue Mach bands. However, for the purposes of this paper, they will simply be called chromatic Mach bands to indicate the absence of brightness changes in the stimulus that could give rise to the bands. More careful analysis of the chromatic Mach bands is required in order to describe them in the terms used by Pease.

Section II provides a context for the current experiment by describing what a chromatic Mach band should look like, reviewing pertinent results already obtained, and presenting a color HVS model used in generating the stimulus. The production of the stimulus on a standard workstation color monitor and the appearance of the illusion is described in Section III. Finally, Section IV discusses the implications of these results.

II. BACKGROUND

This section provides a working definition or description of chromatic Mach bands, summarizes the pertinent literature on the search for color Mach bands, and describes the color HVS model used in this paper for producing chromatic Mach bands.


Fig. 1. Classic Mach band illusion. The density distribution in the top image is as shown in the middle plot, yet the apparent brightness of the stimulus is similar to that shown in the bottom plot, with bright and dark bands appearing at the knee points where the ramp meets the uniform areas (adapted from [2]).

Manuscript received March 17, 1996. This work was supported in part by the Defense Dissemination Program Office.

C. E. Martin and M. Kabrisky are with the Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright Patterson AFB, OH 45433 USA.

S. K. Rogers is with the Cognitive Systems Group, Battelle Memorial Institute, Columbus, OH 43201-2693.

Publisher Item Identifier S 1083–4427(98)00128-3.