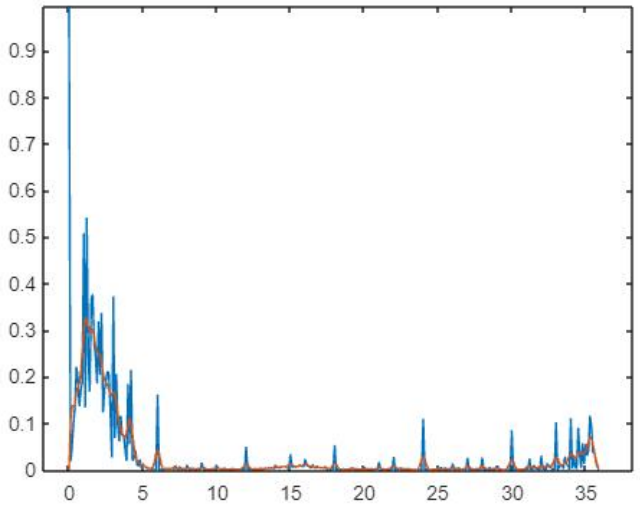
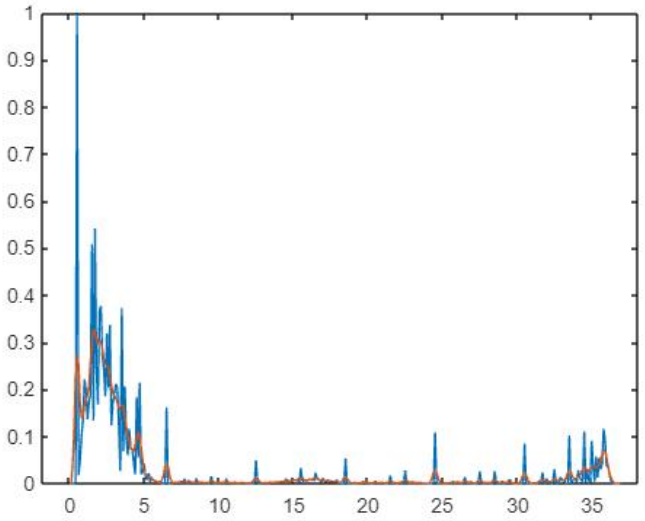
Moving Average(MA): Low pass filtering of data
https://zhuanlan.zhihu.com/p/38276041

1. Treat as continuous range
   Nelder-Mead-Method --> "fminsearch.m"
   Lambda function --> use “fitgauss.m”
2. local maximum
3. Correctness checking (Support vector machine)

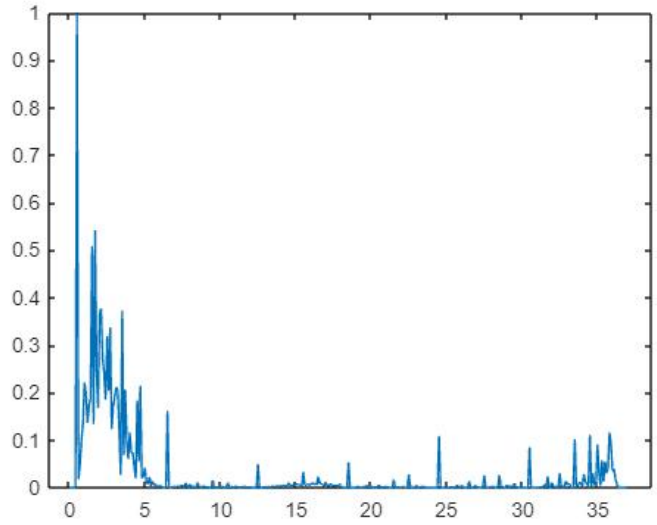
1. Treat as “continuous range”:

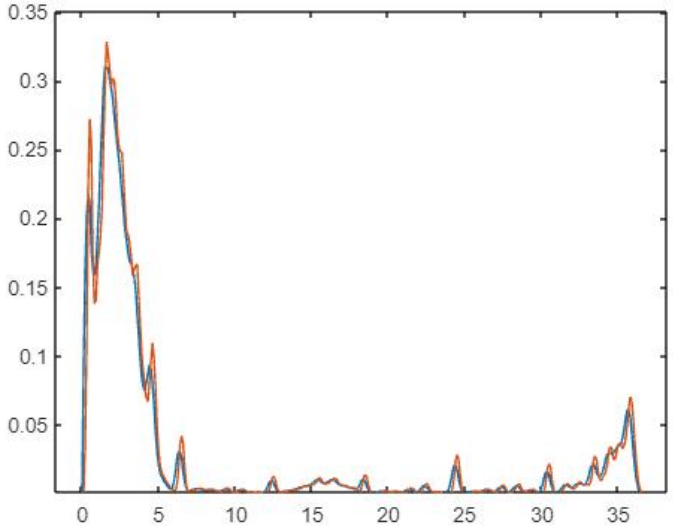
Step 0: Data zero padding





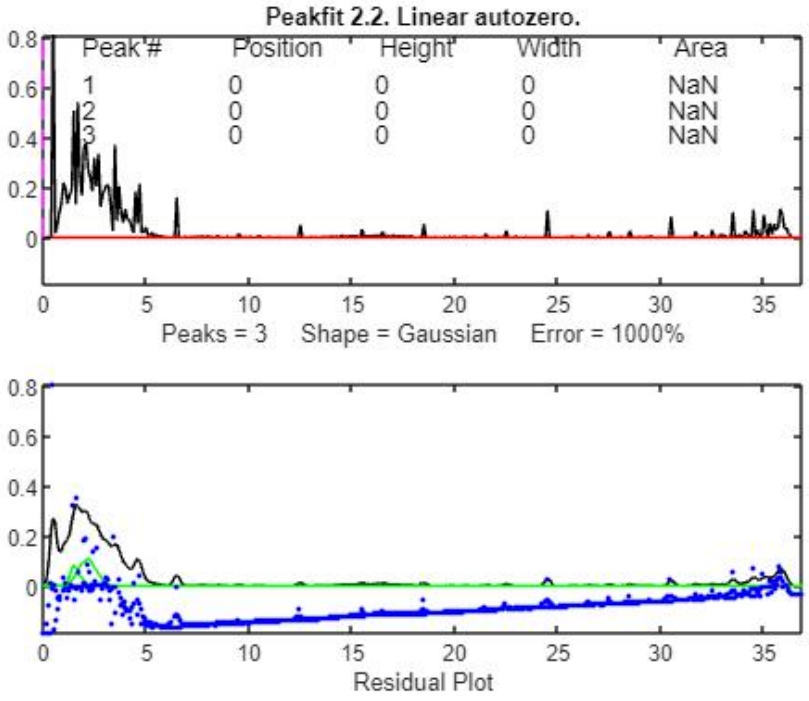
Step 1: Smoothing low-pass filter



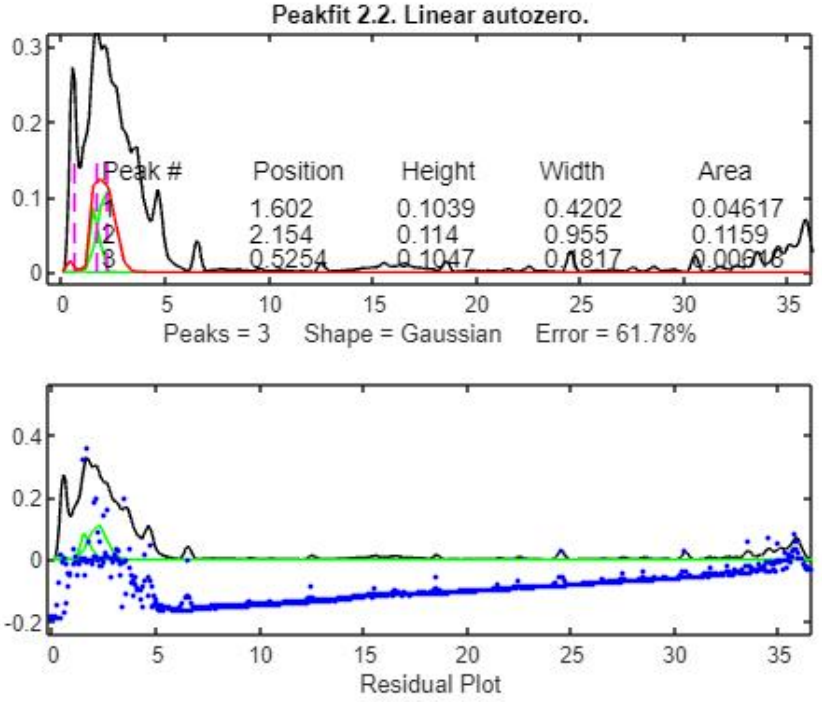


1. Treat as “continuous range”:

Step 1: Smoothing low-pass filter <-- REASON



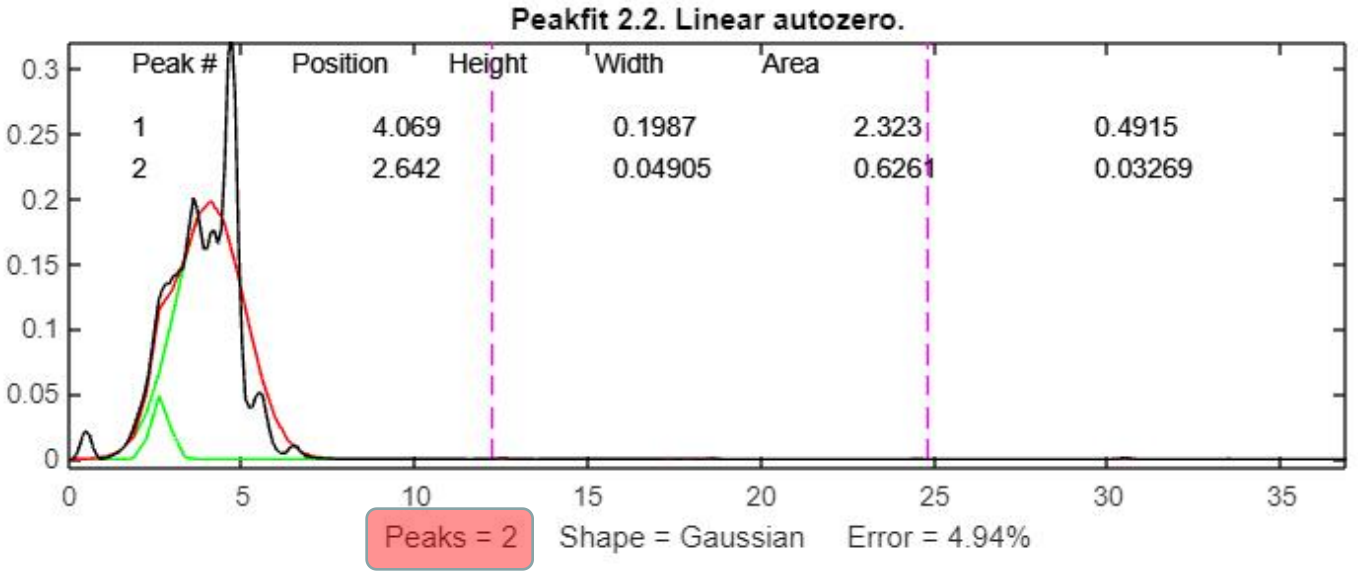
Before Smoothing

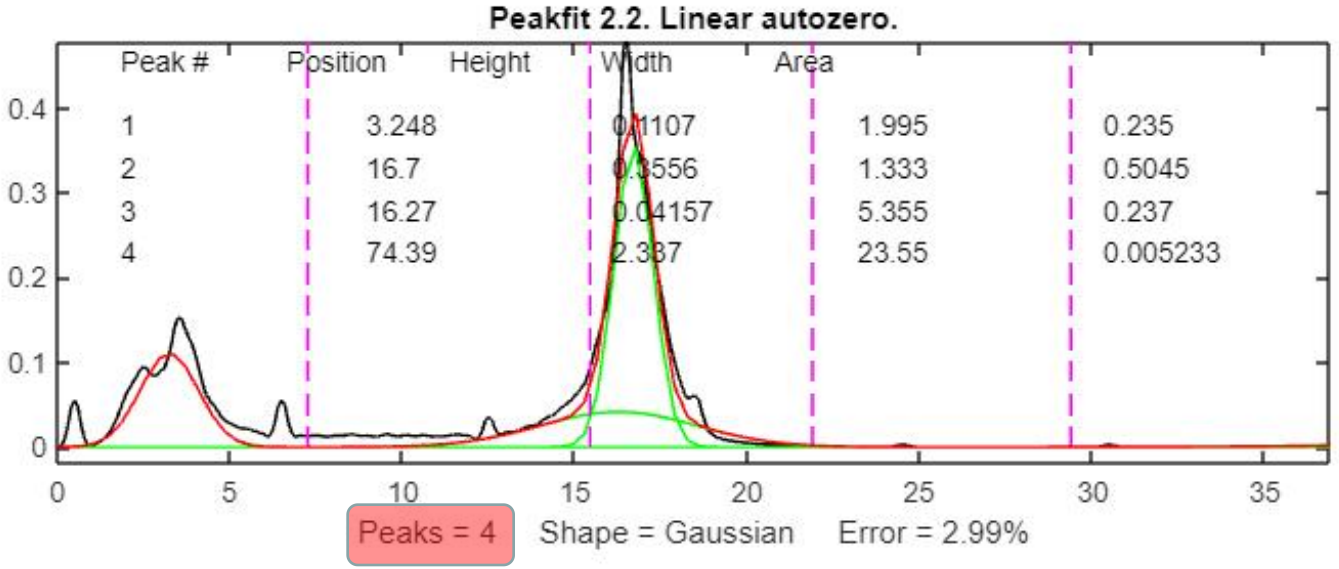


After Smoothing

1. Treat as continuous range：

Step 2: Decide the continuous range





1. Treat as “continuous range”:

Technique explaination

Step1. Nelder-Mead-Method --> "fminsearch"

fminsearch was originally designed for finding the minimum values of functions, but it can be applied to least-squares curve fitting by creating an anonymous function (a.k.a. "lambda" function) that computes the model, compares it to the data, and returns the fitting error.

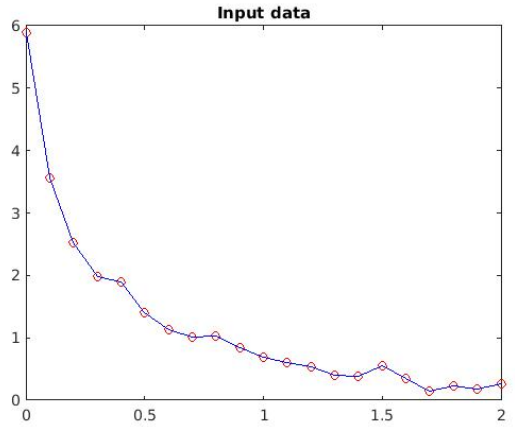
For example, writing:

parameters = fminsearch(@(lambda)(fitfunction(lambda,x,y)),start)

performs an iterative fit of the data in the vectors x,y to a model described in a previously-created function called fitfunction, using the first guesses in the vector start. The parameters of the best-fit model are returned in the vector "parameters", in the same order that they appear in "start".

First, create some sample data and plot it.

```
t = (0:.1:2)';
y = [5.8955 3.5639 2.5173 1.9790 1.8990 1.3938 1.1359 1.0096 1.0343 ...
    0.8435 0.6856 0.6100 0.5392 0.3946 0.3903 0.5474 0.3459 0.1370 ...
    0.2211 0.1704 0.2636]';
plot(t,y,'ro'); hold on; h = plot(t,y,'b'); hold off;
title('Input data'); ylim([0 6])
```



The goal is to fit the following function with two linear parameters and two nonlinear parameters to the data:

$$y = C(1)*\exp(-\lambda(1)*t) + C(2)*\exp(-\lambda(2)*t)$$

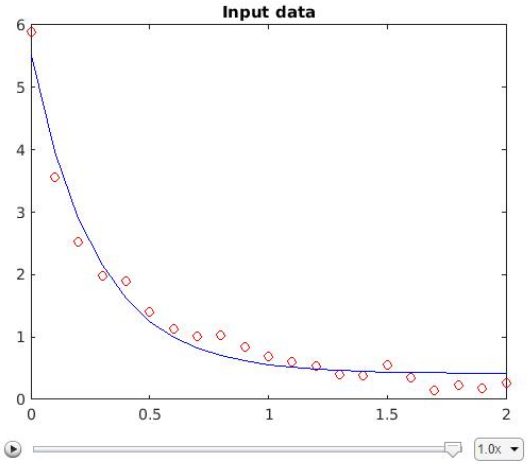
To fit this function, we've create a function FITFUN. Given the nonlinear parameter (lambda) and the data (t and y), FITFUN calculates the error in the fit for this equation and updates the line (h).

```
type fitfun
```

```
function err = fitfun(lambda,t,y,handle)
%FITFUN Used by FITDEMO.
%   FITFUN(lambda,t,y,handle) returns the error between the data and the values
%   computed by the current function of lambda.
%
%   FITFUN assumes a function of the form
%
%     y =  c(1)*exp(-lambda(1)*t) + ... + c(n)*exp(-lambda(n)*t)
%
%   with n linear parameters and n nonlinear parameters.

%   Copyright 1984-2002 The MathWorks, Inc.
%   $Revision: 5.8 $  $Date: 2002/04/08 20:04:42 $

A = zeros(length(t),length(lambda));
for j = 1:length(lambda)
   A(:,j) = exp(-lambda(j)*t);
end
c = A\y;
z = A*c;
err = norm(z-y);

set(gcf,'DoubleBuffer','on');
set(handle,'ydata',z)
drawnow
pause(.04)
```

Make a guess for initial estimate of lambda (start) and invoke FMINSEARCH. It minimizes the error returned from FITFUN by adjusting lambda. It returns the final value of lambda.

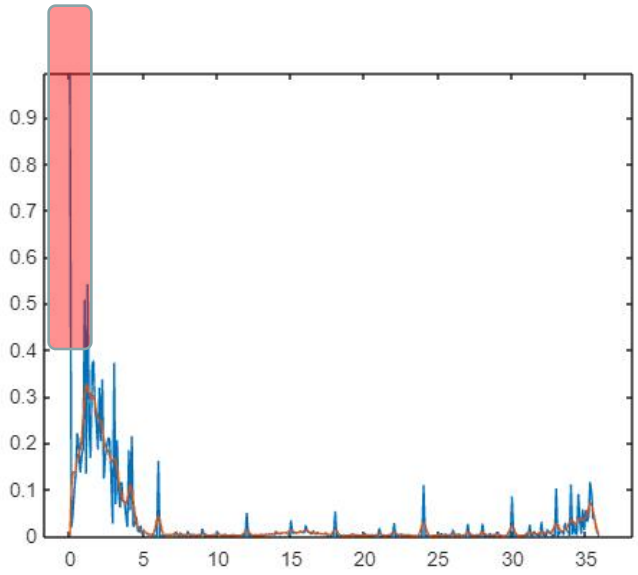
```
start = [1;0];
options = optimset('TolX',0.1);
estimated_lambda = fminsearch('fitfun',start,options,t,y,h)
```

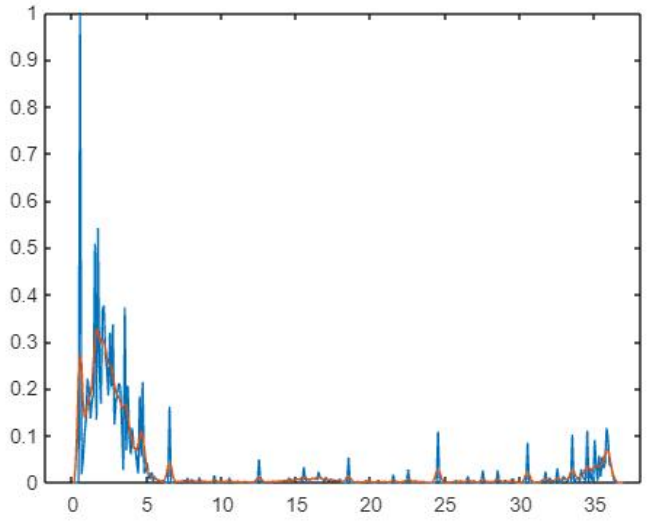


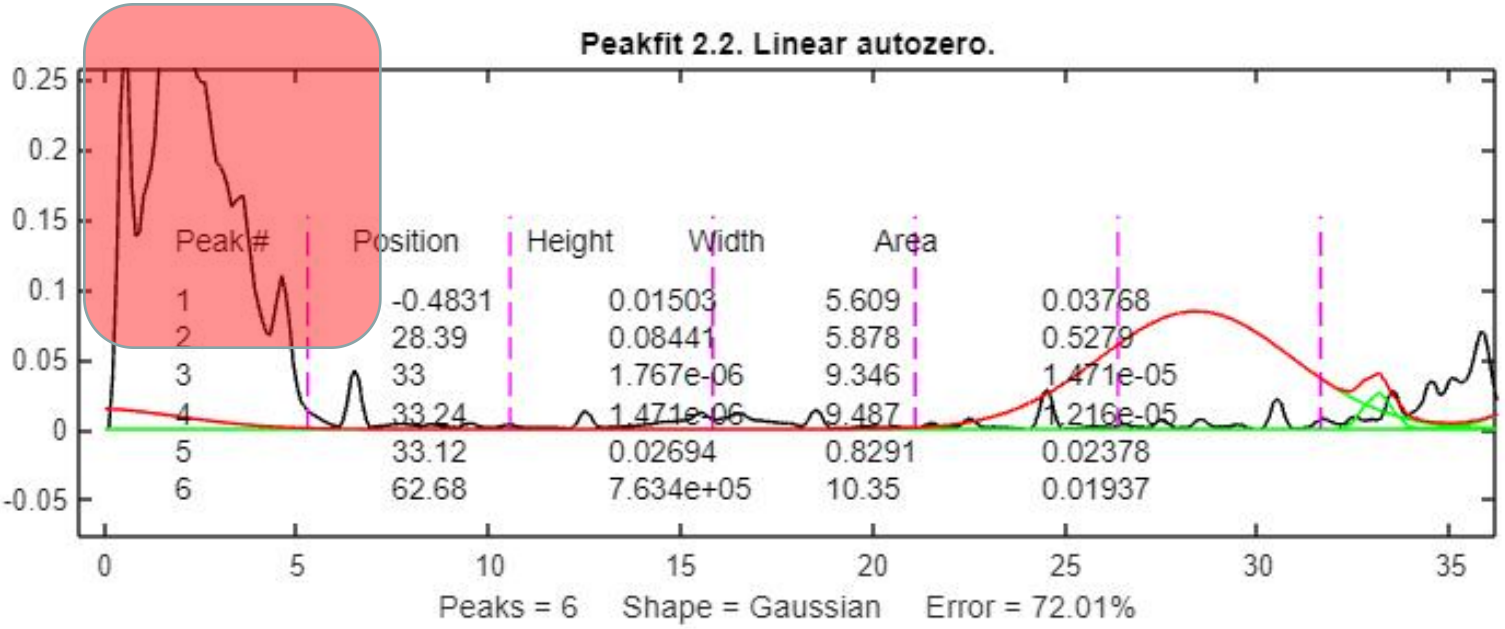

```
estimated_lambda = 2×1
    3.5897
    0.0030
```

2. Local maximum：

Problem 1: Bad matching for edge even after doing Data zero padding

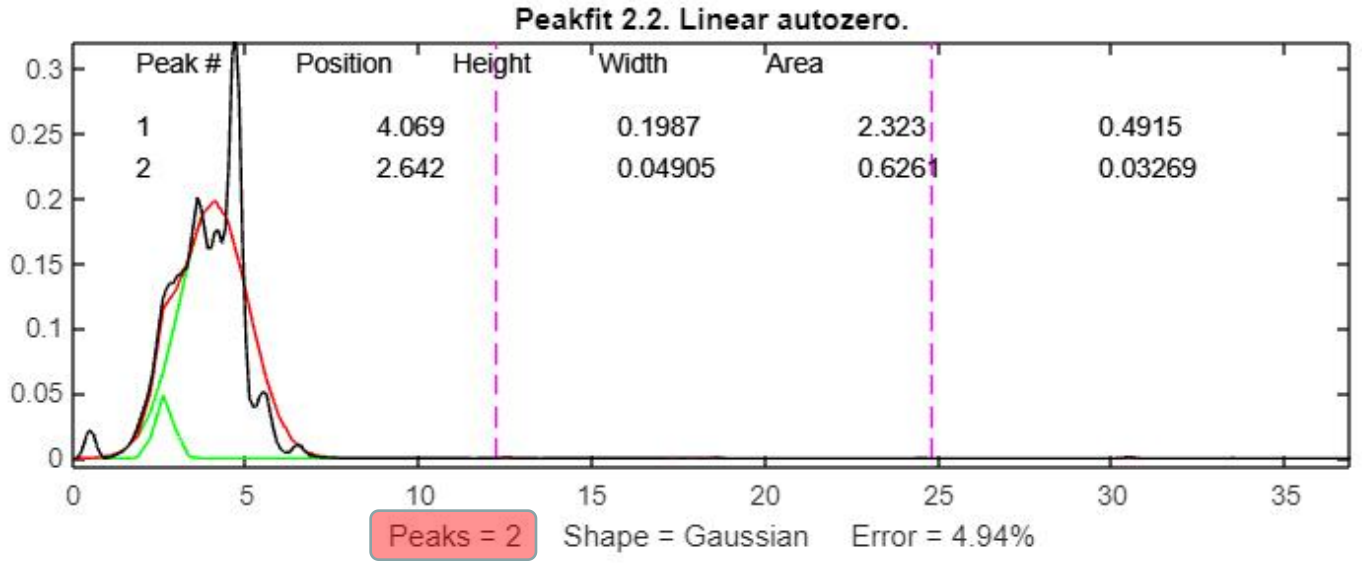


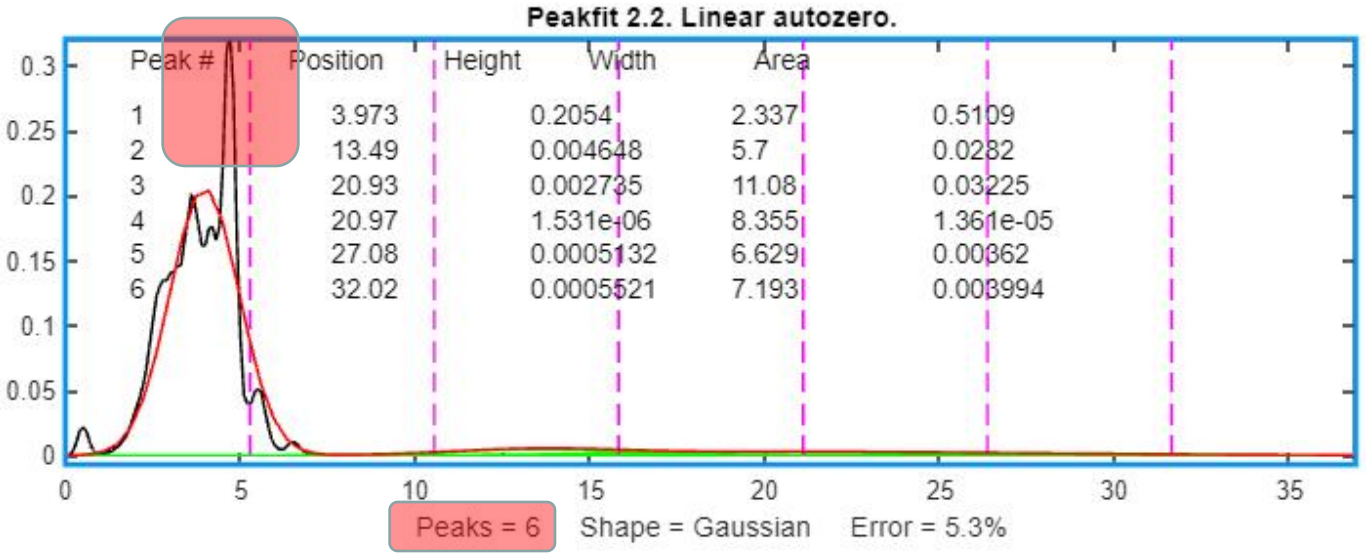




2. Local maximum：

Problem 2: Inaccurate matching for local max even with larger amout peak matching

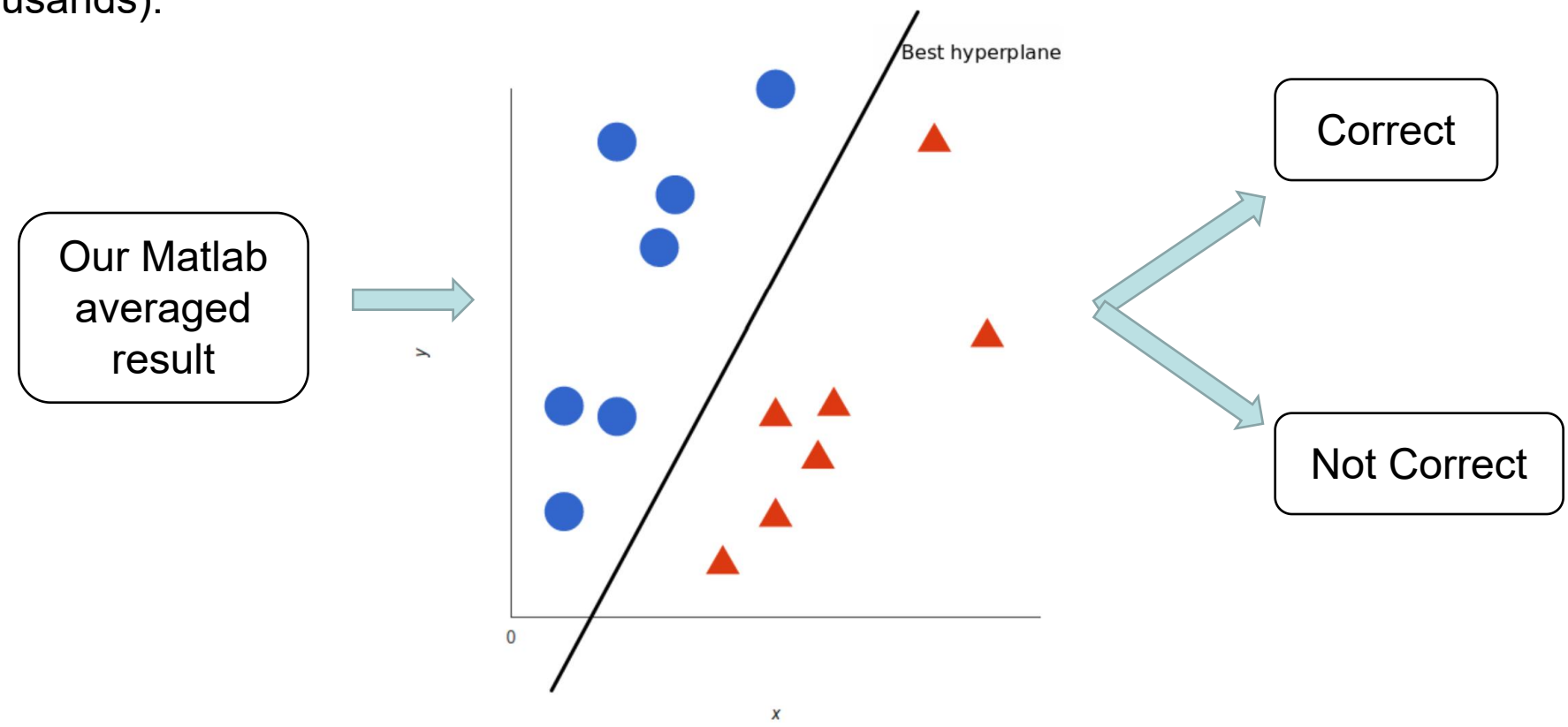




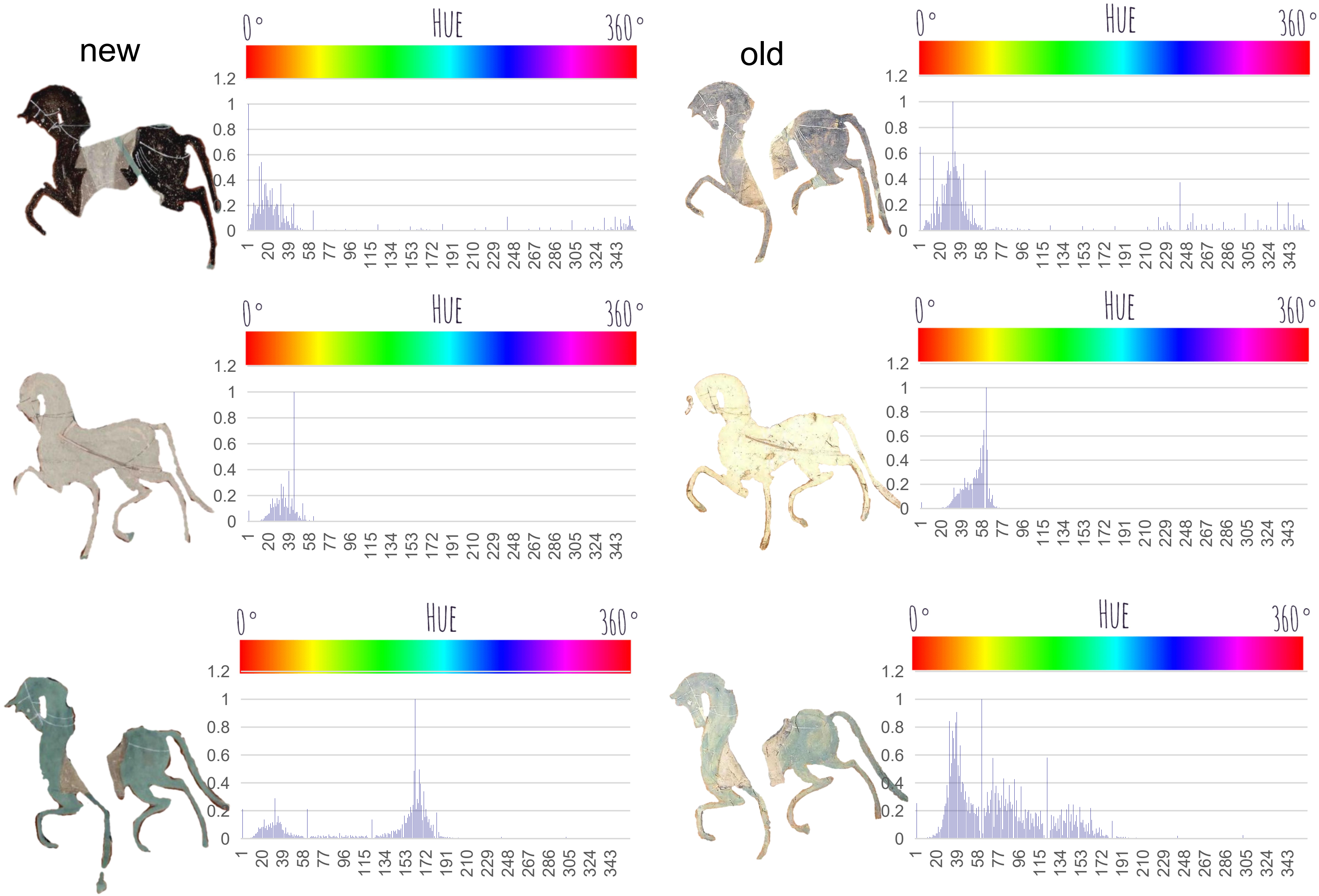
3. SVM -- Correctness checking：

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they're able to categorize new input.

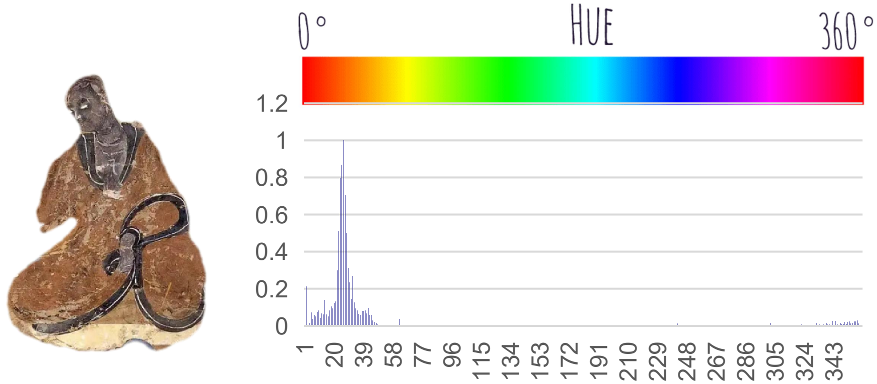
Compared to newer algorithms like neural networks, they have two main advantages: higher speed and better performance with a limited number of samples (in the thousands).

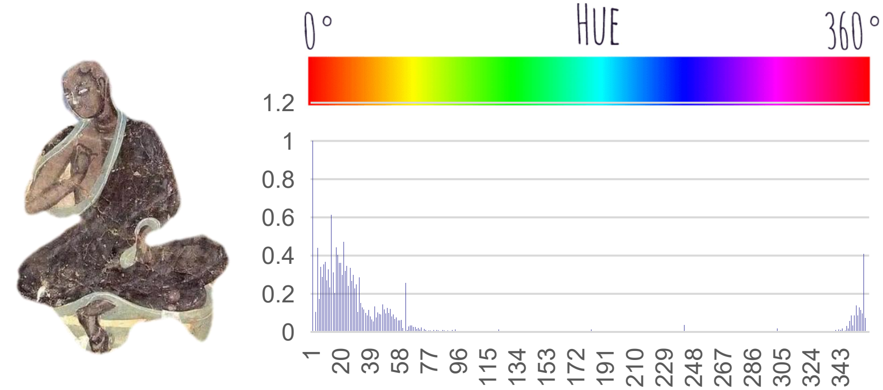


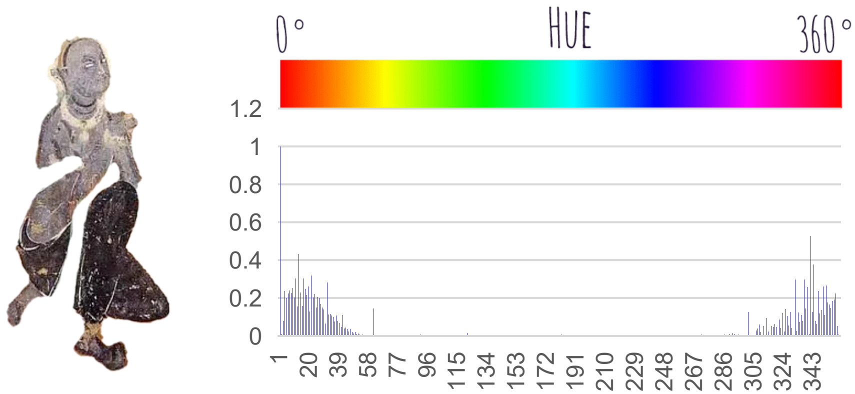
Picture Demo #1: Horse

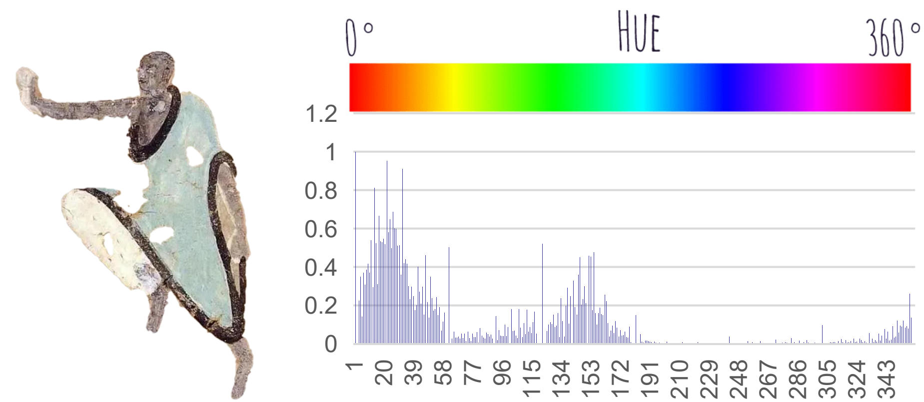


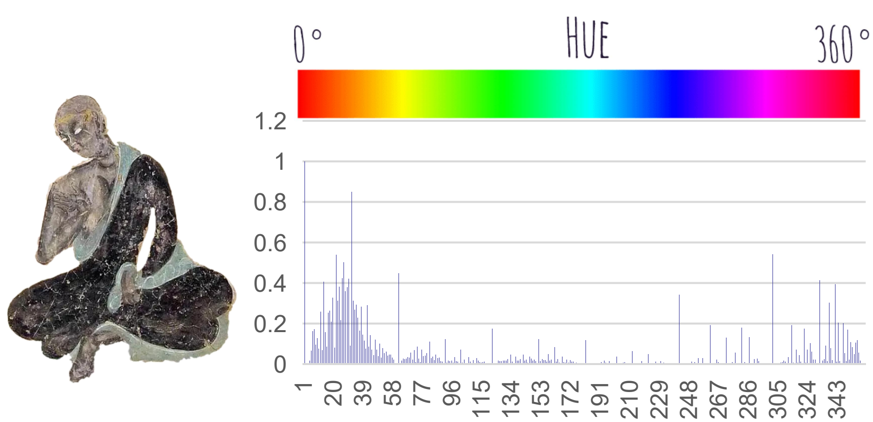
Picture Demo #2: Man

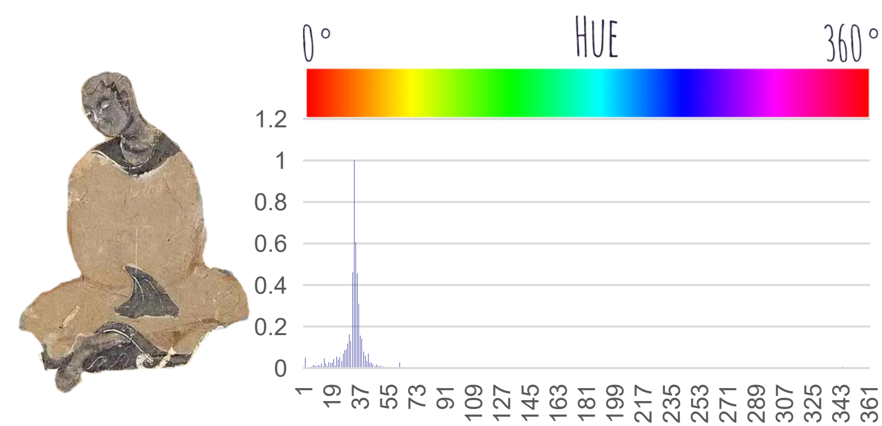




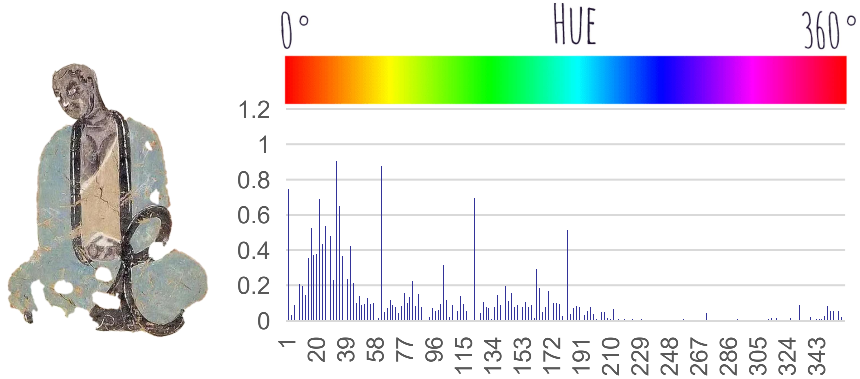


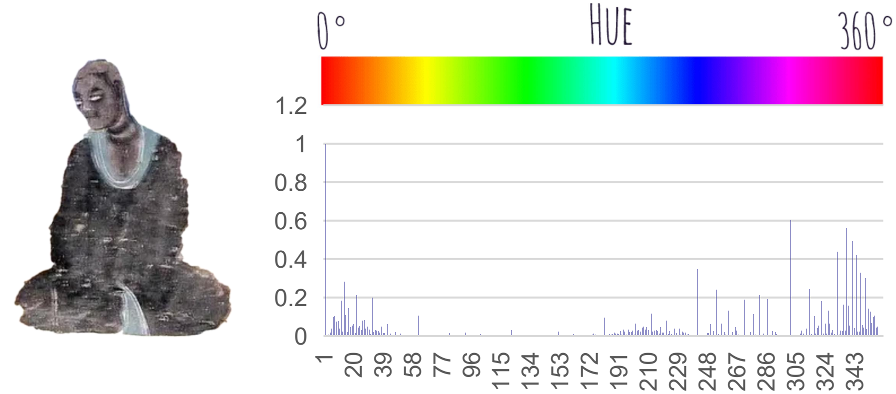


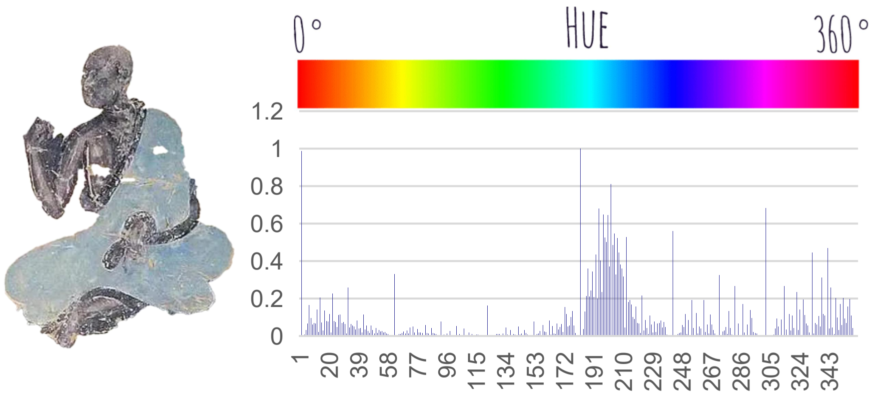


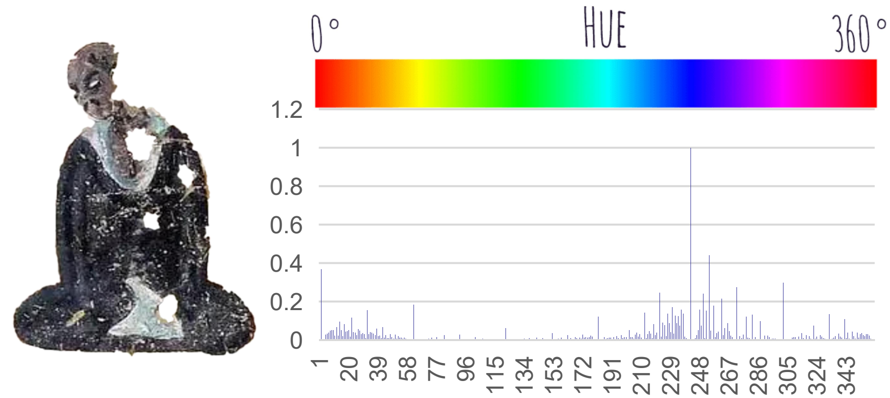


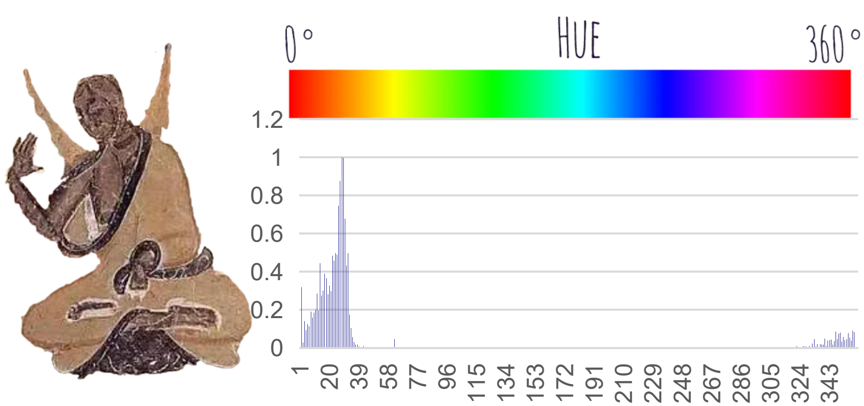
Picture Demo #2: Man

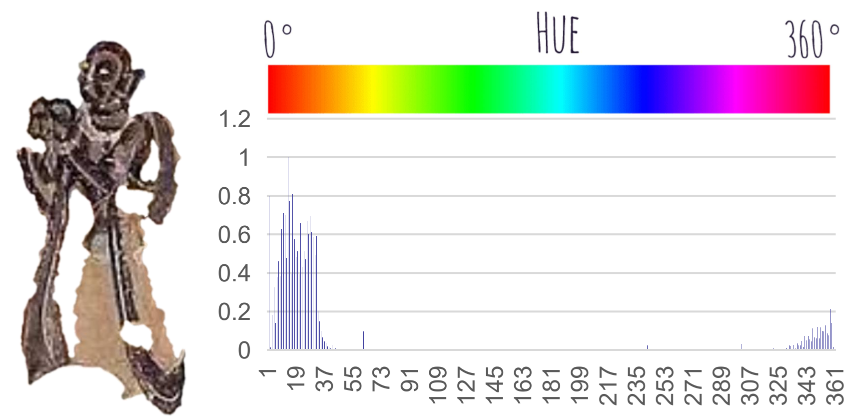




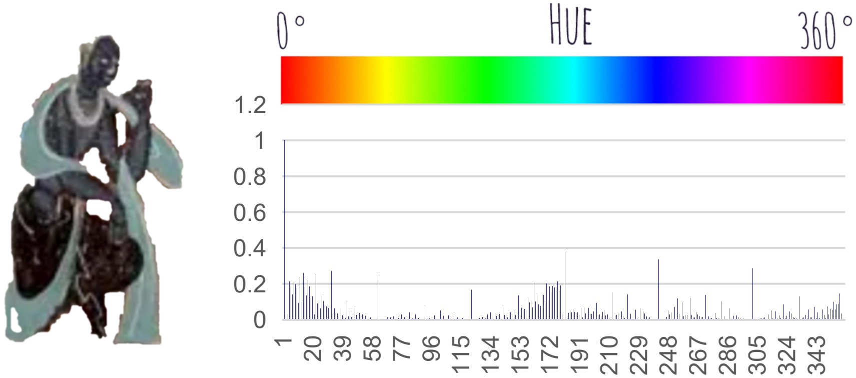


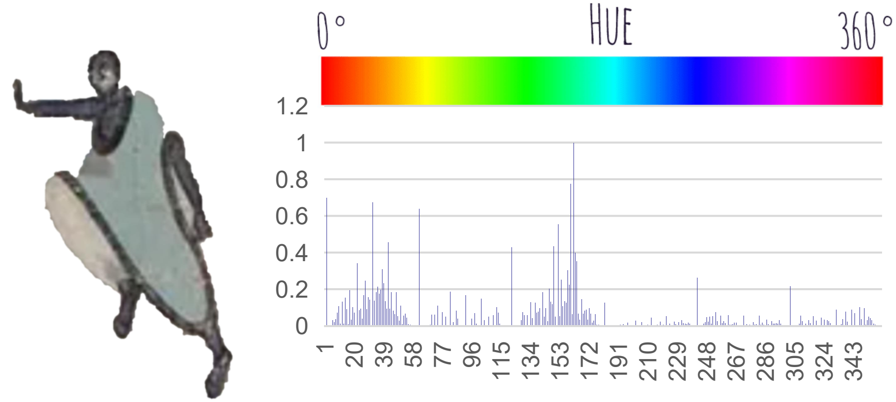


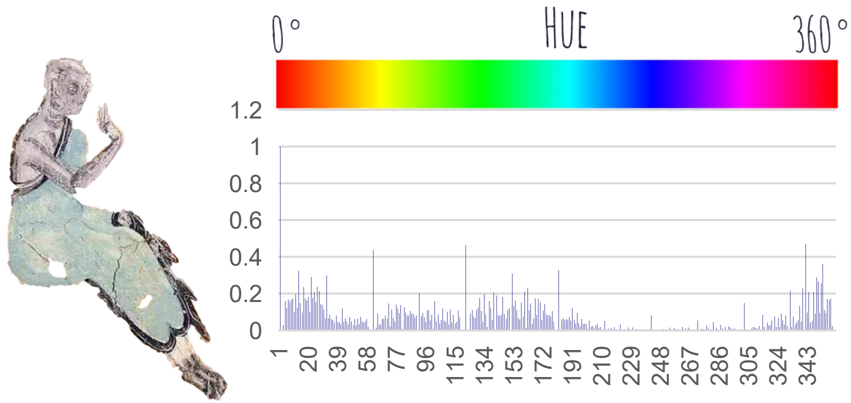


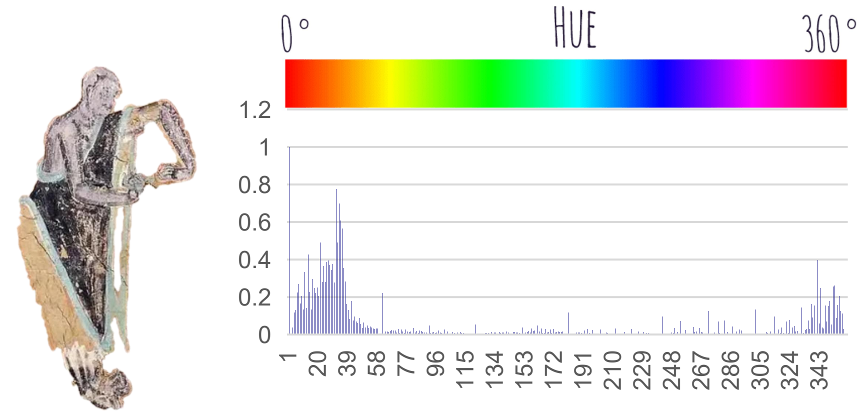


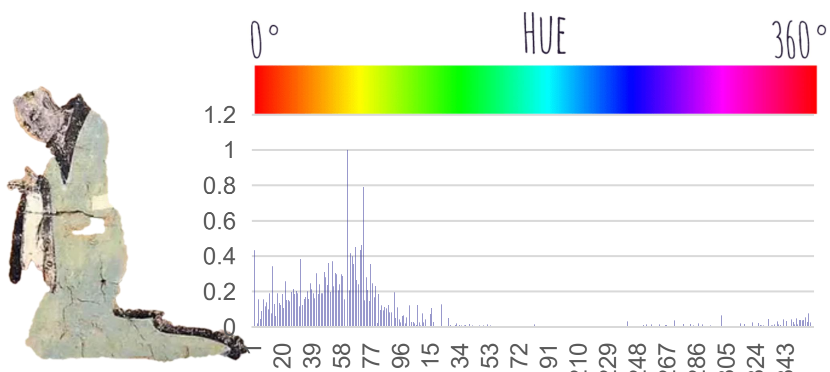
Picture Demo #2: Man

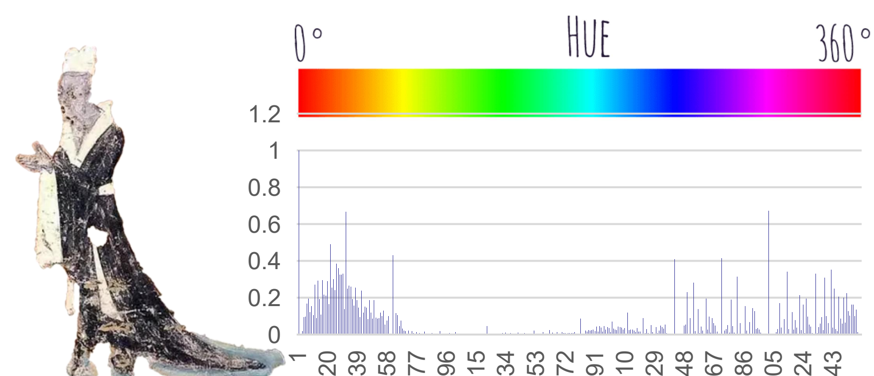




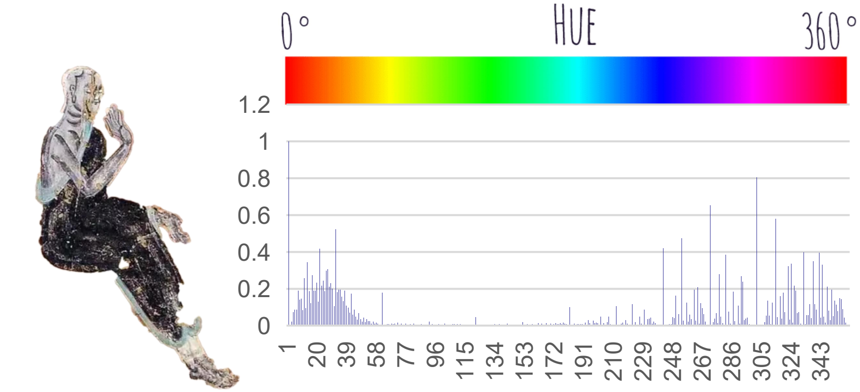


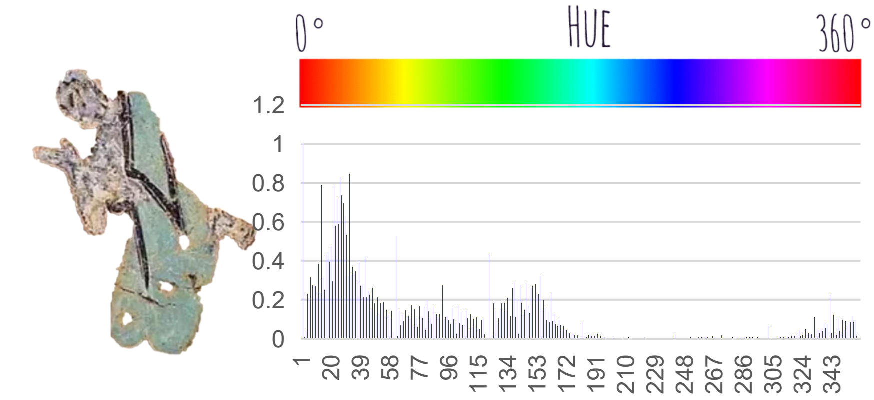


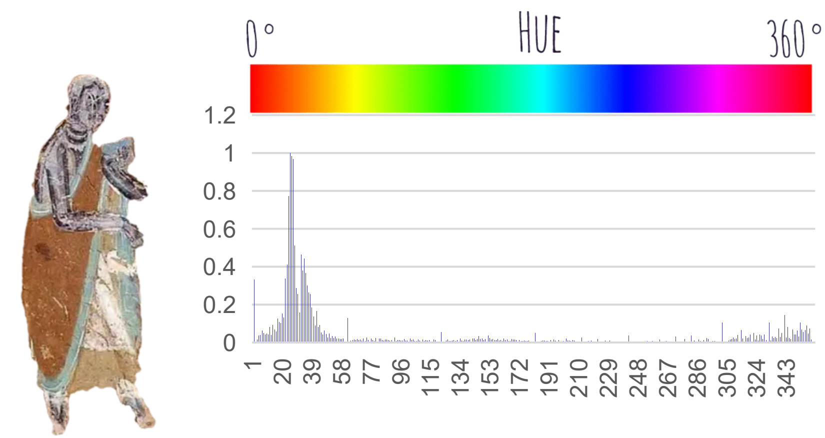


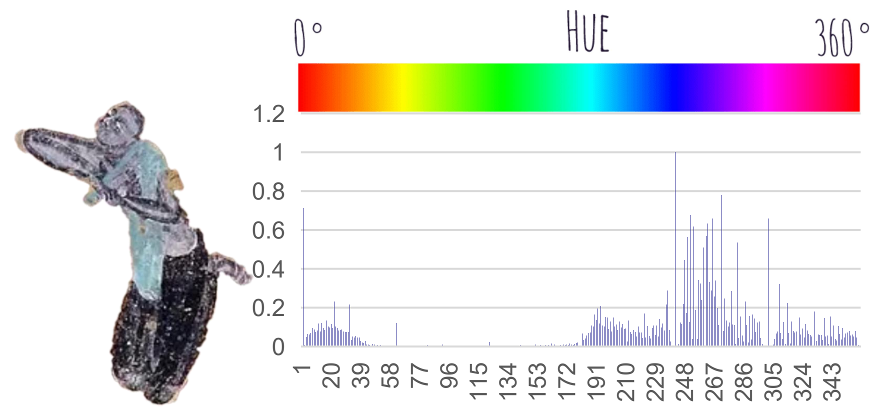


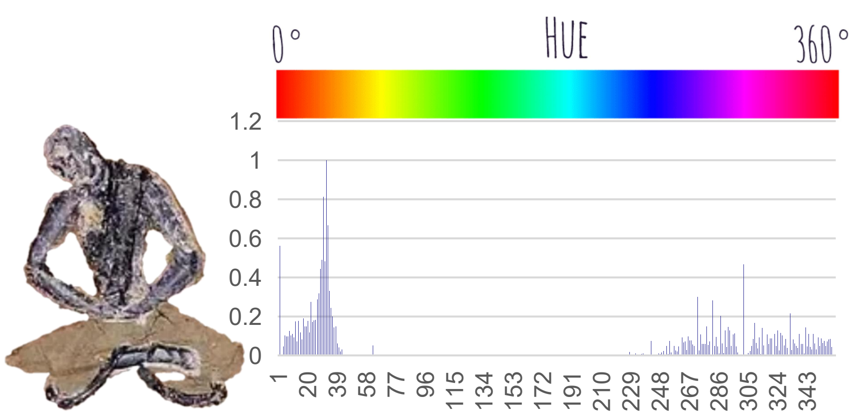
Picture Demo #2: Man

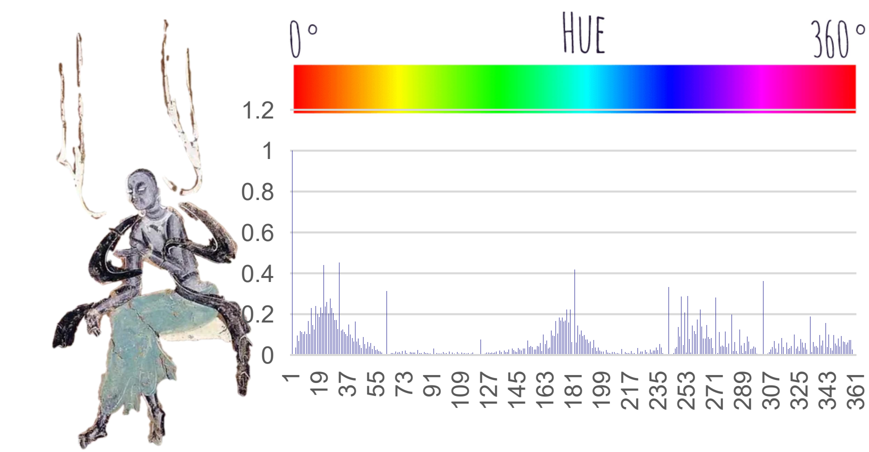




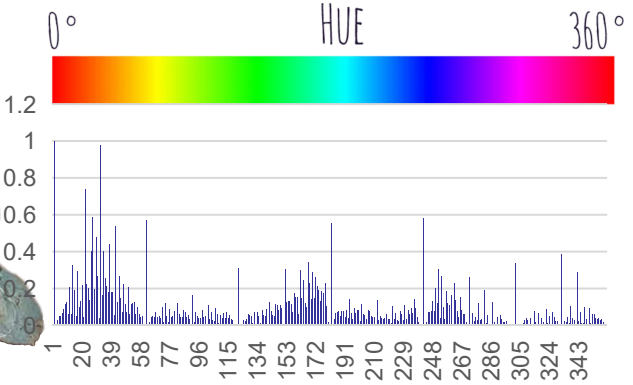


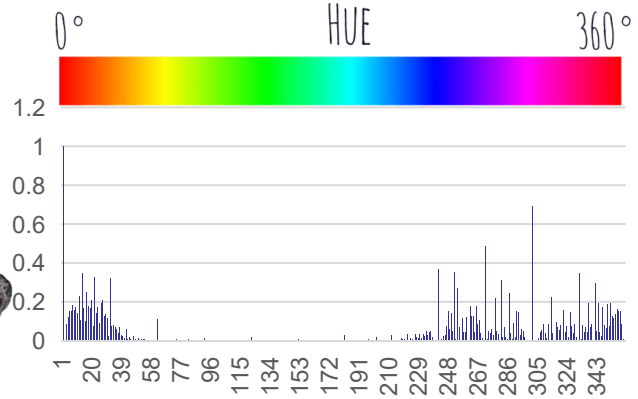


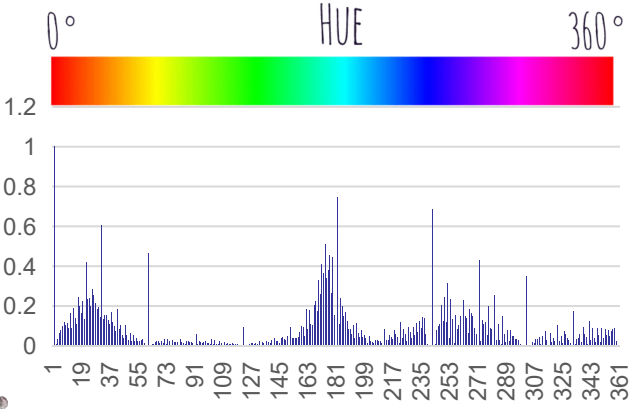


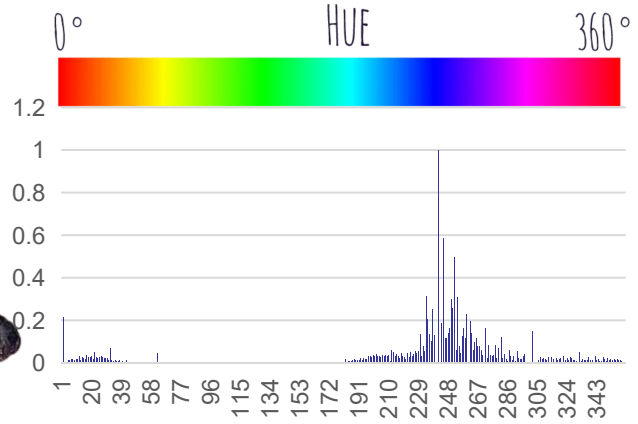


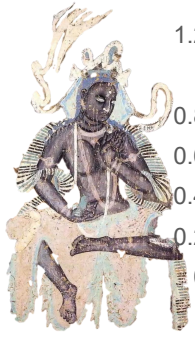
Picture Demo #2: Man

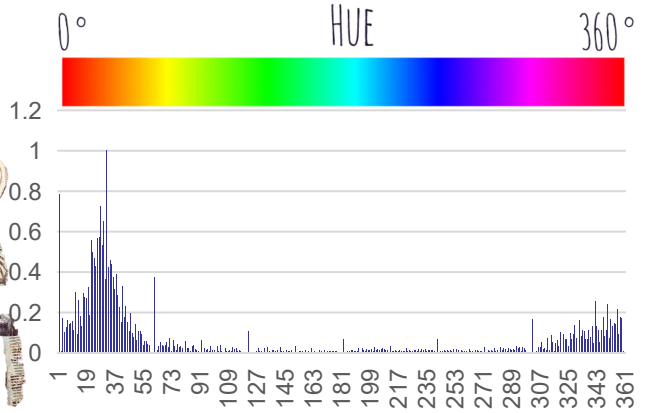


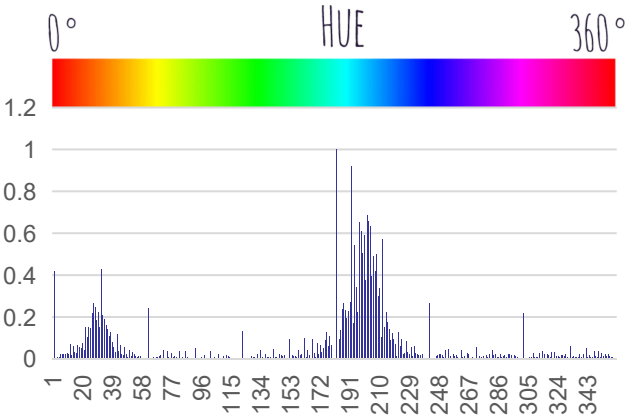




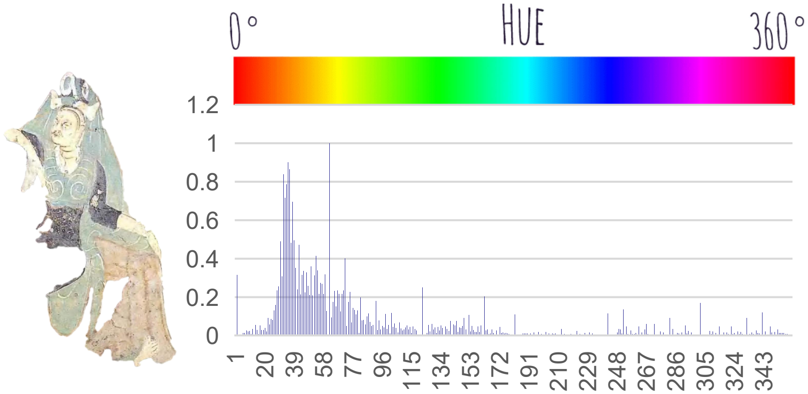


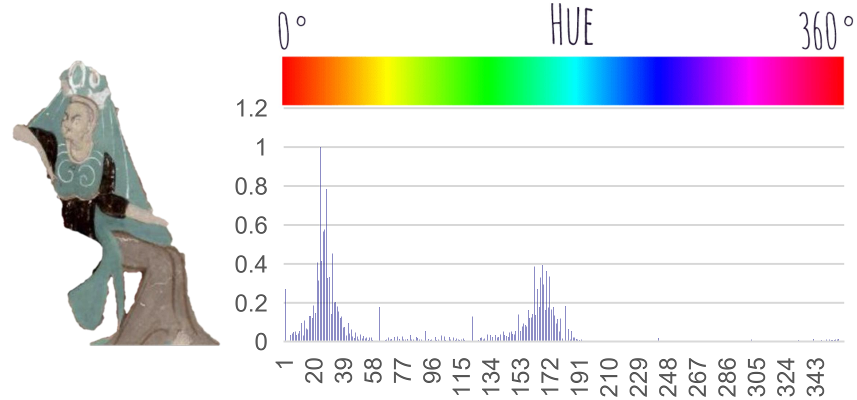


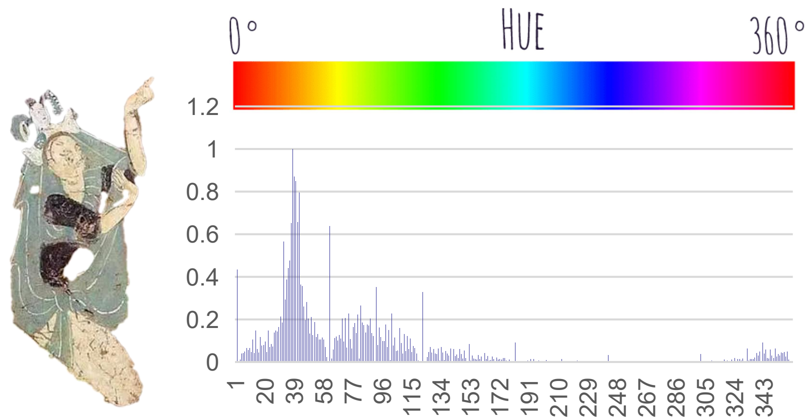


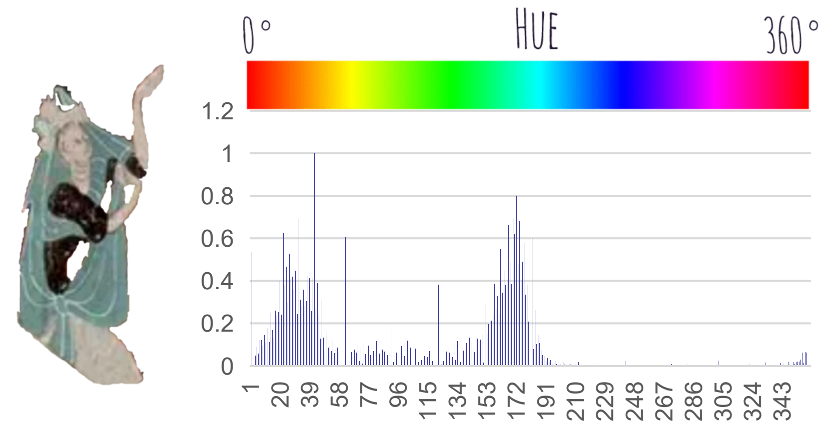


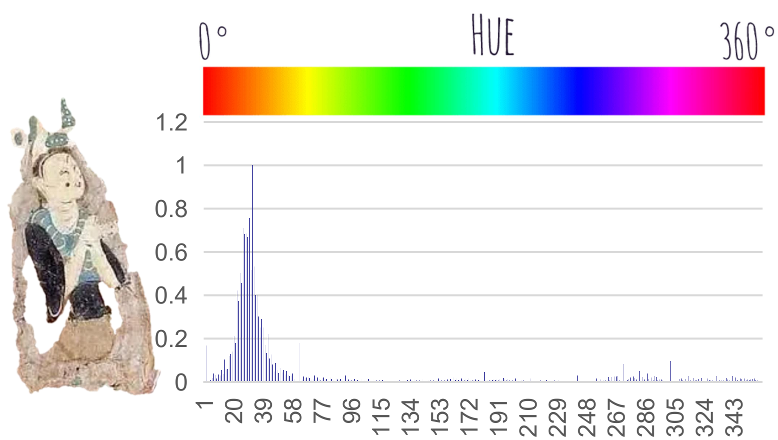
Picture Demo #3: Woman

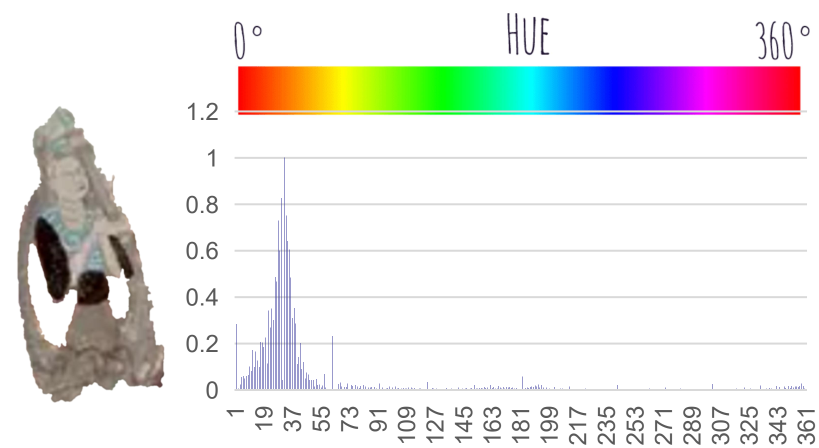




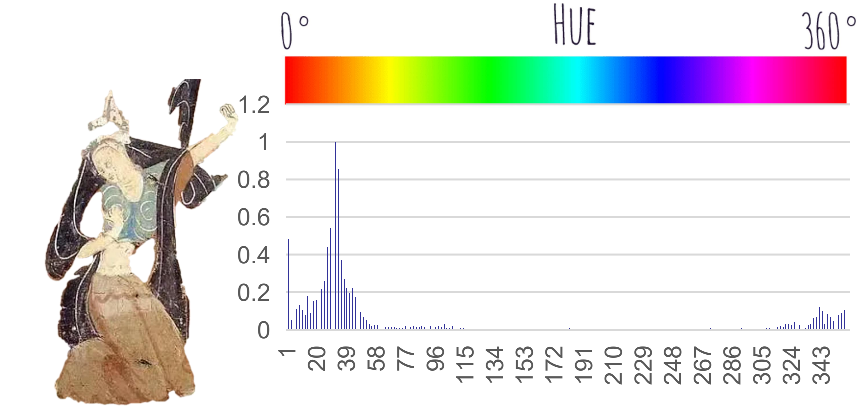


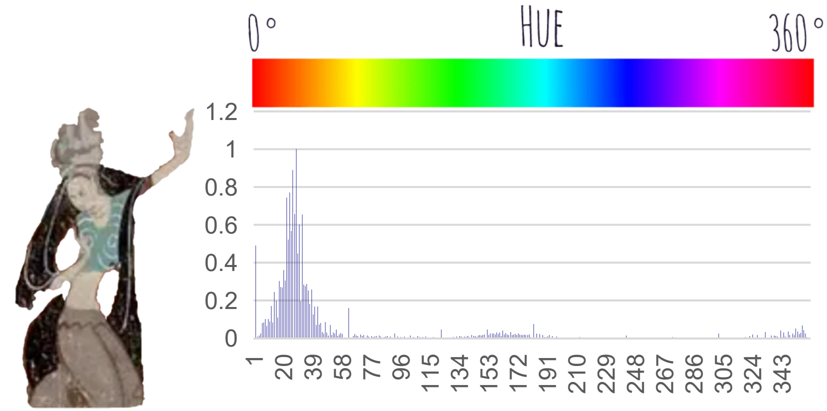


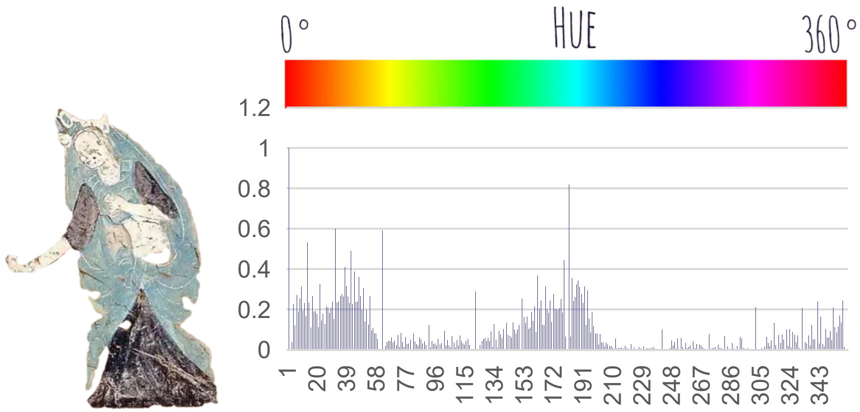


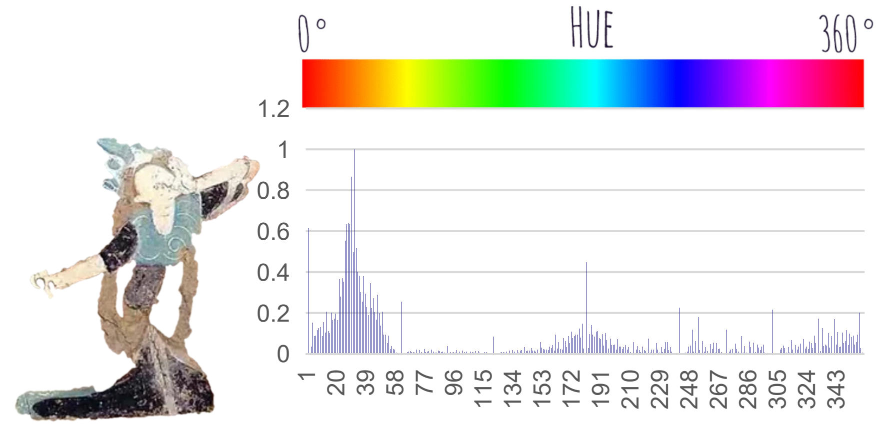


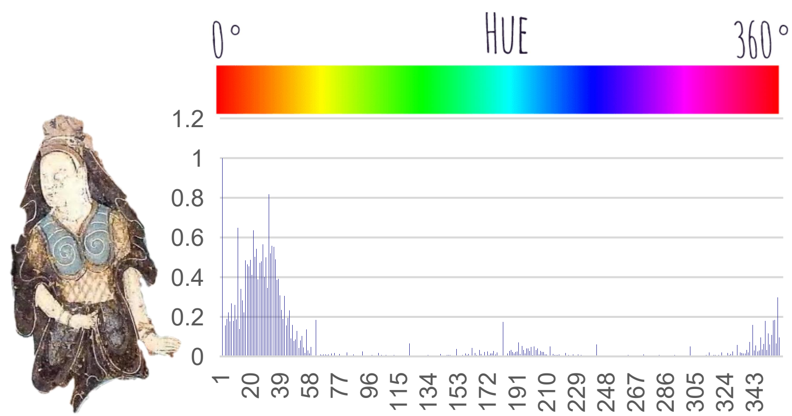
Picture Demo #3: Woman

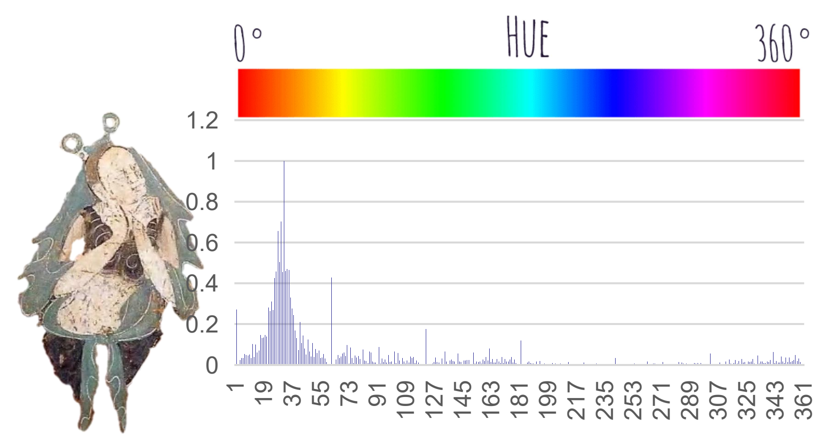












Picture Demo #3: Woman