```
(* Selecting Pixels from an Image Stack that Match Temporal RGB Functions *)
(* Copyright May 10,
 2006    Doug Youvan     www.youvan.com  &  www.pseudocolor.com *)

(* Matrix of target values (columns) as a funtion of time points (rows),
 and for purposes of demonstration, here we use numchannels =
  3  so that the data correspond to RGB data as a function of time,
 with the following (adjustable by user) constraints in
  time: r ~ exponential base 2 decreasing, g ~ constant,
 b ~ exponential base 2 increasing; however if numchannels >3,
 this program accepts hyperspectral data *)

(* Initialize *)

numtimes = 4; numchannels = 3; width = 256; height = 256; numpix = width * height;

(* Target values *)

target = Table[{0.5^(i - 1), 0.5, N[(2^(i - 1)) / numtimes / 2]}, {i, numtimes}];
(* N[] coverts fractions to real numbers *)
MatrixForm[target] (* see first output, below *)


(* Now make an image stack (rank 4 tensor) where each pixel position's r,
 g,b,t values are in the same format as the'target', calculated above;
 and there are width * height number of these 'composite r,g,b,
 t pixels'; this is a rank 2 matrix where each element is rank 2,
 forming a rank 4 tensor named is[x,y] of random numbers; '
  real' data can be inserted in matrix is[x,y] *)

is = Table[{x, y}, {x, height}, {y, width}];
For[x = 1, x ≤ height, x++,
  For[y = 1, y ≤ width, y++,
   is[[x, y]] = Table[{Random[], Random[], Random[]}, {i, numtimes}]
  ]];
Dimensions[is];

(* Calculate SSD differences between data and target, yielding a rank 2 tensor *)

graymat = Table[{x, y}, {x, height}, {y, width}];
graymat[[All, All]] = 0.;
For[x = 1, x ≤ height, x++,
  For[y = 1, y ≤ width, y++,
   graymat[[x, y]] = Total[Flatten[is[[x, y]] - target]^2]
  ]];

graymin = Min[graymat];
graymax = Max[graymat];

(* rescale graymat according to min max to get grayscales of 1 - 256 *)

For[x = 1, x ≤ height, x++,
  For[y = 1, y ≤ width, y++,
   graymat[[x, y]] = Floor[(graymat[[x, y]] - graymin) * (255 / (graymax - graymin))] + 1
```

```
  ]];

Min[graymat];
Max[graymat];

(* Make a pseudocolor 'hotmap' with 256 RGB elements as in Example 2 *)

hotmap = Table[{
    Piecewise[{
      {1,  1 <= i < 42},
      {1, 42 <= i < 84},
      {Floor[5.8 * (i - 83)], 84 <= i < 128},
      {256, 128  <= i < 170},
      {256, 170 <= i < 212},
      {256, 212 <= i <= 256}
     }],
    Piecewise[{
      {1,  1 <= i < 42},
      {(i - 41) * 6, 42 <= i < 84},
      {256, 84 <= i < 128},
      {256 - (i - 128) * 6, 128  <= i < 170},
      {1, 170 <= i < 212},
      {Floor[(i - 211) * 5.7], 212 <= i <= 256}
     }],
    Piecewise[{
      {Floor[i * 5.8],  1 <= i < 42},
      {256 - ((i - 42) * 6), 42 <= i < 84},
      {1, 84 <= i < 128},
      {1, 128  <= i < 170},
      {(i - 169) * 6, 170 <= i < 212},
      {256, 212 <= i <= 256}
     }]
   },
   {i, 1, 256, 1}];

hotmap[[1, All]] = 1 ;
(* This changes 1,1,5 to 1,1,1 at i=1 for the blue channel *)

(* index graymat values into hotmap as done in E2; however,
 this is an improved version for a matrix that is already floored *)

colormat = Table[{x, y}, {x, height}, {y, width}];
colormat[[All, All]] = {0, 0, 0};
For[x = 1,  x ≤ height,  x++,
  For[y = 1,  y ≤ width,  y++,
   colormat[[x, y]] = hotmap[[graymat[[x, y]] ]] / 256
  ]];

Min[colormat];
Max[colormat];

(* Display SSD distances in hotmap scale *)

gmat = Graphics[RasterArray[Apply[RGBColor, colormat, {2}]],
   ImageSize → {2 * width, 2 * height},  AspectRatio → Automatic];
```

```
Show[gmat];


(* recursively clip high SSD values and rescale low
   SSD values via hotmap: Look for dark pseudocolored pixels ->
  minimal SSD for best fit to function *)

For[loop = 1, loop ≤ (numtimes + 2), loop++,

 graymat[[All, All]] = 2 * graymat[[All, All]];

 For[x = 1, x ≤ height, x++,
  For[y = 1, y ≤ width, y++,
   If[graymat[[x, y]] > 256, graymat[[x, y]] = 250]
  ]]; (* 250 on hotmap is a light pink background *)


 colormat = Table[{x, y}, {x, height}, {y, width}];
 colormat[[All, All]] = {0, 0, 0};
 For[x = 1, x ≤ height, x++,
  For[y = 1, y ≤ width, y++,
   colormat[[x, y]] = N[hotmap[[graymat[[x, y]] ]] / 256]
  ]];

 gmat = Graphics[RasterArray[Apply[RGBColor, colormat, {2}]],
   ImageSize → {2 * width, 2 * height}, AspectRatio → Automatic];
Show[gmat];

]; (* end of recursive clip and display loop *)
```

Out[71]//MatrixForm=

$$\begin{pmatrix} 1 & 0.5 & 0.125 \\ 0.5 & 0.5 & 0.25 \\ 0.25 & 0.5 & 0.5 \\ 0.125 & 0.5 & 1. \end{pmatrix}$$