

Image Overlay Using Transparency

By Steve Eddins

When we need to look at two images together—for example, to compare the input and output of a particular image processing operation, or to compare different images of the same scene—overlying one image on top of the other is often more effective than viewing the images side-by-side. Each pixel in a Handle Graphics® image object can be assigned a different level of transparency using the 'AlphaData' property of the image¹.

This article focuses on two transparency patterns that can be applied to create a wide variety of image visualizations: a checkerboard that shows portions of two images in an alternating pattern of squares, and a data-dependent pattern that uses varying shades of a single color.

Using a Checkerboard Transparency Pattern

We will use this technique to compare the input and the output of a grayscale conversion.

First, we display the input and output images together (Figure 1).

```
rgb = imread('peppers.png');
imshow(rgb);
I = rgb2gray(rgb);
hold on
% Save the handle for later use
h = imshow(I);
hold off
```

Not too surprisingly, only the grayscale image is visible. We will give the grayscale image a checkerboard transparency pattern so that some pixels are opaque and others are transparent (Figure 2).

```
[M,N] = size(I);
block_size = 50;
P = ceil(M / block_size);
Q = ceil(N / block_size);
alpha = checkerboard(block_size, ...
    P, Q) > 0;
alpha = alpha(1:M, 1:N);
set(h, 'AlphaData', alpha);
```



FIGURE 1. Grayscale output image overlaid on color input image.



FIGURE 2. Checkerboard transparency pattern applied to the grayscale image, revealing sections of the underlying color image.

¹ The 'AlphaData' is a $M \times N$ matrix of the same size as the image with each element in the range $[0 \ 1]$ indicating the "opacity" of a pixel.



FIGURE 3. DEM of Peppercorn Hill and North Pond, Massachusetts. Bright pixels correspond to high spots in the terrain. (Original data courtesy of the U. S. Geological Survey)



FIGURE 4. Influence map showing downhill water flow.



FIGURE 5. Solid green image overlaying the DEM image.

Now we can see portions of both images and can easily compare the grayscale input and output.

Using Image Data to Control Transparency

Our second transparency example gets a bit more creative. This technique “colors” some pixels in one image according to the values of pixels in a second image. Our example uses this method to visualize downhill water flow on top of a digital elevation model (DEM) (Figure 3).

```
E = imread('peppercorn_hill.png');
imshow(E, 'InitialMag', 'fit')
```

First, we create an influence map, a visualization of downhill water flow, starting from the peak of Peppercorn Hill (Figure 4).

```
I = imread('influence_map.png');
imshow(I, 'InitialMag', 'fit')
```

It is difficult to interpret the influence map image outside the context of the original DEM—for example, it is hard to see exactly where the downhill flow reaches North

Pond. To visualize the two images together, we do the following:

1. Display the original DEM image.
2. Lay a solid green image over the DEM image (Figure 5).

```
imshow(E, 'InitialMag', 'fit')
% Make a truecolor all-green image.
green = cat(3, zeros(size(E)), ...
           ones(size(E)), zeros(size(E)));
hold on
h = imshow(green);
hold off
```

3. Use the influence map pixels to control the transparency of each pixel in the green image (Figure 6).

```
% Use our influence map as the
% AlphaData for the solid green image.
set(h, 'AlphaData', I)
```

We can now easily see that the water flows from the peak into the pond and then out at the pond’s southern end. ■

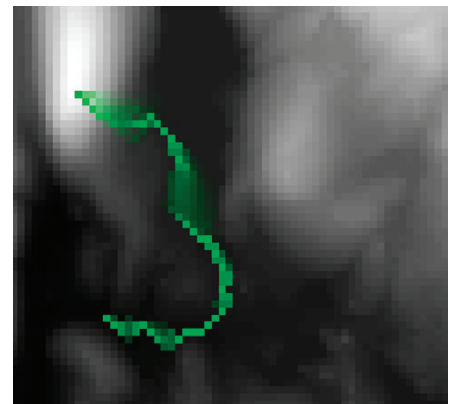


FIGURE 6. Water flow image overlaid on the DEM.

Resources

TRANSPARENCY IN MATLAB

www.mathworks.com/nn9/transparency

IMAGE PROCESSING PRODUCTS

www.mathworks.com/nn9/imageprocessing