C.2.2 Independent section

1. The `image` function that we have been using can be defined as:

$$\textit{MatlabImage}: \textit{ColormapImages} \times \textit{Colormaps} \rightarrow \textit{Images}$$

where

$$\textit{ColormapImages} = [\{1, \cdots, 200\} \times \{1, \cdots, 200\} \rightarrow \{1, \cdots, 256\}]$$

and

$$\textit{Colormaps} = [\{1, \cdots, 256\} \rightarrow [0, 1]^3]$$

and

$$\textit{Images} = [\textit{DiscreteHorizontalSpace} \times \textit{DiscreteVerticalSpace} \rightarrow \textit{Intensity}^3].$$

2. The following program creates a movie:

```
k = 0:199;
numFrames = 15;
m = moviein(numFrames);
for frame = 1:numFrames
   x = sin(k*2*pi/200 + pi/2 + (frame - 1)*2*pi/numFrames) + 1;
   b = repmat(x', 1, 200) * 128;
   image(b), axis image
   m(:,frame) = getframe;
end
movie(m)
```

Again, an inelegant solution using for loops looks like this:

```
numFrames = 15;
m = moviein(numFrames);
for frame = 1:numFrames;
   for row = 1:200
      for col = 1:200
         sinImage(row,col) = 128*(1 + cos(2*pi*row/200 ...
         + (frame-1)*2*pi/numFrames));
      end
   end
   image(sinImage)
   axis image
   m(:,frame) = getframe;
end
movie(m)
```

In this case, the for loops are dramatically slower. You should avoid them when possible. The command

```
movie(m, 100, 30)
```

repeats the movie 100 times at 30 frames per second (if the computer is fast enough).

3. To get the colormap for the red separation, we do

```
>> mask = [ones(256,1), zeros(256,2)];
>> redcolormap = map.*mask;
>> colormap(redcolormap)
```

which results in the image shown at the left in figure C.5. The green color separation is generated by

```
green = helen(:,:,2);
image(green), axis image
mask = [zeros(256,1), ones(256,1), zeros(256,1)];
greencolormap = map.*mask;
colormap(greencolormap)
```

which results in the image shown in the center in figure C.5. The blue color separation is generated by

```
blue = helen(:,:,3);
image(blue), axis image
mask = [zeros(256,2), ones(256,1)];
bluecolormap = map.*mask;
colormap(bluecolormap)
```

which results in the image shown at the right in figure C.5.

4. The blurred image of figure C.6 is constructed as follows:

```
blurred = zeros(300,200);
for row = 3:298
   for col = 3:198
      avg = 0;
      for i = -2:2
         for j = -2:2
            avg = avg + bwImage(row-i, col-j);
         end
      end
      blurred(row,col) = avg/25;
   end
end
image(blurred), axis image
```

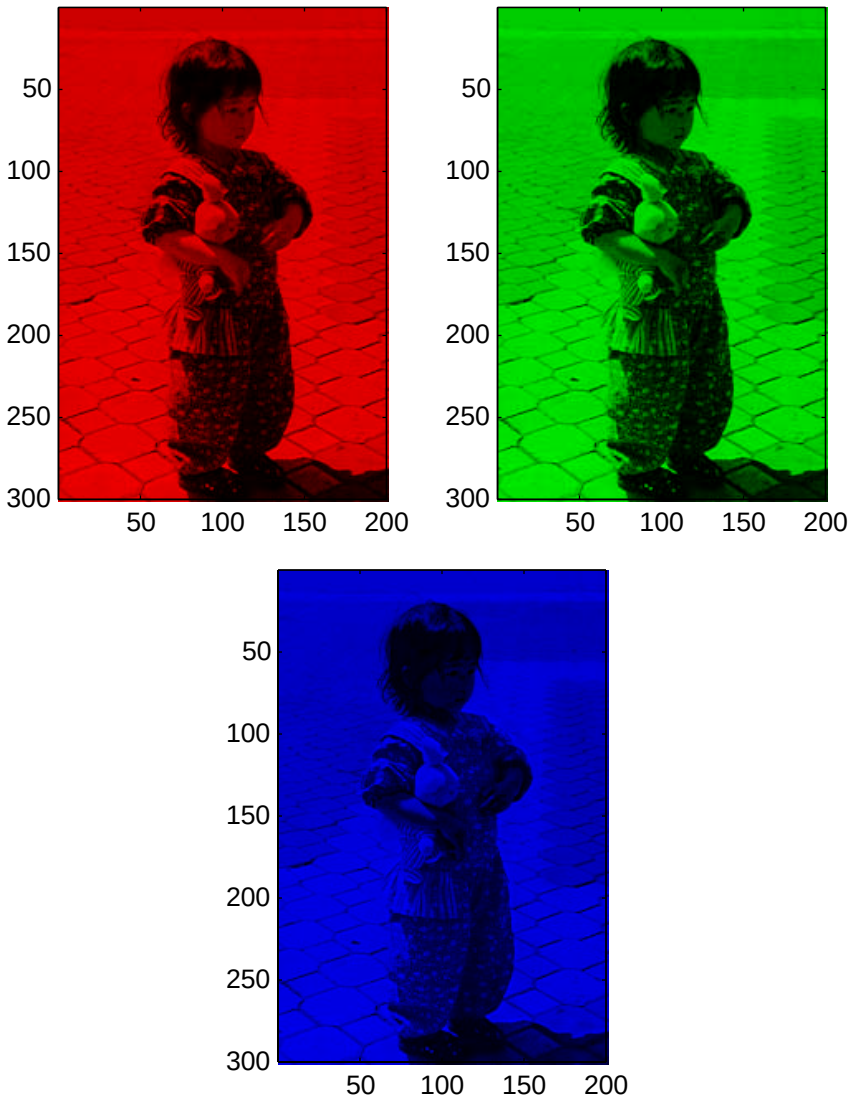

Figure C.5: Color separations in red, green, and blue.

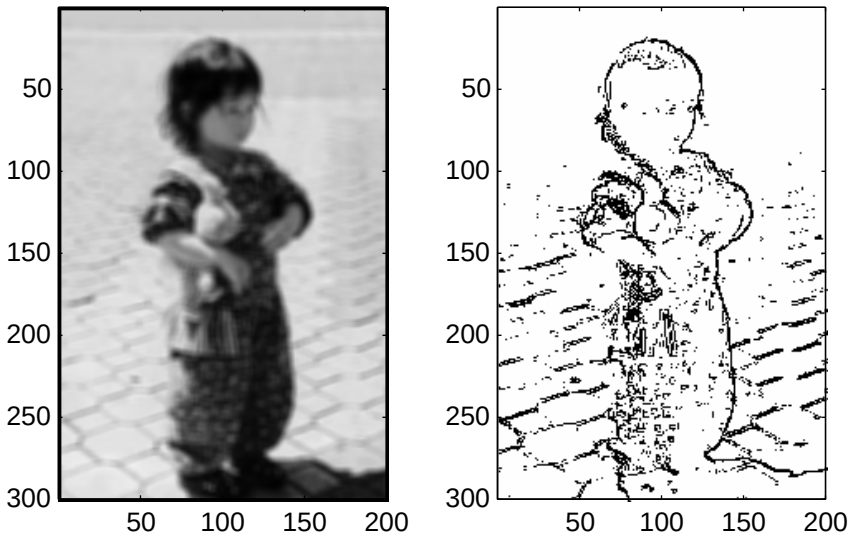

Figure C.6: Blurred image constructed with a 5×5 moving average (left), and simple edge detection (right).

5. The following program results in the plot shown on the right in figure C.6.

```
edges = ones(size(bwImage))*255;
threshold = 40;
for row = 2:300
   for col = 2:200
      vertDiff = bwImage(row, col) - bwImage(row - 1, col);
      horDiff = bwImage(row, col) - bwImage(row, col - 1);
      if ((abs(vertDiff) > threshold) | (abs(horDiff) > threshold))
         edges(row,col) = 0;
      end
   end
end
image(edges), axis image
```

The threshold of 40 yields a reasonable compromise between detail and clutter.