Sobel Filtering:

Sobel filtering is a *three step* process. Two 3×3 filters (often called *kernels*) are applied separately and independently. The weights these kernels apply to pixels in the 3×3 region are depicted below:

-1	0	+1
-2	0	+2
-1	0	+1

+1	+2	+1
0	0	0
-1	-2	-1

Again, notice that in both cases, the sum of the weights is 0. The idea behind these two filters is to approximate the derivatives in x and y, respectively. Call the results of these two filters $D_x(x, y)$ and $D_y(x, y)$. Both D_x and D_y can have positive or negative values, so you need to add 0.5 so that a value of 0 corresponds to a middle gray in order to avoid clamping (to [0..1]) of these intermediate results.

The final step in the Sobel filter approximates the gradient magnitude based on the partial derivatives ($D_x(x, y)$ and $D_y(x, y)$) from the previous steps. The gradient magnitude, which is the result of the Sobel Filter $S(x, y)$, is simply:

$$S(x, y) = \sqrt{(D_x(x, y))^2 + (D_y(x, y))^2} \tag{1}$$

Please note that your textures should not store $D_x(x, y)$, but should rather store $D_x(x, y)+0.5$, as I mentioned above... This means before computing the value in Equation 1, you need to first subtract the 0.5 you added when computing D_x and D_y!

So, in summary, the three steps are:

- Compute the image storing partial derivatives in x ($D_x(x, y)$) by applying the right 3×3 kernel to the *original* input image.
- Compute the image storing partial derivatives in y ($D_y(x, y)$) by applying the left 3×3 kernel to the *original* input image.
- Compute the gradient magnitude $S(x, y)$ based on D_x and D_y.

Two further things to notice about Sobel filters: (a) both the derivative kernels depicted above are seperable, so they could be split into disjoint x and y passes, and (b) the entire filter can actually be implemented in a single-pass GLSL filter in a relatively straightforward manner.