**2.1.1**

1) Refer to source code.

2) The code ran 2.18x faster. The serial algorithm took 2.349s while the SIMD implemented calculation took 1.077s.

3)   
\_mm\_loadu\_ps – This is used to load unaligned single precision floating-point data from memory using the frame pointer into the SIMD registers.  
\_mm\_hadd\_ps – This is used to do a horizontal add between two SIMD registers. This enables 4 additions simultaneously.   
\_mm\_store\_ps – This is used to store the calculated data after addition in the SIMD registers back into memory.

4) The code first initiates the vectorized loop where it calculates the sum of a 4x4 block starting from the top left corner of the blur block. It will try to fit this much 4x4 blocks into the loop to do the calculations. The fringe rows , or the remainder rows that are not divisible by 4, are then added using lesser vectors. The amount of vectors used is determined by the remainder. The fringe columns, are then calculated serially. All these values are added up into the average divided by the size of the blur block.

5) Aligned/unaligned loads were considered. Because the frame values in memory might not be aligned, I used the load unaligned intrinsic instruction to load the values properly. Load unaligned decreases performance. To get better performance, I would not use this intrinsic instruction but instead grab the aligned values in the middle and deal with the unaligned values serially.

**2.2.1**