Loop unrolling:

Int x[100]

for(i=0;1<100;i++)

X[i] = 0

#loops = 100

#instr per loop: 5

Unrolled:

Int x[100]

for(i=0;1<100;i+=2)

X[i] = 0

X[i+1] = 0

#loops = 50

#instr per loop: 8

Further:

Int x[100]

for(i=0;1<100;i+=4)

X[i] = 0

X[i+1] = 0

X[i+2] = 0

X[i+3] = 0

#loops = 25

#instr per loop: 14

Number of total instructions: 500, 400, 350

-find the most appropriate loop step for best optimization, there is an upper bound.Unrolling too many times can lead to excessive register use.

Loop Blocking:

Similar to loop unrolling, trying to get better cache performance.

for(int i =0; i<n i++)

for(int j=0; j<n; j++)

Two matrices, a and b

Take entire row of a and multiply by col of b

Inner-most loop, k is the only one that is changing.

Use spatial locality to advantage

Smooth function strategy:

Smooth function takes one element and takes neighboring pixels and set to the mean color value.

Loop through and call smoothing function, add changes to matrix - lots of function overhead setting input values each time. \*Maybe incorporate function code instead of calling func every time.