Image Segmentation Design

EdgeKey allows the use of ints r,c,I as a map key. The class should be able to return From and To corrdinates.

Rows and cols need to be global or shared somehow. It’s all operating on one image, the row and col shouldn’t be all over the place.

Class Edge

Int r,c,I;  
 int weight

Int id(){ c + r\*rows + i \* rows \* cols; }

bool operator <(const EdgeKey & e) const {

return id() < e.id()

}

Make Edge class. It stores r,c,I, and weight. Overload operator< and base it on weight alone. (Or make a compare function)

bool operator <(const Edge& e) const {

return weight < e.weight;

}

struct EdgeWeightComparator

{

bool operator() (const Edge &a, const Edge &b)

{

return a.weight < b.weight;

}

};

Probably won’t use the ID as the edges will be placed in a list and sorted by weight.

Use STL Pair to store row/col, begin/end values.

Component class

Component  
 int ID;  
 Pair Base; //this is the smallest row and col of the component  
 Pair Box; // the size of the component  
 int Size; // Size of component (number of Vertex)  
 Map<int,Pair> // int is local row number, Pair is col begin/end  
 bool IsInside(Pair) // returns true if Vertex (Pair) is inside.

Component Storage and management.

Map<int, (shared ptr)component> CompMap; // This maps ID to component

Map<Pair, (shared ptr)component> VertextCompMap// This maps every vertex to the component it is in. Since Comp has an ID it can be used to find if two vertex are in same component.

With the above construction CompMap may not be needed. Just init VertextCompMap with Comps that have unique IDs. The combine operation replaces the old comp with the new comp in the set. The old should be deleted automatically.

Start with an algorithm that operates on a grey scale image (this will apply to case where it is run three times, one on each color plane.)

Global parameters:

Rows, cols, K

Put these in a structure and pass a naked pointer to the structure to each class instance that needs this information. Only way I can figure to avoid global.

Open image as int32 and map to Eigen matrix.

Create the Edge vector and pre-allocate its size.

Compute the Edge vector. I guess this means all the Edge instances are already created.

Sort the Edge vector.