Code Description - Probabilistic Image Segmentation

Region.h

- *struct Coord*: Defines a two dimensional vector to save the pixel coordinates of the region
- *calculateGLColorMeanValue*: Calculates the mean intensities of the L*a*b channels of the region.
- *calculateGLTexture*: calculates the standard deviation of the region to describe its texture. Achieved by building a histogram to obtain a probability function and by computing the regions mean intensity value. The for loop "calculate variance" is used to calculate the following formula:
  $\sigma = \sqrt{\sum_{n=1}^{L}(i-\mu)^2 p(i)}$
  μ: intensity mean value of the region,
  $i = 1...L$: greylevels of the image
  $p(i)$: probability of greylevel i.
- *addPixel(Coord p)*: This function adds a pixel into the pixel list of the region.
- *addBoundaryPixel(Coord p)*: This function adds a pixel into the boundary list of the region.
- *findBoundaryPixel(Coord p)*: Returns true if the pixel is in the *Coord p* boundary list.
- *Coord getAt(int pos)*: Gets to a certain position in the region list and returns the image coordinates.
- *getNoOfPixels()*: Returns the amount of pixels in the region.
- *getNoOfBoundaryPixels()*: Returns the amount of the regions boundary pixels.
- *Coord getBoundaryPixel(int i)*: Returns pixel coordinates from the image of the ith entry in the boundary list.
- *void adjustBoundaryPixels*: After merging two regions, the algorithm runs over the merged boundary lists and checks whether a certain pixel should be in the boundary list or not.

RegionGrowing.h

- *float calculateArrangement(int reg1,int reg2)*: Input are two regions reg1 and reg2. For loop takes a value of the boundary list reg1 and checks if one of its 4-neighborhood pixels is present in the boundary list reg2. If yes it increases count by 1. The final arrangement value gets calculate dividing count by the sum of the boundary lists of reg1 and reg2.

- *void displayContours(char *name)*: Display contours of all regions by changing the color of boundary pixels to red.

- *void displayBoundaryOfaRegion(int i,char *name)*: Display contours of a specific region by changing the color of the boundary pixels of that region to red.

- *void Merge(int r1,int r2)*: Input are two regions r1 and r2. Adds the pixels of r2 to the region list of r1. Also adds boundary list r2 to boundary list r1 and adjusts the resulting boundary list of r1.

- *int Probability(int reg)*:
  - Computes merging probability of adjacent regions.
  - Calculates the posterior probabilities with Bayes theorem $\hat{P}(merge|x_{1..n}) = \frac{\hat{p}(x_{1....n}|merge)\hat{P}(merge)}{\hat{p}(x_{1....n}|merge)\hat{P}(merge)+\hat{p}(x_{1....n}|\overline{merge})\cdot\hat{P}(\overline{merge})}$
  - The merging probability threshold is defined in this function

- *vector< float > PriorProbs()*: Computes the prior probabilities P(merge) and P(mergeNOT) from training data by divide number of merged/not merged events over total length of training data vector.

- *float likelihoods* Computes the approximated likelihood of a feature. Uses previous calculated parameters, cluster centers, B-terms, a-terms and the bandwidth. $\hat{p}(x_e)^{(r)} = \sum_{\forall\ s.t\ |x_e-c_l|\leq r_{ex}} \sum_{k=0}^{p-1} \sum_{s=0}^{\lfloor r/2 \rfloor} \sum_{t=0}^{r-2s} a_{st} B_{kt}^{l} e^{\frac{-|x_e-c_l|^2}{2h^2}} \cdot \left(\frac{x_e-c_l}{h_1}\right)^{k+r-2s-t}$

- *double factorial(int n)*
  - Computes the faculty n! of the input n.
  - Required to calculate the a-terms

- *vector< float > choose_parameters(double h, int N)*
  - Computes the parameter for the Fast Gaussian KDE approximation.
  - number of intervals K
  - interval length rx

  - cut off radius rr
  - cluster cutoff radius ry
  - truncation number p
  - factor q

- *calculateLikelihoodDataVectors()*
  - Calculates for each feature the data vector for the likelihood estimation.
  - Accesses the training data vector.

- *void CalcVectorMaxandMin(int col, vector< Stat > featureVector)*
  - Calculates the min and max value of an input vector
  - Required to normalize the feature data vectors

- *void normalizeTrainVector(vector< Stat > input)* Normalizes the input vector by formula:
  $\frac{x-min}{max-min}$

- *void calculateParameters()* Calls the *choose_parameters(double h, int N)* function to obtain parameters for each feature.

- *void calculateLikelihoodCenters()* Calls the CalcClusterCenter function to obtain the cluster centers for each feature.

- *void calculateLikelihoodIndices()* Calls the function CalcClusterIndex to obtain corresponding cluster centers for each feature.

- *void CalcClusterIndex(double rx, int K, vector< float > inputVector, vector< int > pClusterIndex)* Computes the corresponding cluster center for each entry of the inputVector

- *void calculateLikelihoodBterms()* Calls the function *compute_Bterms* to obtain the the B-terms for the normalized data vectors of each feature

- *void compute_Bterms(int K, int p, int r, double h, int N, double q, vector< int > pClusterIndex, vector< float > pClusterCenter, vector < float > inputVec, vector< float > BtermVec)*
  Computes the B-terms by applying the formula:
  $B^l_{kt} = \frac{2^k}{k!} \sum_{x_i \epsilon S_l} q_i e^{\frac{-|x_i - x_l|^2}{h_2^2}} \left(\frac{x_i - c_l}{h_2}\right)^k \left(\frac{x_i - c_l}{h_1}\right)^t \; k = 0...p-1, t = 0....r$

- *void calculateLikelihood_a_terms()* Calls the function *compute_a* to obtain the the a-terms for the normalized data vectors of each feature.

- *void compute_a(int r, vector< float > a_termsOut)* Computes the a-terms by applying the formula:
  $a_{st} = \frac{(-1)^{s+t} \cdot r!}{2^s s! t! (r-2s-t)!}$

- *void RunSLIC(char* imagePath,int noOfSuperpixels,int nc)*: Starts to execute the SLIC algorithm.

- *void FindAdjacentRegions(string adjacentRegionsFile)*: This function iterates through all of the regions and finds and stores all adjacent regions for every region.

- *void ReadAdjacentRegionsFromFile(string adjacentRegionsFile)*: This function is used in place of FindAdjacentRegions(). If some image has been processed earlier, then data about its adjacent regions would have been saved in a file. If this image is processed again, then instead of finding the adjacent regions again, the data is simply loaded from a file in order to save computation time.

- *void Perform()*: This function iterates through all of the regions and calculates the probability of merging of each region with its adjacent regions and merges with the highest probability neighbor. This is repeated until all probabilities are < 0.5.

- *void SaveResult(const char *resultImagePath)*: Stores the segmentation results in a folder so that they can be viewed afterwards.

- *void SaveBoundaryMapImage(const char *boundaryMapImagePath)*: Stores a boundary map of the final segmented image that can be used to evaluate the results.

- *void ClearData()*: Clears all the data (list/vectors) that was used for processing one image so that no useless data is used by the processing of further images.

- *void Start*: This function is the main public function of the class and is called from the driver program. This function executes all algorithms step by step e.g. first it runs SLIC, then finds adjacent regions, then gets training data, then performs the segmentation and finally saves the boundary map.

Training.h

- *float calculateArrangement(int reg1,int reg2)*: Input are two regions reg1 and reg2. For loop takes a value of the boundary list reg1 and checks if one of its 4-neighborhood pixels is present in the boundary list reg2. If yes it increases count by 1. The final arrangement value gets calculate dividing count by the sum of the boundary lists of reg1 and reg2.

- *bool Check(Region *reg1,Region *reg2)*: This function takes as input two regions and checks whether they are falling into the same segment of the region based on the training data, which was previously loaded.

- *void Merge(int r1,int r2)*: Input are two region labels r1 and r2. Adds the pixels of r2 to the region list of r1. Also adds boundary list r2 to boundary list r1 and adjusts the resulting boundary list of r1

- *void displayContours(char *name)*: Display contours of all regions by changing the color of boundary pixels to red.

- *void LoadTrainingData(string infile)*: This function loads the data from the file obtained from Barkeleys data set and stores it in a vector.

- *void RunSLIC(char* imagePath,int noOfSuperpixels,int nc)*: Starts to execute the SLIC algorithm.

- *void FindAdjacentRegions(string adjacentRegionsFile)*: This function iterates through all of the regions and finds and stores all adjacent regions for every region.

- *void ReadAdjacentRegionsFromFile(string adjacentRegionsFile)*: This function is used in place of FindAdjacentRegions(). If some image has been processed earlier, then data about its adjacent regions would have been saved in a file. If this image is processed again, then instead of finding the adjacent regions again, the data is simply loaded from a file in order to save computation time.

- *void Perform(string outfile)* : The function checks every pair of adjacent regions (by calling function Check()) whether they fall into the same original region or not and on this basis stores YES or NO alongwith other parameters (color difference, texture difference and arrangement of the two regions).

- *void SaveResult(const char *resultImagePath)*: Stores the segmentation results in a folder so that they can be viewed afterwards.

- *void ClearData()*: Clears all the data (list/vectors) that was used for processing one image so that no useless data is used by the processing of further images.

- *void Start(char* imagePath,int n,string outfile)*: This is the main driver function of this class, which executes the training. It runs the SLIC algorithm so as to get some segmentation of the image.

- *vector<Stat> GetTrainingData(string filename)*: This function is formed so as other classes can import the training results to perform the decision making for the segmentation.

Timer.h

This code file is used to calculate the running time of an Algorithm.

- *void PrintTime()*: Prints the time in proper format i.e. in appropriate floating point value with proper unit of time.
- *void Start()* : Starts the stopwatch.
- *void End()* : Ends the stopwatch.
- *void LocalEnd()*: Ends the stopwatch and prints the time in some specific format depending upon from where the function was called.
- *void TotalEnd()*: Ends the stopwatch and prints the time in some specific format depending upon from where the function was called

Structs.h

This file contains different structs that we have used at various points in our code. The purpose of this file is that there are some structs that are used in more than one place. So, instead of declaring them again and again, we just placed them in a file and included that file into the file where it should be used.

slic.cpp and slic.h

We downloaded the SLIC code from here: https://github.com/PSMM/SLIC-Superpixels. We added a function getResults(IplImage *image,vector <Region> &regions,vec2di &labels) to get the output results of the SLIC algorithm.

GroundTruth.cpp and GroundTruth.h

- *GroundTruth()*: Default constructor for a GroundTruth object.
- *GroundTruth(int rows, int cols)*: Constructor for a GroundTruth object.
  Input: Number of rows and columns of the image that has to be segmented.
  Constructor initializes the ground truth matrix with the given dimensions and -1 in each entry. Each entry represents the segment of the pixel it represents.
- *GroundTruth(std::vector¡Region¿ *regions, int rows, int cols)*: Constructor for a GroundTruth object.
  Input: Vector of regions, number of rows and number of columns of the image that has to be segmented. Does the same as the above constructor but in addition it assigns each entry of the matrix the segment value of the corresponding pixel.

- *int getSegmentOfPixel(int row, int col)*
  Input: row and column index of a pixel.
  Output: the segment value of the input pixel.
- *void setSegmentOfPixel(int value, int row, int col)*
  Input: Segment value, row and column index of a pixel. The procedure assigns the entry for the given coordinates the given value.
- *void setNumberOfSegments(int number)*
  Input: Number of segments. Sets the number of segment of a GroundTruth object.
- *void importData(char* fileName)*
  Input: Segmentation file (.seg).
  Reads a .seg-file and sets up a GroundTruth matrix and the number of segments.
- *void showGroundTruth()* Converts the GroundTruth matrix into an openCV matrix.
- *void printGroundTruth()* Writes the GroundTruth matrix in the terminal.