

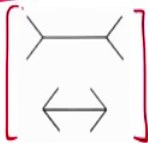
**Deep Learning for Computer Vision**  
**Prof. Vineeth N Balasubramanian**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Hyderabad**  
**Lecture 12**  
**Image Segmentation**

Last lecture, we spoke about feature extractors and feature distractors. In particular, we spoke about shift and it's variants. Before we go to other kinds of features, we will take a slight detour, and talk about another important task in processing of images, which is image segmentation.

(Refer Slide Time: 0:46)

**Human Vision: Gestalt and Grouping**



- Gestalt theory emerged in the early 20th century with the following main belief: "The whole is greater than the sum of its parts"
- Gestalt theory emphasized grouping as an important part of understanding human vision.



- The famous Muller-Lyer illusion above illustrates the Gestalt belief - humans tend to see things as groups and not individual components.

Source: David Forsyth

Vineeth N B. (IIT-H) 12.5 Image Segmentation



Human vision has been studied extensively from several perspectives, and one of the perspectives of human perception has been based on what is known as the Gestalt theory, which emerged almost 100 years ago, which had this fundamental principle that the whole is greater than the sum of its parts.

And one of the important cornerstones of this theory was the idea of grouping, to understand human vision. So this is a very popular optical illusion. It is known as the Muller-Lyer illusion which asks a viewer as to which of these horizontal lines is longer. And because of the human perceptions bias towards viewing things as a group and not the individual components, the illusion actually arises here and makes one line look longer than the other while the horizontal lines are exactly the same length in both of these images.

(Refer Slide Time: 2:00)

Human Vision: Gestalt and Grouping

Similarity

Common Fate

Continuity

Common Region

Closure

Credit: David Forsyth

Vineth N B. (IIT-H)

3.5 Image Segmentation

So Gestalt theory proposes various factors and images, which can result in grouping. So on one hand you could have proximity to be a reason, when there are objects or artifacts in the images, which are simply close to each other and these groups lie in different parts of the image, that could result in one kind of a grouping or it could just be the similarity of the objects themselves.

Some objects are hollow circles, some objects are filled circles, and just because these objects are different they get grouped differently, when we perceive this image. Other factors could be parallelism, and there are groups of lines that have similar properties in terms of parallelism we end up grouping them or another option could also be just symmetry.

Just seeing artifacts in the symmetric sense makes us group them in one particular manner. Further, you could also have similarity of a different kind, you could have a common fate in terms of these arrows telling you where these dots are going. So in the second example here, all these dots are more or less lying in a straight line, but the arrows let us group them differently. So in a sense of where they are going to go along the direction of the arrows, gives the human visual system a queue to how they should be grouped.

Or it could be purely, because of a common region that was already marked as a circle, it could be due to continuity of artifacts, it could be due to closure of artifacts, as you can see, there are many factors that result in human perception viewing things as groups.

(Refer Slide Time: 3:57)

**Human Vision: Gestalt and Grouping**

- Gestalt theory is fairly descriptive - loose set of rules to explain why some elements can be grouped together in an image
- However, rules are insufficiently defined to be directly used to form algorithmic tools for grouping objects in images

**Further Reading**

- Chapter 15.1, Forsyth, *Computer Vision: A Modern Approach*

Vineth N B. (IIT-H) 3.5 Image Segmentation

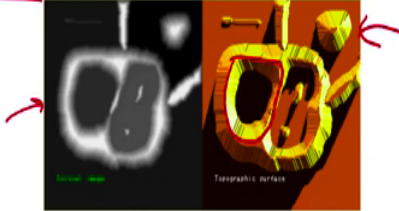
And Gestalt theory is fairly descriptive, it lays down various different rules and principles that could lead to grouping when humans perceive images. However, these rules by themselves are not clearly defined enough to become algorithms, so you really, cannot take the rule directly and implement a pseudo code for that rule that could help you find say groups in an image.

So for the rest of this lecture, we are going to talk about that task called image segmentation, where we try to group similar pixels into different groups. You could call this akin to clustering, where you group different pixels into their own clusters. So a significant part of this lecture is taken from David Forsyth's book and Szeliski's books and we will share these references at the end.

(Refer Slide Time: 4:53)

**Watershed Segmentation Method**

- An early method for image segmentation (1979)
- Segments an image into several "catchment basins" or "regions"
- Any grayscale image can be interpreted as a 3D topographical surface



◦ Image can be segmented into regions where rainwater would flow into the same lake

Credit: S Beucher  
Vineth N B. (IIT-H) 3.5 Image Segmentation

One of the oldest methods for image segmentation is known as watershed segmentation. This was developed way back in 1979. An image segmentation, remember is the task of grouping pixels in an image, right, so that is the task that we are looking for. So watershed segmentation method, segments and image into what are known as catchment basins or regions. It goes along with the name that is why it is called watershed segmentation that will be become clearer in the next slide.

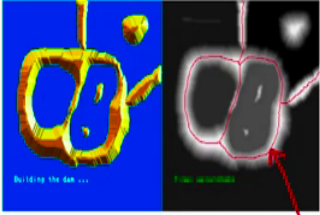
So you can view any grayscale image as a 3D topological surface. So for example, if you have this grayscale image, very similar to how we viewed an image as a function way back in one of the earlier lectures, so you can view an image as a 3D topological surface where the whiteness peaks at these bevelled locations and the whiteness subsides in other locations, and the dark areas are where you have black pixels in the original image.

What watershed segmentation tries to do is to segment region, segment the image into regions with a conceptual idea of rain water flowing into the same lake, so you are going to create a methodology, we are going to assume that you are going to flood water in your image, and wherever you have catchment basins in the image, imagine this 3D topographic representation of the original image. Wherever you have water stagnating, those groups of pixels form segments. Let us see this in a bit more detail.

(Refer Slide Time: 6:50)

**Watershed Segmentation Method**


- Flood the landscape from local minima and prevent merging of water from different minima



- Results in partitioning the image into catchment basins and watershed lines

Credit: S Beucher

Vineth N B. (IIT-H) 3.5 Image Segmentation



So you identify various local minima in your image, remember, local minima, as simply values were our pixel locations where your image intensity value is low. Remember again, that image intensity can lie between 0 and 255 or 0 and 1 if you normalize the values in the image.

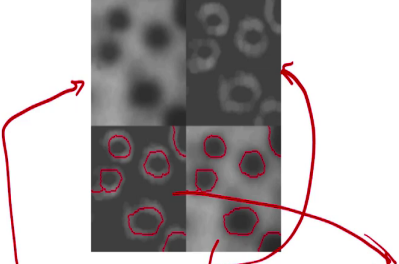
So, you pick in, you take certain neighborhoods say whatever 5X5 patch or 7X7 patch whatever that may be, and using that process, you find out local minima that lie across the image and you flood the landscape from local minima, and prevent merging of water from different minima, which means you would start with a local minima, and slowly keep expanding, which is equivalent to flooding here until you go to a point which is reachable in the same number of pixels from another local minima.

So that is where the boundary between these two regions would lie. So the simple process results is nothing very fancy about this other than what I just mentioned. So this process results in partitioning the image into catchment basins and the watershed lines. Remember, this method was proposed in the late 70s and it was used for many years to segment images into various parts.

(Refer Slide Time: 8:20)

**Watershed Segmentation Method**




Generally applied on image gradients instead of applying directly on images



(Top left) Original image; (Top right) Gradient image; (Bottom left) Watersheds of gradient image; (Bottom right) Final segmentation output

Credit: S Beucher

Vineeth N B (IIT-H) 8:25 Image Segmentation




It was generally applied on image gradients rather than the image itself, just to give you a bit of visual example. So, the original image is this one, we ideally want to group the pixels. So, you first take a gradient image. Now, you know how to take a gradient image run an edge filter on the original image.

So, now the watersheds of your gradient image are given in your bottom left image and you finally refine and get your segmentation output and put it back on the original image the last image. A simple method, but works rather effectively, however, it does have some limitations.

(Refer Slide Time: 9:06)

**Watershed Segmentation Method**




- In practice, often leads to over-segmentation due to noise and irregularities in image
- Hence usually used as part of an interactive system, where user marks "centers" of each component, on which flooding is done



**Further Reading**

- Chapter 5.2.1, Szeliski, *Computer Vision: Algorithms and Applications*

Vineth N B. (IIT-H)      5.2.5 Image Segmentation



If there is a lot of noise and irregularities in the image for example, let us say you have a heavily textured image, so you can imagine a lot of undulations in the texture of the image or if there is noise in the image, you are going to end up with something like this, because they are going to be lot of catchments. So now trying to flood those areas will lead you to many watershed lines that separate many different regions, resulting in what is known as over segmentation.

So keeping this in mind, watershed segmentation is typically used as part of an interactive system, where a user points out a few different centers in the image, and then the algorithm floods from those centers alone, not worrying about other centers that could be in the image, so this idea can help overcome the limitation of over segmentation. If you would like to know more, you can read chapter 5.2.1 Szeliski's book. This is one of the earliest methods.

(Refer Slide Time: 10:17)

**Categories of Methods: Region Splitting and Merging**

- **Region splitting methods** involve splitting the image into successfully finer regions.
  - We'll discuss one such method in the upcoming slides
- **Region merging methods** successively merge pixels into groups based on various heuristics such as color differences
  - Figure on right shows an image segmented into such *superpixels*
  - Generally used as preprocessing step to higher-level segmentation algorithms

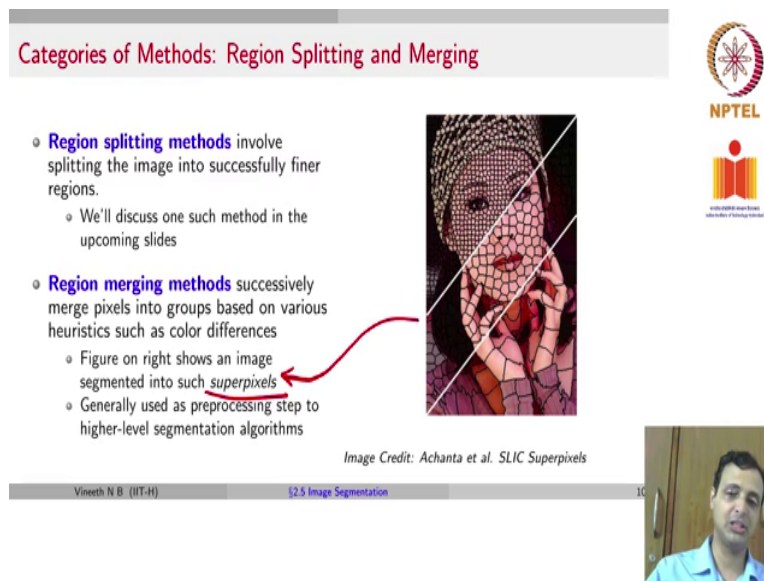


Image Credit: Achanta et al. SLIC Superpixels

Vineth N B. (IIT-H) 3.5 Image Segmentation 10

Since then, there have been many methods that people have developed, and one could broadly categorize methods into two kinds, Region Splitting, and Region Merging. In region splitting methods, as the name suggests, the idea is to start with the image, and then keep splitting it into finer and finer regions.

We will see one example of this method a little later in this lecture. The other kind, which is a bit more popular are what are known as region merging methods, where you start with a pixel as a region and keep merging similar pixels and forming regions and you can keep going forward all the way till the complete image.

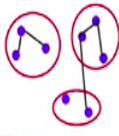
So here is an example of an image, which has been merged into groups of pixels, groups of pixels are also sometimes called superpixels, and this is sometimes generally used as a preprocessing step for higher level segmentation algorithms.



(Refer Slide Time: 11:19)


**Graph-based Segmentation**

- Felzenszwalb and Huttenlocher (2004) proposed a graph-based segmentation algorithm which uses *relative dissimilarities* between regions to decide which ones to merge (region-merging method)
- An image = graph  $G = (V, E)$  where pixels form vertices  $V$  and edges  $E$  lie between adjacent pixels



- A pixel-to-pixel dissimilarity metric  $w(e)$  is defined where edge  $e = (v_1, v_2)$  and  $v_1, v_2$  are two pixels. This measures, for instance, intensity differences between  $N_8$  neighbors.

Vineth N B. (IIT-H) 3.5 Image Segmentation 11



One of the popular methods to do image segmentation was graph-based segmentation, which was proposed in the early 2000s by Felzenszwalb and Huttenlocher, they proposed a graph-based segmentation algorithm, which used dissimilarities between regions to find out which regions to merge in a given iteration. So, if you considered an image as a graph  $G$  with vertices  $V$ , and edges  $E$ , so, you have a set of vertices and a set of edges and the pixels form your vertices initially and edges are defined between adjacent pixels.

So, we define a pixel to pixel dissimilarity metric for example, this could be based on the intensity values in these two pixels. You could also define this using more sophisticated mechanisms by probably taking a neighborhood and taking oriented gradients and so on and so forth, but whatever be that measure you have a pixel to pixel dissimilarity metric, which is defined as  $W$  of  $E$  which is the weight of that edge joining those two pixels on which we are defining the dissimilarity where edges is given by  $v_1, v_2$ , where  $v_1$  and  $v_2$  are two pixels.

For example, this dissimilarity metric could be the intensity differences between the  $N_8$  neighbors of both these pixels. When we say  $N_8$  neighbors, we take a three cross three neighborhood and excluding the central pixel you will have eight different neighbors, we are calling that as  $N_8$  neighbors, so you take two different pixels, take their  $N_8$  neighbors and find their intensity differences that could be one way of computing  $W$  of  $E$ .

(Refer Slide Time: 13:10)

**Graph-based Segmentation**




- For a region  $C$ , its **internal difference** is defined as the largest edge weight in the region's minimum spanning tree:

$$\text{Int}(C) = \max_{e \in \text{MST}(C)} w(e)$$

- The **minimum internal difference** between two adjacent regions is defined as  $\tau(C)$  is a manually chosen region penalty:

$$\text{MInt}(C_1, C_2) = \min(\text{Int}(C_1) + \tau(C_1), \text{Int}(C_2) + \tau(C_2))$$

Vineth N B. (IIT-H) 3.5 Image Segmentation



Once,  $W$  of  $E$  is defined the method defines a quantity called internal difference for every region  $C$ . Initially, remember, the region  $C$  would just be a pixel, but in future iterations a region could be a collection of pixels.

The internal difference is defined as the largest edge weight in the regions minimum spanning tree, which means if you have a bunch of different pixels you find the minimal spanning tree of the bunch of pixels that you have. Now you take the largest edge weight in the minimum spanning tree. We are going to define that as internal difference. To some extent, that is going to be the largest edge weight in that region, it is just a simplified way of saying that, that is one quantity we are going to define.

(Refer Slide Time: 14:06)

**Graph-based Segmentation**




- For a region  $C$ , its **internal difference** is defined as the largest edge weight in the region's minimum spanning tree:

$$\text{Int}(C) = \max_{e \in \text{MST}(C)} w(e)$$

- The **minimum internal difference** between two adjacent regions is defined as  $\tau(C)$  is a manually chosen region penalty:

$$\text{MInt}(C_1, C_2) = \min(\text{Int}(C_1) + \tau(C_1), \text{Int}(C_2) + \tau(C_2))$$

Vineth N B. (IIT-H) 3.5 Image Segmentation 12



Another quantity we are going to define is called minimum internal difference. Minimum internal difference is given by, if you had two regions  $C_1$  and  $C_2$ , the minimum internal difference between these two regions is given by minimum of the internal difference of  $C_1$  plus  $\tau(C_1)$ , where  $\tau(C_1)$  is some manually chosen penalty.

So, I will explain that in a moment, and then the second quantity is the interval difference of  $C_2$  and  $\tau(C_2)$ . So,  $\tau(C_1)$  and  $\tau(C_2)$  could just be say the number of pixels that you have in the region may be normalized based on that or so on and so forth, and so you have a quantity and you take the minimum of both these quantities.



You have region one for whom you define, its internal difference plus some penalty region  $C_2$ , whose internal difference and its corresponding penalty, and you take the minimum of these two.


(Refer Slide Time: 15:08)

### Graph-based Segmentation

- For a region  $C$ , its **internal difference** is defined as the largest edge weight in the region's minimum spanning tree:
 
$$\text{Int}(C) = \max_{e \in \text{MST}(C)} w(e)$$
- The **minimum internal difference** between two adjacent regions is defined as  $\tau(C)$  is a manually chosen region penalty):
 
$$\text{MInt}(C_1, C_2) = \min(\text{Int}(C_1) + \tau(C_1), \text{Int}(C_2) + \tau(C_2))$$
- For any two adjacent regions with at least one edge connecting their vertices, difference between these two regions = minimum weight edge connecting these two regions

$$\text{Diff}(C_1, C_2) = \min_{e=(v_1, v_2), v_1 \in C_1, v_2 \in C_2} w(e)$$

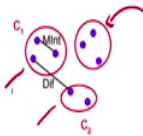
Vineth N B. (IIT-H)
3.5 Image Segmentation




Once you have this for any two adjacent regions, which have at least one edge connecting any of their vertices, we define a quantity called  $\text{Diff}(C_1, C_2)$  which is given by the minimum weight edge connecting these two regions. So for example, you consider all edges  $V_1, V_2$  such that  $V_1$  comes from region  $C_1$  and  $V_2$  comes from region  $C_2$ , so all edges between those two regions and you take the minimum weight of the edges that connect these two regions.


(Refer Slide Time: 15:46)

### Graph-based Segmentation

- A predicate  $D(C_1, C_2)$  for any two regions  $C_1$  and  $C_2$  is defined as:
 
$$D(C_1, C_2) = \begin{cases} \text{true,} & \text{if } \text{Diff}(C_1, C_2) > \text{MInt}(C_1, C_2) \\ \text{false,} & \text{otherwise} \end{cases}$$



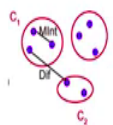
Vineth N B. (IIT-H)
3.5 Image Segmentation


With these quantities defined, the method defines a predicate  $D$  for any two regions  $C_1$  and  $C_2$  as if  $\text{Diff}(C_1, C_2)$  is greater than the minimum internal difference between  $C_1$  and  $C_2$ , then you say it is true, otherwise, you say it is false. This illustration should help you understand this a bit better. Remember,  $\text{diff}$  is the minimum edge weight between two regions, whereas minimum internal difference is a minimum internal difference within each of these between  $C_1$  and  $C_2$ , whichever is the minimum. So, if the difference between the two regions is greater than the minimum internal difference, you do not want to do anything.

(Refer Slide Time: 16:36)



### Graph-based Segmentation


A predicate  $D(C_1, C_2)$  for any two regions  $C_1$  and  $C_2$  is defined as:

$$D(C_1, C_2) = \begin{cases} \text{true,} & \text{if } \text{Diff}(C_1, C_2) > \text{MInt}(C_1, C_2) \\ \text{false,} & \text{otherwise} \end{cases}$$


For any two regions, if the predicate  $D$  evaluates to false, regions are merged. Else, regions are considered separate.

⇒ This algorithm merges any two regions whose difference is smaller than minimum internal difference of these two regions.

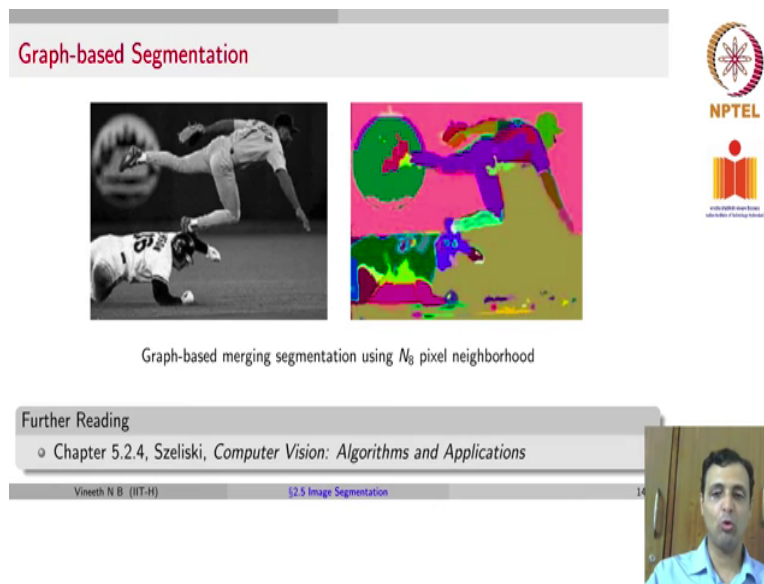


Vineth N B. (IIT-H)
5.2.5 Image Segmentation
13 / 31

If it is the other way, regions would be merged. It means if this predicate turns out to be false you know that  $\text{diff}$  is equal or lesser than minimum internal difference, which means  $C_2$  is as close to  $C_1$  as the farthest separated pixels in  $C_1$  so you may as well merge  $C_1$  and  $C_2$ . That is the main idea with this particular method. So to summarize, this algorithm merges any two regions whose difference is smaller than the minimum internal difference of these two regions.

(Refer Slide Time: 17:18)

**Graph-based Segmentation**



Graph-based merging segmentation using  $N_8$  pixel neighborhood

Further Reading

- Chapter 5.2.4, Szeliski, *Computer Vision: Algorithms and Applications*

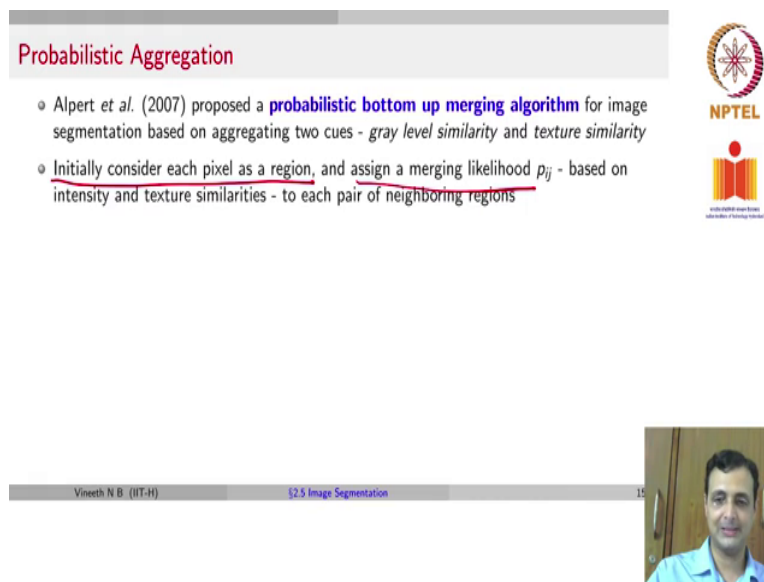
Vineth N B. (IIT-H) 5.2.5 Image Segmentation 14

Here is an example of how this works using the  $N_8$  pixel neighborhood. So you can see that the results are fairly good, the different players get separated reasonably well, parts of the body gets separated reasonably well. If you are more interested, please look at chapter 5.2.4 in Szeliski's book.

(Refer Slide Time: 17:41)

**Probabilistic Aggregation**

- Alpert et al. (2007) proposed a **probabilistic bottom up merging algorithm** for image segmentation based on aggregating two cues - *gray level similarity* and *texture similarity*
- Initially consider each pixel as a region, and assign a merging likelihood  $p_{ij}$  - based on intensity and texture similarities - to each pair of neighboring regions



Vineth N B. (IIT-H) 5.2.5 Image Segmentation 15

The next method we are going to look at is another popular method called probabilistic aggregation, which was proposed by Alpert et al in 2007. This method is again, a region merging algorithm and it is a probabilistic approach as the name suggests, and it uses two different cues, intensity values for the gray level values, as well as some texture content of the specific regions in the image.

So initially, the method considers each pixel as a region and then you assign a merging likelihood  $p_{ij}$ . So for every two pixels,  $i$  and  $j$ , there is a likelihood  $p_{ij}$ , which says how likely are these two pixels to merge? And how do you assign that  $p_{ij}$ ? Based on their intensity and texture similarities.



We are not going to go into each of these details. I will point you to the reference if you are interested in knowing how exactly it is implemented you can look at the paper for those details, but it is broadly based on intensity. You know how to compute intensity by now you know how to compute texture we talked about it the last time, so the  $p_{ij}$  is based on intensity and texture similarities between the two pixels.


(Refer Slide Time: 19:00)

### Probabilistic Aggregation

- Alpert *et al.* (2007) proposed a **probabilistic bottom up merging algorithm** for image segmentation based on aggregating two cues - *gray level similarity* and *texture similarity*
- Initially consider each pixel as a region, and assign a merging likelihood  $p_{ij}$  - based on intensity and texture similarities - to each pair of neighboring regions
- Given a graph  $G^{[s-1]} = (V^{[s-1]}, E^{[s-1]})$ ,  $G^{[s]}$  is constructed by selecting subset of seed nodes  $C \subset V^{[s-1]}$ , we merge nodes/regions if they are *strongly coupled* to regions in  $C$ . Strong coupling is defined as:
 
$$\frac{\sum_{j \in C} p_{ij}}{\sum_{j \in V} p_{ij}} > \text{threshold} \quad (\text{usually set to } 0.2)$$
- Once a segmentation is identified at a coarser level, assignments are propagated to their finer level "children", followed by further coarsening

Credit: Szeliski



Now, given a graph in a particular iteration, let us say it is an  $(S-1)$ -th iteration the graph is given by a set of vertices  $V^{(S-1)}$  and  $E^{(S-1)}$ . So the graph at the next iteration  $G^{(S)}$ , where you hope to merge a few regions is obtained by you consider a subset of node  $C$  from  $V^{(S-1)}$  and you merge nodes or regions if they are strongly coupled to regions in  $C$ .

So what you would do here is you would consider those pixels so which means you would say that if you took a subset of nodes in  $V^{(S-1)}$ , whichever other nodes, right, we would ideally consider other nodes which are in  $V^{(S-1)} - C$ . So those will be the other pixels in  $V^{(S-1)}$ . You consider those pixels and for whichever pixels you have  $V$  summation of  $p_{ij}$   $j$  belonging to  $C$  divided by summation of  $p_{ij}$   $j$  belonging to  $V^{(S-1)} - C$ , when this ratio is greater than a threshold, you know that there are certain pixels that are quite likely to merge as the pixels in the region  $C$  itself.

So, if this ratio is greater than a threshold which the paper recommends to set to point 2 then you merge those images.




(Refer Slide Time: 20:35)

### Probabilistic Aggregation

- Alpert *et al.* (2007) proposed a **probabilistic bottom up merging algorithm** for image segmentation based on aggregating two cues - *gray level similarity* and *texture similarity*
- Initially consider each pixel as a region, and assign a merging likelihood  $p_{ij}$  - based on intensity and texture similarities - to each pair of neighboring regions
- Given a graph  $G^{[s-1]} = (V^{[s-1]}, E^{[s-1]})$ ,  $G^{[s]}$  is constructed by selecting subset of seed nodes  $C \subset V^{[s-1]}$ , we merge nodes/regions if they are *strongly coupled* to regions in  $C$ . Strong coupling is defined as:
 
$$\frac{\sum_{j \in C} p_{ij}}{\sum_{j \in V} p_{ij}} > \text{threshold (usually set to 0.2)}$$
- Once a segmentation is identified at a coarser level, assignments are propagated to their finer level "children", followed by further coarsening

Credit: Szeliski

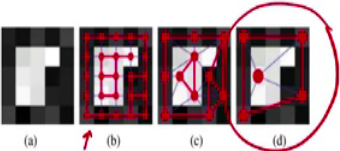
Vineth N B. (IIT-H)      5.2.5 Image Segmentation      15

And once you merge those regions, you propagate those assignments to the final level children, which would be the property of any region merging algorithm and you do this coarsening process recursively.

(Refer Slide Time: 20:54)

### Probabilistic Aggregation






**Figure 5.15** Coarse to fine node aggregation in segmentation by weighted aggregation (SWA) (Sharon, Galun, Sharon *et al.* 2006) © 2006 Macmillan Publishers Ltd [Nature]: (a) original gray-level pixel grid; (b) inter-pixel couplings, where thicker lines indicate stronger couplings; (c) after one level of coarsening, where each original pixel is strongly coupled to one of the coarse-level nodes; (d) after two levels of coarsening.

**Further Reading**

- Chapter 5.2.5, Szeliski, *Computer Vision: Algorithms and Applications*

Image Credit: Szeliski

Vineth N B. (IIT-H)      5.2.5 Image Segmentation      16

Here is an example of how probabilistic aggregation works in practice. So this is an image taken from Szeliski's book. So you can see here that image A is the original gray-level pixel grid, so it is a 5x5 image. So you can see the inter-pixel couplings in B image so a thick edge



means a stronger coupling, a thin edge means a weaker coupling so you can see that the pixels that are white are strongly coupled to each other, because of the intensity similarity.


Then in an image  $C$ , you perform one level of coarsening where you combine edges that are close to other edges, that you take one particular pixel when you start which would be  $C$ , then you take all the other  $V^{(S-1)} - C$  of all the other pixels in  $V^{(S-1)} - C$  and then you would try to find out which of them has a higher probability of merging based on intensity similarity and texture similarity again.

And then you keep repeating this process, you can see that after two levels of coarsening, you get a fairly good estimate of different regions in the image.

(Refer Slide Time: 22:12)

**Mean Shift Segmentation**




- A mode-finding technique based on non-parametric density estimation
- Feature vectors of each pixel in the image are assumed to be samples from an unknown probability distribution



- We estimate p.d.f. using non-parametric estimation and find its *modes*.
- Image is segmented pixel-wise by considering every set of pixels which climb to the same mode as a consistent segment.

Image Credit: Szeliski

Vineth N.B. (IIT-H) 3.5 Image Segmentation 17



Moving on to another popular method known as Mean Shift Segmentation, this method is a different approach and it uses a mode finding technique based on non-parametric density estimation.

So, non-parametric density estimation is a standard procedure for that people in machine learning use to estimate the density of any given set of observations. So the feature vectors of each pixel in the image are assumed to be some samples from some unknown probability distribution and we are going to estimate the probability density function of the image and find its modes.


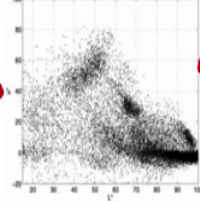
Remember, in some sense the modes would be like what we had for the watershed segmentation. Although, in that case we did a minimum, now we are going to talk about a

maximum. And then the image is finally segmented. Once you identify the modes in the image the image is segmented pixel wise by considering every set of pixels which climb to the same mode, as a consistent segment we will describe this for the next few slides.

(Refer Slide Time: 23:24)

### Mean Shift: Example

- Consider an example image below on the left. The graph on the right shows the distribution of  $L^*u^*$  features of each pixel (in the  $L^*u^*v^*/CIELUV$  space<sup>1</sup>)







- Our aim is to obtain modes of distribution on the right, without actually explicitly computing the density function! How to do this?

Image Credit: Comaniciu and Meer

<sup>1</sup><https://en.wikipedia.org/wiki/CIELUV>

Vineeth N B. (IIT-H) 3.5 Image Segmentation

So, let us consider an example image as you can see here, so, it is fairly real world image. On the right, you see the representation of the image. So, we talked about color spaces in week one. So, one of the color spaces that is popularly discussed is known as the  $L^*u^*v^*$  or LUV color space. So, this is simply for simplicity a plot of only the  $L^*u^*$  features of this image just for simplicity of plotting.

So, in this feature space, we ideally want to find out what are the modes of the overall distribution of the intensities in the  $L^*u^*$  space. So how do you find the modes of such a distribution?

(Refer Slide Time: 24:13)

rate the mode finding approach.

ving the data with kernel of width  $h$ , where  $k$  is

$$\sum_i k \frac{\|x - x_i\|^2}{h^2}$$

### Mean Shift: Example

- To find the modes (peaks), mean shift restarts.
- First, we pick a guess  $y_0$  for a local ma:

Image Segmentation 19 / 31 Vineeth N B (IIT-H)

If you recall your studies in machine learning, you would recall that one of the popular upload approaches to estimate the probability density function or modes would be through kernel density estimation. So we are going to use the same method here to be able to estimate what is the PDF and what are the modes of the distribution of intensities in a particular space while we have taken example, as the LUV space, you could do such a mod finding in any other space for that matter too.

Just for simplicity, we are going to consider a one dimensional example. So remember once again, that this is a one dimensional signal, which can be extrapolated to image as a 2D signal. So the way we would estimate this function is, we would convolve the data let us assume that you have observations given by these impulses here so you can see those vertical lines, those are your observations that are given, so you would estimate your density function  $f$ , which is what we want to estimate by convolving, your data, your observations with some kernel of width  $h$ .



Let us assume that you have a kernel  $k$ , so you could do it as a falling of function where  $h$  is the width, and you have  $x-x_i$  to be your function, your quadratic function that you are falling off over.

So convolving your observations with this kernel will give you an estimate of what is your original  $F$  function, that your observations were coming from? What do we do with this?


(Refer Slide Time: 25:49)

Mean Shift: Example

- To find the modes (peaks), mean shift uses a gradient ascent method with multiple restarts.
- First, we pick a guess  $y_0$  for a local maximum, which can be a random input data point  $x_i$ .



Vineeth N B. (IIT-H) 3.5 Image Segmentation 20



So to find the modes, mean shift segmentation uses a gradient ascent method with multiple restarts. Why ascent? Because we want to find the maximum of or the modes of that probability distribution.

So we start with some guests,  $y_0$  for a local maximum, which could just be any point, remember, we are going to perform gradient ascent, which is an iterative procedure. So you start with some point and at every iteration you add some constant times the gradient to the previous estimate and you update your estimate again, and again and again. So you have a guess,  $y_0$ , which could be any random point  $x_i$ .



(Refer Slide Time: 26:35)

Mean Shift: Example


- To find the modes (peaks), mean shift uses a gradient ascent method with multiple restarts.
- First, we pick a guess  $y_0$  for a local maximum, which can be a random input data point  $x_i$ .
- Then, we calculate the gradient of the density estimate  $f(\mathbf{x})$  at  $y_0$  and take an ascent step in that direction.

$$\nabla f(\mathbf{x}) = \sum_i (\mathbf{x}_i - \mathbf{x}) g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h^2}\right)$$

where  $g(\cdot) = -k'(\cdot)$ , the derivative of kernel  $k$



Vineeth N B. (IIT-H) 3.5 Image Segmentation 20



So we calculate the density or the gradient of the density estimate  $f(x)$ , we just talked about how  $f(x)$  is computed using that kernel. How do you choose that kernel? We will talk about it in a moment. So you have the gradient of the density estimate  $f(x)$  at  $y_0$  and we take an essence step in that direction. So the gradient is going to be given by something like this summation  $(x_i - x)g$  of this quantity where  $g$  is nothing but  $k$  prime or the derivative of the kernel  $k$ , which is written as the simple derivative.



(Refer Slide Time: 27:14)

**Mean Shift: Example**


- The gradient of the density function can be re-written as:

$$\nabla f(x) = \left[ \sum_i G(x - x_i) \right] m(x) \text{ where } m(x) = \frac{\sum_i x_i G(x - x_i)}{\sum_i G(x - x_i)} - x$$

where  $G(x - x_i) = g\left(\frac{\|x - x_i\|^2}{h^2}\right)$

Vineth N B. (IIT-H) 2.5 Image Segmentation 21





**Mean Shift: Example**


- To find the modes (peaks), mean shift uses a gradient ascent method with multiple restarts.
- First, we pick a guess  $y_0$  for a local maximum, which can be a random input data point  $x_i$ .
- Then, we calculate the gradient of the density estimate  $f(x)$  at  $y_0$  and take an ascent step in that direction.

$$\nabla f(x) = \sum_i (x_i - x) g\left(\frac{\|x - x_i\|^2}{h^2}\right)$$

where  $g(\cdot) = -k'(\cdot)$ , the derivative of kernel  $k$

Vineth N B. (IIT-H) 2.5 Image Segmentation 22



So, the gradient of the density function can actually be written slightly differently, you can write the gradient again, let us see the previous slide. So, this was the gradient that we just had  $(x_i - x)g\left(\frac{\|x - x_i\|^2}{h^2}\right)$  when  $g$  is  $-k'(\cdot)$  or the gradient of kernel.

So we are going to show that this can actually be written slightly differently, where the gradient can be written as  $\Sigma G(x-x_i)$ , where  $G(x-x_i)$  is simply the same small  $g$  that we saw on the previous slide into  $m(x)$  where  $m(x)$  is like a weighted contribution of the  $X$  values that you are considering in this particular combination. So  $\frac{\Sigma x_i G(x-x_i)}{\Sigma G(x-x_i)}$ .

(Refer Slide Time: 28:16)



**Mean Shift: Example**

- The gradient of the density function can be re-written as:


$$\nabla f(\mathbf{x}) = \left[ \sum_i G(\mathbf{x} - \mathbf{x}_i) \right] m(\mathbf{x}) \text{ where } m(\mathbf{x}) = \frac{\sum_i \mathbf{x}_i G(\mathbf{x} - \mathbf{x}_i)}{\sum_i G(\mathbf{x} - \mathbf{x}_i)} - \mathbf{x}$$

where  $G(\mathbf{x} - \mathbf{x}_i) = g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h^2}\right)$

- How? Homework!** ←
- The vector  $m(\mathbf{x})$  is known as the **mean shift**, the difference between  $\mathbf{x}$  and the gradient weighted mean of the neighbors around  $\mathbf{x}$ .

Vineth N B. (IIT-H) 3.5 Image Segmentation 21



Why is this a correct way of writing from the previous slide? I am going to leave it to you for homework. It is not too difficult to find this out, try to work it out stepwise and you will actually get the answer.



So this vector  $m(x)$ , which corresponds to the gradient is also known as the mean shift. It is the difference between  $x$  sorry, I did not point you to this  $-x$  here, but the mean shift is the difference between  $x$  and the gradient weighted mean of the neighbors around  $x$ ,  $x$  is where you place the filter and  $(x-x_i)$ s are the values in the range of that filter and you do convolution very similar to how we saw convolution earlier. In this case, we are only talking about 1D convolution for simplicity.


So the vector  $m(x)$  is the mean shift the difference between  $x$  and the gradient weighted mean of the neighbors around  $x$ .

(Refer Slide Time: 29:13)

### Mean Shift: Example

- The gradient of the density function can be re-written as:
 
$$\nabla f(\mathbf{x}) = \left[ \sum_i G(\mathbf{x} - \mathbf{x}_i) \right] m(\mathbf{x}) \text{ where } m(\mathbf{x}) = \frac{\sum_i \mathbf{x}_i G(\mathbf{x} - \mathbf{x}_i)}{\sum_i G(\mathbf{x} - \mathbf{x}_i)}$$
- where  $G(\mathbf{x} - \mathbf{x}_i) = g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h^2}\right)$
- How? Homework!**
- The vector  $m(\mathbf{x})$  is known as the **mean shift**, the difference between  $\mathbf{x}$  and the gradient weighted mean of the neighbors around  $\mathbf{x}$ .
- Current estimate of the mode  $\mathbf{y}_k$  at iteration  $k$  is replaced by its locally weighted mean:
 
$$\mathbf{y}_{k+1} = \mathbf{y}_k + m(\mathbf{y}_k) = \frac{\sum_i \mathbf{x}_i G(\mathbf{y}_k - \mathbf{x}_i)}{\sum_i G(\mathbf{y}_k - \mathbf{x}_i)}$$

Vineth N B. (IIT-H)
3.5 Image Segmentation




And once you get your mean shift, which is also your gradient, in this particular case, subject to a scalar multiple, then your next iteration in your gradient descent, your gradient ascent, you are going to say  $\mathbf{y}_{k+1}$  is equal to  $\mathbf{y}_k + m(\mathbf{y}_k)$ , which can also be written this way.

Remember, we just have  $\mathbf{y}_k$  and  $\mathbf{x}$  would get canceled because that is where we are placing  $\mathbf{y}_k$  at this point in time and you will be left only with  $\mathbf{y}_k$  as the quantity which will become  $\mathbf{y}_{k+1}$ .

(Refer Slide Time: 29:58)


### Mean Shift Segmentation

- Relies on selecting a suitable kernel width  $h$
- ~~Above description~~ strictly color based, however, better results can be obtained by working with feature vectors which include both color and location

Readings

- Chapter 5.3.2, Szeliski, *Computer Vision: Algorithms and Applications*

Vineth N B. (IIT-H)
3.5 Image Segmentation


And you keep doing this process iteratively and you would finally reach the mode of the distribution, and then you would find out all the neighboring points which by climbing would lead to this mode will form one segment.

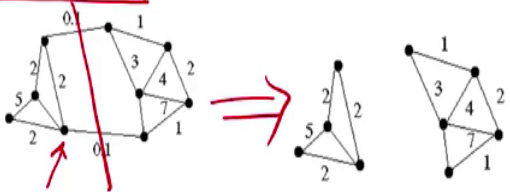
Other points, which by climbing would reach to other modes, would go to those segments. Clearly, this method relies on selecting a suitable kernel width in addition to the kernel itself, we use a quadratic kernel here what width you choose is also important and will impact the final result and that is chosen empirically.

In the above description, the approach seemed to be color-based we took it in the LUV space. You can also consider other kinds of features, other kinds of feature spaces to find the modes in it. More interested, please look at chapter 5.3.2 in Szeliski's book.

(Refer Slide Time: 30:55)

### Normalized Cuts for Segmentation

- **Region-splitting method** where a graph representing pixels in an image is successively split into parts
- **Edge weights between pixels in graph measure their similarity**



- Graph split into two parts by finding and deleting a **cut-set** with minimum sum of weights i.e., a **min-cut**







Image Credit: K Shafique  
 Vineth N B (IIT-H) 52.5 Image Segmentation 23



One more method which is not a region merging method, but a region splitting method is normalized cups for segmentation. Once again, a very popular method was used for many years. Once again a graph-based method, but a different approach.

In this case, a graph representing pixels in an image is successively split into parts. It is not about merging this time, it is about splitting. How do you split it, you mark edge weights between pixels based on their similarity, so if two pixels are very similar, maybe they have similar intensities, you give them a high edge weight and if two pixels are further apart in their similarity you give them low weight.

So here is an example of such a graph that is constructed. So the goal of this would be to find the min cut, you are not going to work out the entire math here. If you are interested, you can look at the references and learn more about it, but you try to find the min cut, which is going



to give you two different regions as shown here. And those edges when you remove, you get the corresponding two segments, which is your original objective.



(Refer Slide Time: 32:19)

**Normalized Cuts**

- **Min-cut** is defined as the sum of all weights being cut:

$$\min_{A, B} \text{cut}(A, B) = \sum_{i \in A, j \in B} w_{ij}$$

where  $A$  and  $B$  are two disjoint subsets of  $V$  (set of all vertices)



Vineth N B. (IIT-H) 52.5 Image Segmentation 24

The min-cut is defined as sum of all weights being cut. So you consider two sets  $A$  and  $B$ , two sets of vertices and for all  $i$  belonging to  $A$  and  $j$  belonging to  $B$ , you add up those edge weights wherever you get for whichever choice of  $A$  and  $B$ , you get a minimum cut value that is the cut that you are going to choose to divide the graph into two segments.

So, clearly as you can observe here, this method only divides two segments at a time. So if you want more segments, you will have to do this more and more times or probably use other heuristics if you wanted to combine them later into fewer signals.

(Refer Slide Time: 33:11)

**Normalized Cuts**

- **Min-cut** is defined as the sum of all weights being cut:

$$\text{cut}(A, B) = \sum_{i \in A, j \in B} w_{ij}$$




where  $A$  and  $B$  are two disjoint subsets of  $V$  (set of all vertices)

- Using **min-cut** criterion directly can result in trivial solutions such as isolating a single pixel.
- This paved the way for the formulation of a **Normalized Cut**, defined as:

$$\text{Ncut}(A, B) = \frac{\text{cut}(A, B)}{\text{assoc}(A, V)} + \frac{\text{cut}(A, B)}{\text{assoc}(B, V)}$$

Credit: Szeliski

Vineth N B. (IIT-H) 3.5 Image Segmentation 24



But one of the problems with using the min-cut is that it can also result in trivial solutions. Why so? Can you think about it? Why does this min-cut problem result in trivial solutions?

The answer is very simple, when  $A$  consists of say only one pixel, and  $B$  consists of say, another region of pixels, this summation may be restricted to a smaller set and hence may have a smaller value when you sum all of them up, which means min-cut could favor regions where there is only one pixel on one side, which may simply be an outlier and may not really form a good segmentation for the image.

How do you overcome this problem? This problem is overcome by a method called normalized cut, which improves the min-cut problem and the normalized cut is defined by

$$\frac{\text{cut}(A, B)}{\text{assoc}(A, V)} + \frac{\text{cut}(A, B)}{\text{assoc}(B, V)}$$

(Refer Slide Time: 34:29)

**Normalized Cuts**




- We define  $\text{assoc}(A, V) = \text{assoc}(A, A) + \text{assoc}(A, B)$  as the sum of all weights associated with vertices in  $A$  where:  
$$\text{assoc}(A, B) = \sum_{i \in A, j \in B} w_{ij}$$
- While computing an optimal normalized cut is NP-complete, there exist approximate solutions (Shi and Malik, 2000).

Readings

- Chapter 5.4, Szeliski, *Computer Vision: Algorithms and Applications*
- Shi and Malik, Normalized Cuts and Image Segmentation, IEEE TPAMI 2000.

Credit: Szeliski

Vineeth N B. (IIT-H) 32.5 Image Segmentation



We will see a moment what they are. Association is defined again, as sum of all the weights in  $A$  and  $V$  where  $\text{assoc}(A, V)$  is given by  $\text{assoc}(A, A) + \text{assoc}(A, B)$ . It is the sum of all the weights.




(Refer Slide Time: 34:50)

**Normalized Cuts**

- Min-cut** is defined as the sum of all weights being cut:  
$$\text{cut}(A, B) = \sum_{i \in A, j \in B} w_{ij}$$
where  $A$  and  $B$  are two disjoint subsets of  $V$  (set of all vertices)
- Using **min-cut criterion** directly can result in trivial solutions such as isolating a single pixel.
- This paved the way for the formulation of a **Normalized Cut**, defined as:  
$$\text{Ncut}(A, B) = \frac{\text{cut}(A, B)}{\text{assoc}(A, V)} + \frac{\text{cut}(A, B)}{\text{assoc}(B, V)}$$

Credit: Szeliski

Vineeth N B. (IIT-H) 32.5 Image Segmentation



And now what you have done by doing this association of  $A, V$  and association of  $B, V$  is to ensure that the denominator contains the number of pixels in each of these regions. So if there was only one pixel in one region, remember that one of these quantities will become very high as against another region, which had many more pixels where the denominator will pull down.

This ensures that you do not get trivial solutions such as what you saw in min-cut, but you get more useful solutions in practice.

(Refer Slide Time: 35:28)



### Normalized Cuts

- We define  $\text{assoc}(A, V) = \text{assoc}(A, A) + \text{assoc}(A, B)$  as the sum of all weights associated with vertices in  $A$  where:
 
$$\text{assoc}(A, B) = \sum_{i \in A, j \in B} w_{ij}$$
- While computing an optimal normalized cut is NP-complete, there exist approximate solutions (Shi and Malik, 2000).


Readings

- Chapter 5.4, Szeliski, *Computer Vision: Algorithms and Applications*
- Shi and Malik, Normalized Cuts and Image Segmentation, IEEE TPAMI 2000.

Credit: Szeliski

Vineth N B. (IIT-H) 32.5 Image Segmentation



So the normalized cut happens to be an NP-complete problem, but there are approximate solutions to solve this slightly more involved, but you can look at the paper called normalized cut and image segmentations if you are interested in how the method is actually implemented, that this is the overall formulation.

(Refer Slide Time: 35:49)

### Normalized Cuts

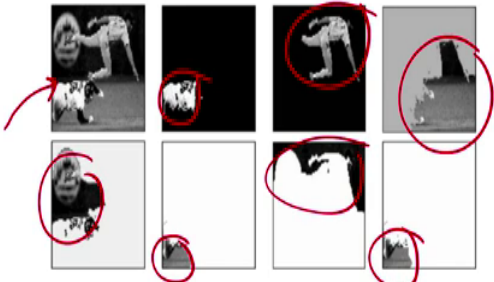





Image components returned by Normalized cuts algorithm

Image Credit: Shi and Malik

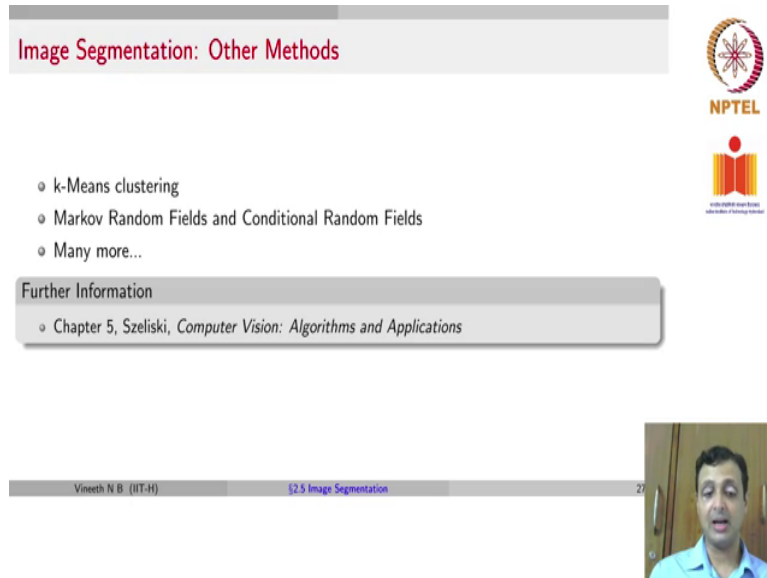



Vineth N B. (IIT-H) 32.5 Image Segmentation



Here is an example of how normalized cuts work. So this is the input image and you can see the various different regions gets segmented using normalized cuts in over the iterations of the method.

(Refer Slide Time: 36:08)

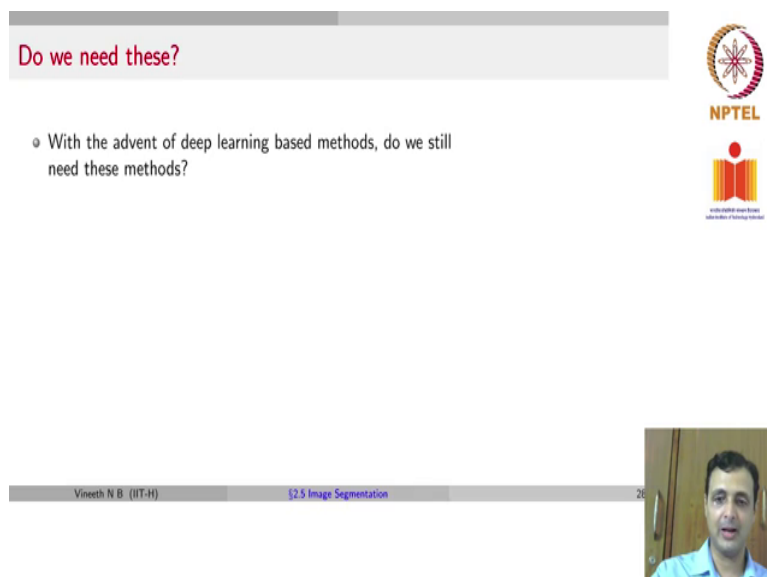


The slide is titled "Image Segmentation: Other Methods" in red text. It lists three bullet points: "k-Means clustering", "Markov Random Fields and Conditional Random Fields", and "Many more...". Below this is a section titled "Further Information" with a bullet point: "Chapter 5, Szeliski, *Computer Vision: Algorithms and Applications*". The slide includes the NPTEL logo and a small video inset of the speaker, Vineth N B. (IIT-H), in the bottom right corner. The footer of the slide reads "Vineth N B. (IIT-H) 32.5 Image Segmentation 27".

There have been many other methods for image segmentation too, such as simply using K means clustering. There have also been methods based on Markov Random Fields and Conditional Random Fields, and many more.

If you want to have a detailed understanding, please read chapter five of Szeliski's book there are more references at the end of this lecture too.

(Refer Slide Time: 36:29)



The slide is titled "Do we need these?" in red text. It contains a single bullet point: "With the advent of deep learning based methods, do we still need these methods?". The slide includes the NPTEL logo and a small video inset of the speaker, Vineth N B. (IIT-H), in the bottom right corner. The footer of the slide reads "Vineth N B. (IIT-H) 32.5 Image Segmentation 28".

But the question that people to ask now is, do we really need all of these segmentation methods. So think the entire course is based on deep learning for computer vision? Are not there deep learning methods that can segment images today? Do you really need all of this? It is a valid question for many tasks today, you do not need these methods. The purpose of covering these topics is to give you a context of computer vision in the pre deep learning era, but some of these methods have also been used in the deep learning era.

(Refer Slide Time: 37:06)

**Do we need these?**

- With the advent of deep learning based methods, do we still need these methods?
- Yes, to an extent. These classical segmentation methods actually inspired early versions of deep learning based methods for object detection and semantic segmentation (we will see this later)
- First R-CNN work (object detection method) used a version of min-cut segmentation method known as CPMC (Constrained Parametric Min Cuts) to generate region proposals for foreground segments

Vineth N B (IIT-H) 32.5 Image Segmentation 26

For those of you who are a little bit more aware about these areas, the earlier methods of deep learning that were used for detection of objects, which means given an image, you try to find an object and draw the bounding box as to where in the image that object lies that is the task of detection and there could be multiple instances of objects or multiple objects in an image.

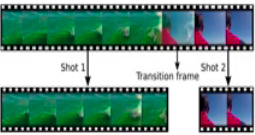
For the initial methods of deep learning for object detection, these kinds of segmentation methods but actually used to generate what are known as region proposals where the objects could lie. As I said, one of the first R-CNN, which is a region CNN work, that was used for object detection, used a min-cut segmentation method known as CPMC, Constraint Parametric Min Cuts, to generate region proposals for foreground segments.

We will talk about this in detail when we come to that section of this course.

(Refer Slide Time: 38:06)

**Beyond Images: Segmentation for Video**

- **Shot boundary detection** - a key problem in video segmentation: Divide a video into collection of *shots*, each taken from a single sequence of camera




- Another interesting problem is **motion segmentation**, where the aim is to detect and isolate motion in the video. Examples: a person running, a car moving, etc.

**Further Information**

- Chapter 15.2, 17.1.4, Forsyth, *Computer Vision: A Modern Approach*

*Image Credit: M Gygli*

Vineth N B. (IIT-H) 3.5 Image Segmentation 25



But that is the reason why image segmentation is useful for you to know at this time. Segmentation can also go beyond images into videos. If you go into videos, the principles for the methods are very similar, although, there is a third dimension that comes into the picture, but the kind of tasks that one wants to solve in videos is say shot boundary detection, which is an important problem in video segmentation, where you have to divide an entire video into shots.

Remember that if there is a very long video, and you want to understand the video, let us say you have a movie and you want to be able to understand the movie by automatically passing the video on the movie, you first have to divide the entire movie into shots, and then try to analyze each shot for understanding the scene understanding the characters so on and so forth.

So that is an important task in video segmentation. Another important task with videos is what is known as motion segmentation, where you may want to isolate the motion in the video of an object. So you could be wanting to isolate a person running. For example, there is a football game and you want to track a person moving with the ball from one end of the field to the other into the field, or you may want to track a car, so on and so forth.

So for further information on this, please look at chapters 15 and 17 of the David Forsyth book in the references of this course.

(Refer Slide Time: 39:37)

**Homework**

**Readings**

- Chapter 5, Szeliski, *Computer Vision: Algorithms and Applications*
- Chapter 15, 17.1.4, Forsyth, *Computer Vision: A Modern Approach*

**Questions**

- Derive the final expression for gradient of the kernel density function used in the mean shift method

Vineth N B. (IIT-H) 3.5 Image Segmentation

That concludes this lecture, which gives you an overview of how image segmentation methods were used in the pre deep learning era. Some of them as I said were used actually even with deep learning methods, although not as much at this time, but for more readings please do read these references given here and recall we left one derivation as homework for you, go back and see, how to derive the final expression for the gradient of the kernel density function used in the mean shift method.

(Refer Slide Time: 40:12)

**References**

- Jianbo Shi and J. Malik. "Normalized cuts and image segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (2000), pp. 888–905.
- D. Comaniciu and P. Meer. "Mean shift: a robust approach toward feature space analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.5 (2002), pp. 603–619.
- Richard Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. London: Springer-Verlag, 2011.
- Radhakrishna Achanta et al. "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods". In: *IEEE transactions on pattern analysis and machine intelligence* 34 (May 2012).
- David Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. 2 edition. Boston: Pearson Education India, 2015.

Vineth N B. (IIT-H) 3.5 Image Segmentation

There are some references.