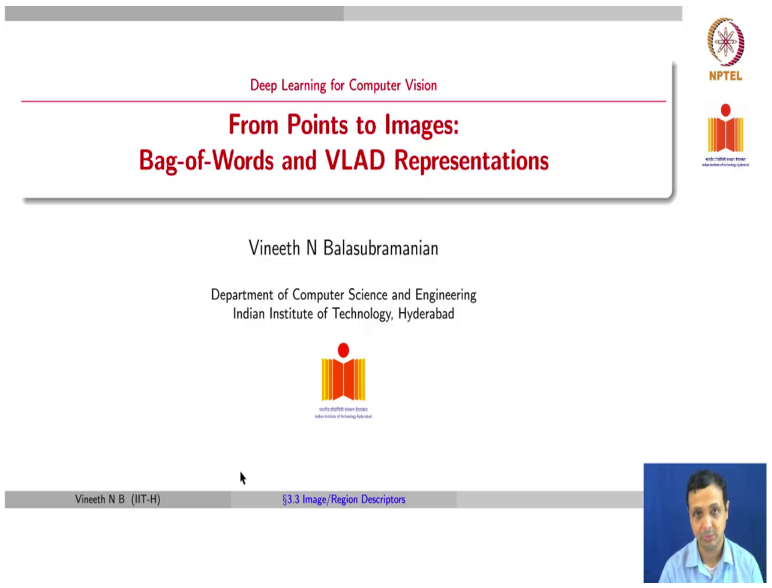
Deep Learning for Computer Vision
Prof. Vineeth N Balasubramanian
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Lecture 17
From Points to Images: Bag-of-Words and VLAD Representations

(Refer Slide Time: 0:15)



Deep Learning for Computer Vision

From Points to Images:
Bag-of-Words and VLAD Representations

Vineeth N Balasubramanian

Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

Moving on from the previous lecture, we will now talk about how you describe entire images. So far, we spoke about how to obtain descriptors for certain key points in images, which could be in several multiples in each image, how do you aggregate these descriptors to obtain a single descriptor for an image and why do we need this.

We talked about image panorama stitching, where you try to match key points. You may now want to match images themselves. For example, in image search or retrieval or even to perform image classification as we will see when we go through these slides. In this lecture, we will focus on two specific kinds of descriptors, the bag-of-words descriptor and another extension of it known as the VLAD descriptor.

(Refer Slide Time: 01:15)

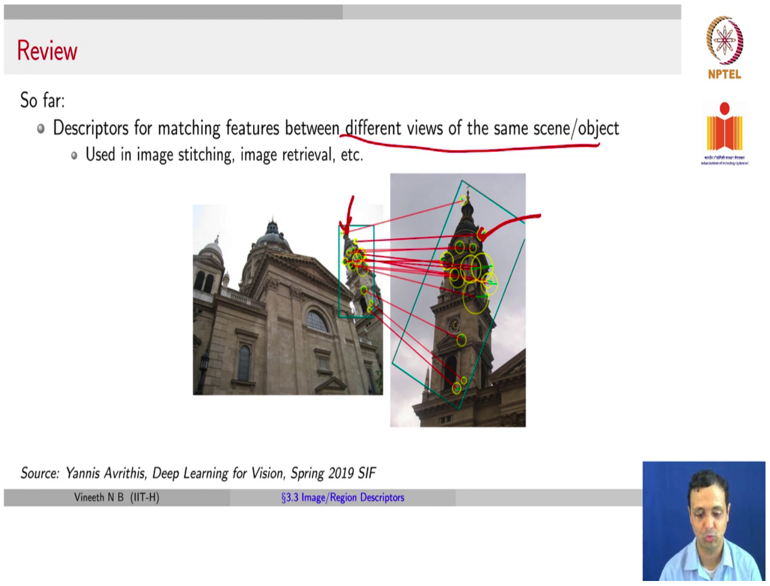
Acknowledgements

- Most of this lecture's slides are based on lectures of **Deep Learning for Vision** course taught by Prof Yannis Avrithis at Inria Rennes-Bretagne Atlantique

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

Once again, the lecture slides are based on Professor Yannis lectures at Inria Rennes.

(Refer Slide Time: 01:21)



Review

So far:

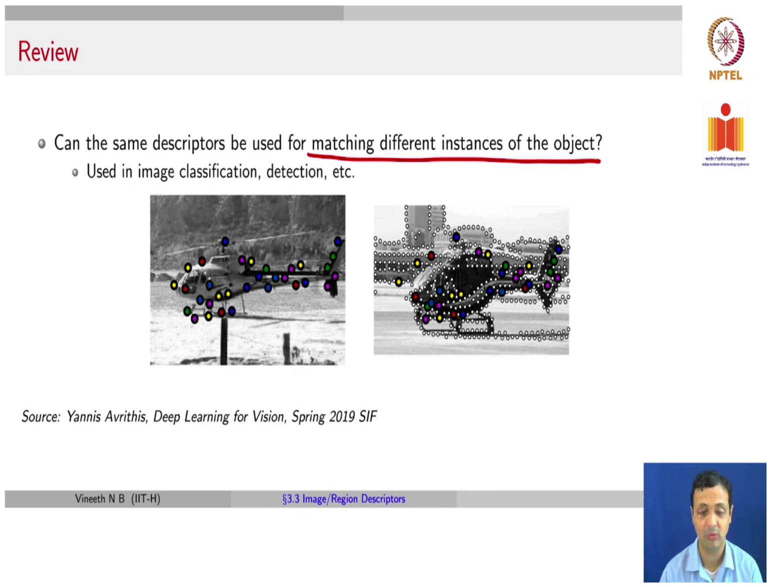
- Descriptors for matching features between different views of the same scene/object
  - Used in image stitching, image retrieval, etc.

Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

So we have spoken so far about obtaining key points from two different views of the same scene or object. You have two different views. You extract key points in both of these views. You get descriptors of key points in both of these views and you match them and perhaps try to match the geometry of the configuration between them using methods such as RANSAC or Hough transform, so on and so forth.

(Refer Slide Time: 01:54)



Review

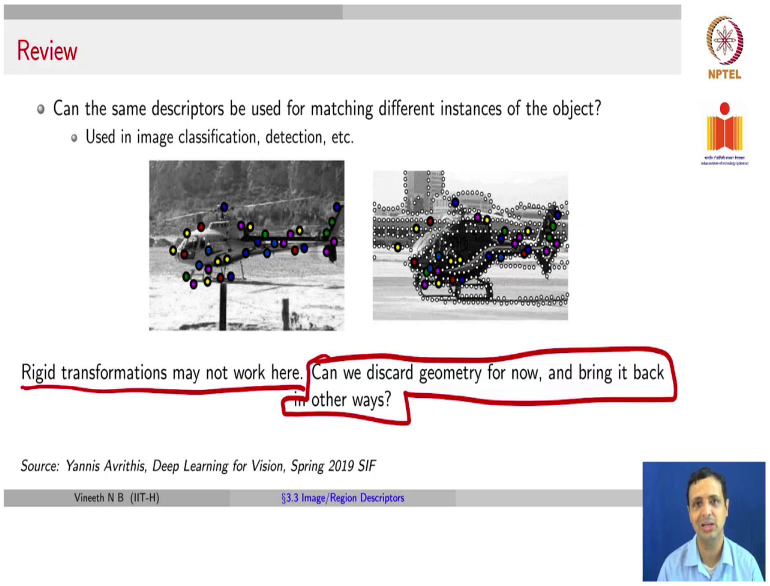
- Can the same descriptors be used for matching different instances of the object?
  - Used in image classification, detection, etc.

Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

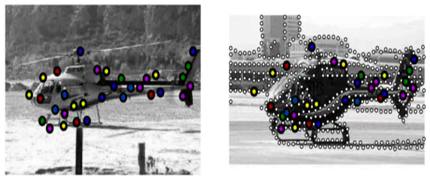
But the question that we are asking here is what if we want to match different instances of an object. The object is no more the same. In the earlier slide, it was the same object from different viewpoints. Now the object could be completely different. In this case, it is a helicopter, but two completely different helicopters. They are not the same object itself. Then how do you match or how do you search? For example, imagine your Google image search where you are going to search for helicopter kind of images. So you post an image and ask the system to retrieve similar images for you. How do you do this is what we are going to talk about now.

(Refer Slide Time: 02:37)



Review

- Can the same descriptors be used for matching different instances of the object?
  - Used in image classification, detection, etc.



Rigid transformations may not work here. Can we discard geometry for now, and bring it back in other ways?

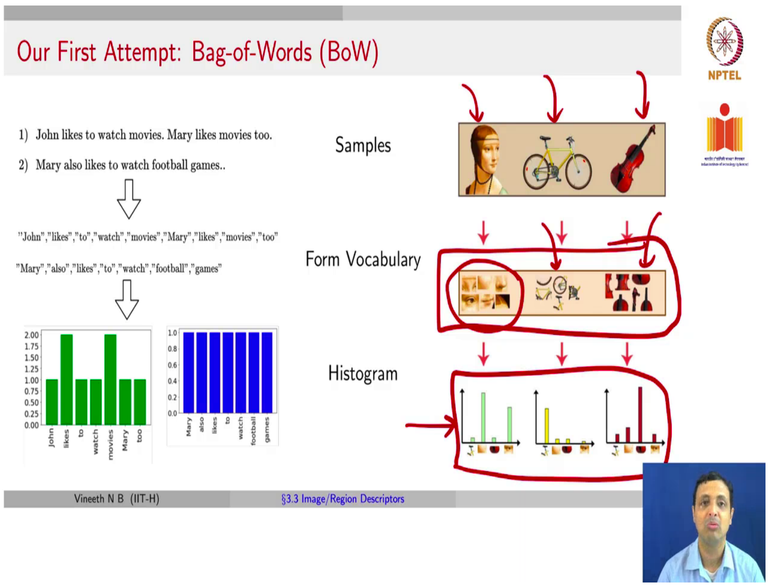
Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

The main observation here is that rigid transformations may not work here. So far, if you only wanted to match two different objects or scenes taken from different perspectives, you could consider one to be a rigid transformation of the other. You could try to estimate your affine parameters, your rotation parameters or your translation parameters, but then these are completely two different instances of an object. It may not be a rigid transformation at all.

So, ideally what we want to do is now to see if we can completely discard geometry itself, because it is no longer going to be a rigid transformation. Can we completely discard the idea of geometry? Try to match images in some other way beyond geometry. And then maybe later using other methods, we will try to bring back certain elements of geometry, little loosely into the mix.

(Refer Slide Time: 03:35)



Our First Attempt: Bag-of-Words (BoW)

1) John likes to watch movies. Mary likes movies too.
2) Mary also likes to watch football games..

"John","likes","to","watch","movies","Mary","likes","movies","too"

"Mary","also","likes","to","watch","football","games"

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

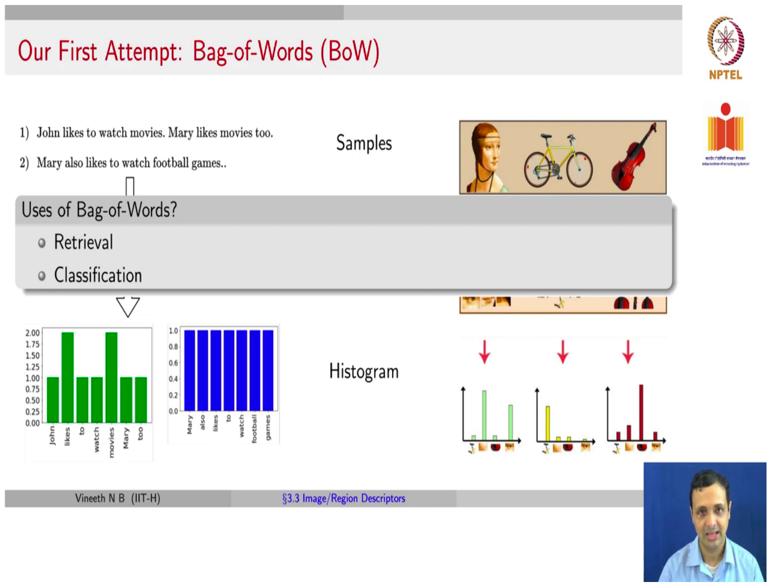
So our first attempt towards getting image level descriptors is a method known as bag-of-words, an old method has been used a lot in text before, but let us try to see how we use this in images. So we are going to start with a set of samples, a set of data points. Then you form a vocabulary of those data points. What do we mean by vocabulary, a set of common words that exist across images. And finally, based on the vocabulary, we are going to get a histogram, which is simply a frequency count of words in a certain document. In our case, visual words in a certain image. Let us try to describe that in more detail.

So if you had text information, your samples would simply be different sentences. Your vocabulary would be the words that occur in these sentences, all the possible words that you have occurring in your sentences. And finally, for each of these sentences, you would come up with a histogram of how many occurrences of a particular word in your vocabulary happened in this particular sentence. So, for each of your sentences, you can look up a vocabulary or a dictionary and try to see how many times each word in the vocabulary or dictionary occurred in this particular sentence.

Let us try to draw a parallel to images now. Let us say you have three different images in this particular case. So you form a vocabulary. So what does a vocabulary mean for images in text? It is perhaps simple. It is all the words that occur in your documents. But if you had images, what would be your vocabulary? So the way we are going to define vocabulary is very simple again. You take different parts of each of these images, group them together and see which of them correspond to something common. For example, it could be possible that there are six kinds of visual words that occur in the first image, which all look similar. Maybe they may look similar in color, in texture or in any other appearance of the representation you choose.

Similarly, the second image has a set of visual words. And the third image has a set of visual words. Once again, we define visual words as extracting several parts from an image. So if you have a set of images, you extract parts from all of those images and then try to pool them in some way and then group them and say these parts in the image have to, seem to have similar properties and I am going to group them into one particular word. We will see this a little bit more clearly over the next few slides. And once you get these visual words, similar to text, you have a histogram of how many times each of these visual words occurred in a given image. That histogram is what is drawn here for each of these three images.

(Refer Slide Time: 06:56)



Our First Attempt: Bag-of-Words (BoW)

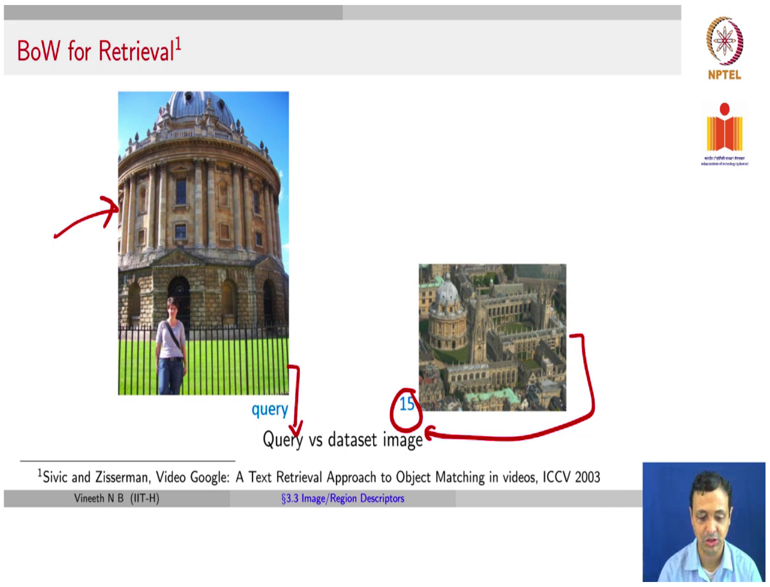
1) John likes to watch movies. Mary likes movies too.

2) Mary also likes to watch football games..

Uses of Bag-of-Words?
- Retrieval
- Classification

Let us see a more real world example to understand this a bit better. Before we go there, what could be used for bag-of-words? It could be used for image retrieval as well as image classification as we just work. Why, because this is a way to represent an entire image as a single vector. Let us see this using a more tangible example.

(Refer Slide Time: 07:17)



BoW for Retrieval[1]

Query vs dataset image

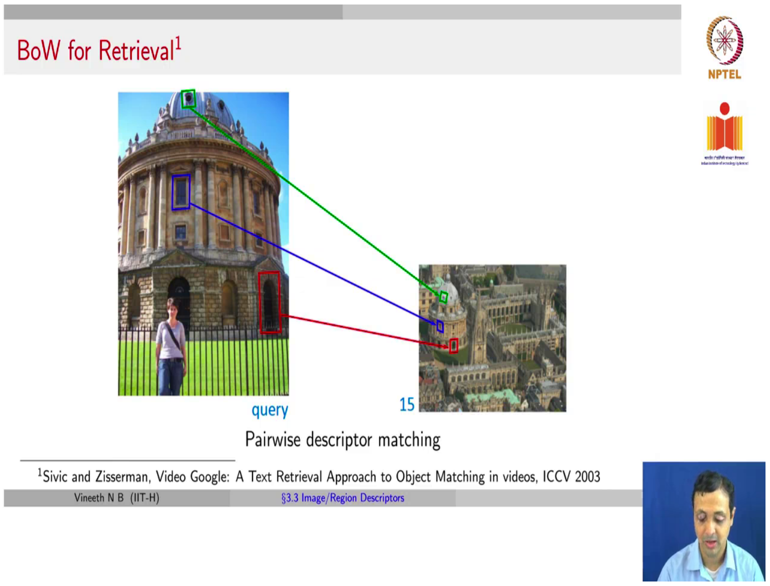
[1]Sivic and Zisserman, Video Google: A Text Retrieval Approach to Object Matching in videos, ICCV 2003

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

Here is an example of an image. And let us assume that this is the query image. And here is an image from a dataset. So you could imagine that the dataset images are your entire repository of images that you have. And the query image is like Google image search, where you put up this image and you are asking for a search engine to retrieve all images similar to this image. For example, you could upload a cat image on any search engine and ask the search engine to retrieve similar cat images, a very similar example here.

Let us assume that there is a 15th image in your dataset image, which has an occurrence of the similar structure. Now let us see what would happen earlier and how we would change things now.

(Refer Slide Time: 08:08)



BoW for Retrieval[1]

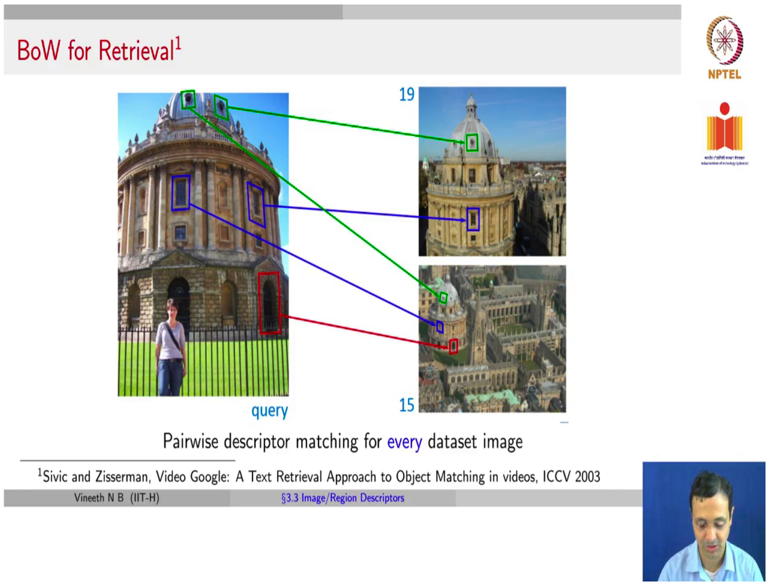
Pairwise descriptor matching

[1]Sivic and Zisserman, Video Google: A Text Retrieval Approach to Object Matching in videos, ICCV 2003

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

Earlier, we would take key points in each of these images, descriptors around these key points in each of these images and try to do a pairwise matching between the descriptors in both of these images. And this is obviously a time taking process, because if you had many key points, many descriptors in each image, you would ideally have to do a pairwise matching between every number in the first image versus every number in the reference image. This can get computationally expensive.

(Refer Slide Time: 08:43)



BoW for Retrieval[1]

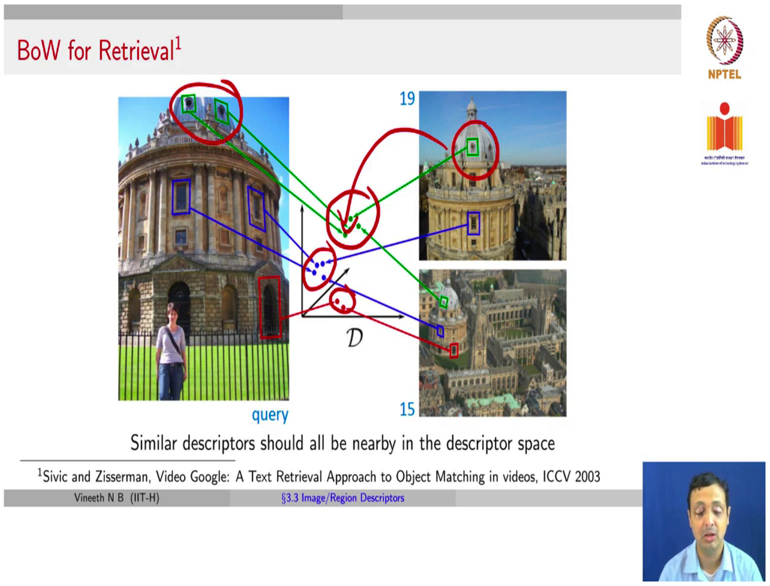
Pairwise descriptor matching for every dataset image

[1]Sivic and Zisserman, Video Google: A Text Retrieval Approach to Object Matching in videos, ICCV 2003

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

If you had another image, you would have to repeat the same process for as many times between the query image and the second reference image and so on and so forth for all reference images.

(Refer Slide Time: 08:55)



BoW for Retrieval[1]

Similar descriptors should all be nearby in the descriptor space

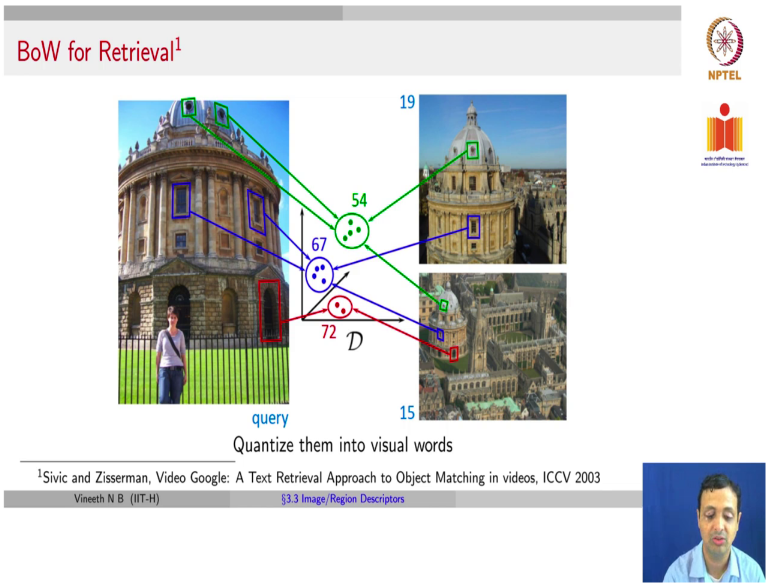
[1]Sivic and Zisserman, Video Google: A Text Retrieval Approach to Object Matching in videos, ICCV 2003

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

So how do we want to resolve this problem? So what we are going to say now is that if there were descriptors that were similar that would match between the two images, instead of matching them pairwise between the query image and the reference image, let us try to assume that similar descriptors will anyway correspond to similar representations in that representation space in a certain metric, let us say the Euclidean metric.

So a group of visual words or regions that are similar to each other would probably have similar representations and they all grouped together in the representation space. Similarly, for those regions marked with blue and the regions marked with red. They are similar and they would probably match in the representation space. How do we use this?

(Refer Slide Time: 09:46)



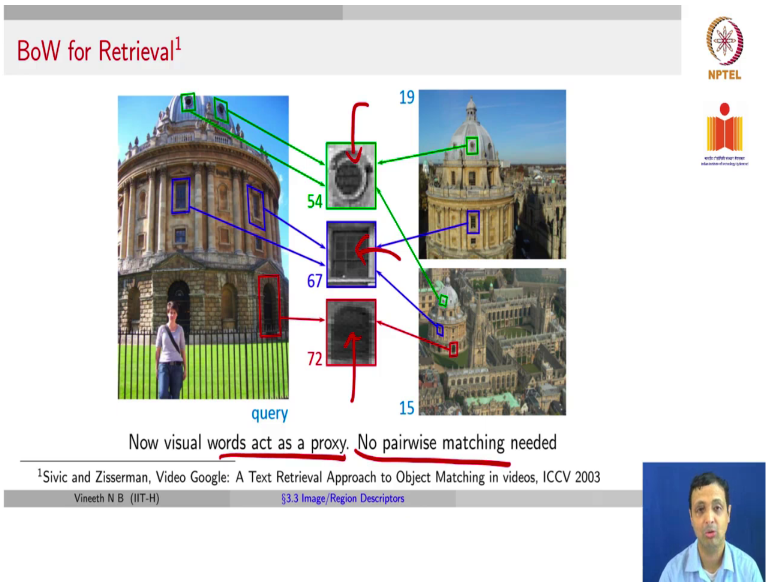
BoW for Retrieval[1]

Quantize them into visual words

[1]Sivic and Zisserman, Video Google: A Text Retrieval Approach to Object Matching in videos, ICCV 2003

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

(Refer Slide Time: 09: 52)



BoW for Retrieval[1]

Now visual words act as a proxy. No pairwise matching needed

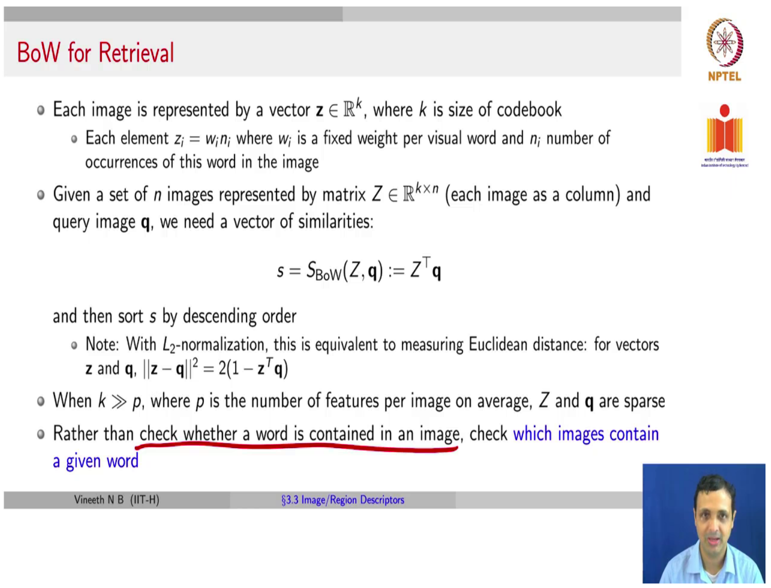
[1]Sivic and Zisserman, Video Google: A Text Retrieval Approach to Object Matching in videos, ICCV 2003

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

So what we would do now is to come up with a representation, a common representation for all similar descriptors, how would we do that? We could simply take your entire repository of images or training images, take visual words from them, which are common to each other and simply obtain their mean in that space. That becomes the visual word corresponding to all of those visual words in the repository of images or training images. So you would do this process offline before the retrieval process starts. So you could construct your visual words offline and keep them stored and ready when your search process actually begins. So these visual words are already stored.

Given a query image, you take the descriptors corresponding to the key points in your query image and now you simply match them with the visual word, which represents the mean of similar visual words in your reference images. You do not need to do pairwise matching with every descriptor in your reference word anymore. You would have a set of visual words that are common across your repository of images. You only need to match with them now to be able to get your answer. This makes this process very feasible to do as well as very effective. There is no pairwise matching required and your visual words now act as a proxy for your descriptors in your repository.

(Refer Slide Time: 11:20)



BoW for Retrieval

- Each image is represented by a vector $\mathbf{z} \in \mathbb{R}^k$, where k is size of codebook
  - Each element $z_i = w_i n_i$ where w_i is a fixed weight per visual word and n_i number of occurrences of this word in the image
- Given a set of n images represented by matrix $Z \in \mathbb{R}^{k \times n}$ (each image as a column) and query image $\mathbf{q}$, we need a vector of similarities:

$$s = S_{\text{BoW}}(Z, \mathbf{q}) := Z^\top \mathbf{q}$$

and then sort s by descending order
  - Note: With L_2-normalization, this is equivalent to measuring Euclidean distance: for vectors $\mathbf{z}$ and $\mathbf{q}$, $||\mathbf{z} - \mathbf{q}||^2 = 2(1 - \mathbf{z}^T \mathbf{q})$
- When $k \gg p$, where p is the number of features per image on average, Z and $\mathbf{q}$ are sparse
- Rather than check whether a word is contained in an image, check which images contain a given word

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

Now let us see this in a more well defined way. So let us imagine that your image now is represented by a vector z belonging to R^k, where k is the size of the codebook or the number of visual words that you choose, which would be a parameter that the user would set for a particular dataset. Each element of z, which is z_i is represented as $w_i n_i$, where w_i is a fixed weight per visual word. You can assign a weight. If you do not have a weight, they could just have uniform weights. And n_i is the number of occurrences of that particular word in this query image. I will repeat that. So w_i is a weight per visual word, which is optional. And n_i is the number of occurrences of that word in this image.

So given a set of n images in your repository in your reference images, which are represented by a Z in $R^{k\times n}$, remember there are going to be K visual words, and n such images. For each image, you would have a k dimensional vector. So, and then n such images becomes $k \times n$.

And for a query image, you would simply compare the similarities between your query images vector representation, which is going to count the number of visual words in the query image and the number of visual words in each of the reference images. You simply take a dot product or cosine distance between these two and you get a set of scores. You sort order your scores in descending order and that gives you the best match followed by the next best match so on and so forth.

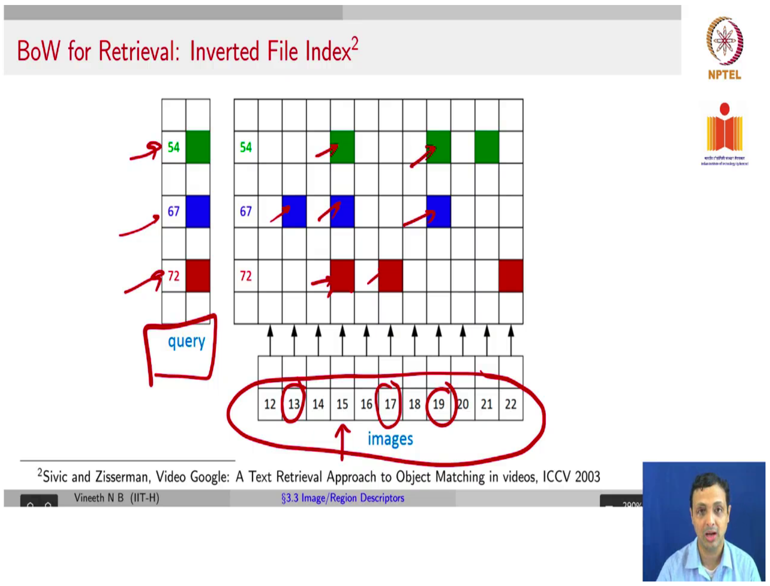
Note here that when we take a cosine distance based on say dot product, remember here what we are doing as a dot product in Z^Tq, you would take every column of Z, which corresponds to one specific image, and you are taking an individual dot product of that reference image with this query image q, the second reference image with the query image q, so on and so forth. And whichever has the highest dot product that image is what you would recommend as the closest match to your query image.

Just to point out to you a more general observation that if you take a dot product, it is similar to measuring Euclidean distance, the similarity that you measure using a dot product or a cosine distance, cosine similarity is complementary to the use of Euclidean distance. Why, because $|| z - q ||^2$ now can also be written as $2(1 - z^Tq)$. Work it out as homework. So, which means something closer as cosine similarity will have a low Euclidean distance. A high cosine similarity will be a low Euclidean distance. So, in some sense, these two matrices comparing similarities or distances are complimentary.

An important observation here is when $k >> p$, where p is the number of features per image on average. Remember k is the number of visual words that you have in your codebook, which would be user defined and p is the average number of features that we have in a given image. We saw with examples like SIFT that could be several hundreds. If $k >> p$, then z and q are going to be sparse. Why, if you have 1,000 visual words and say 100 in each image, then obviously there are going to be quite a few, at least 900 in a given image that will have zero count, because those visual words do not exist in this image. Remember k is the number of visual words and p is the number of features in a given image. So, if $k >> p$, both z and q are going to be sparse.

So to make the computation more effective here, rather than check whether a word is in a given image, which is what you would typically do, you check for every word in a given image, but then the entire image could be a very sparse representation of the possible set of words. We can inward this problem and check which images contain a particular word and we will see now that this can be a more effective way to compute similarity in this scenario. Let us see this a bit more intuitively.

(Refer Slide Time: 16:11)



BoW for Retrieval: Inverted File Index[2]

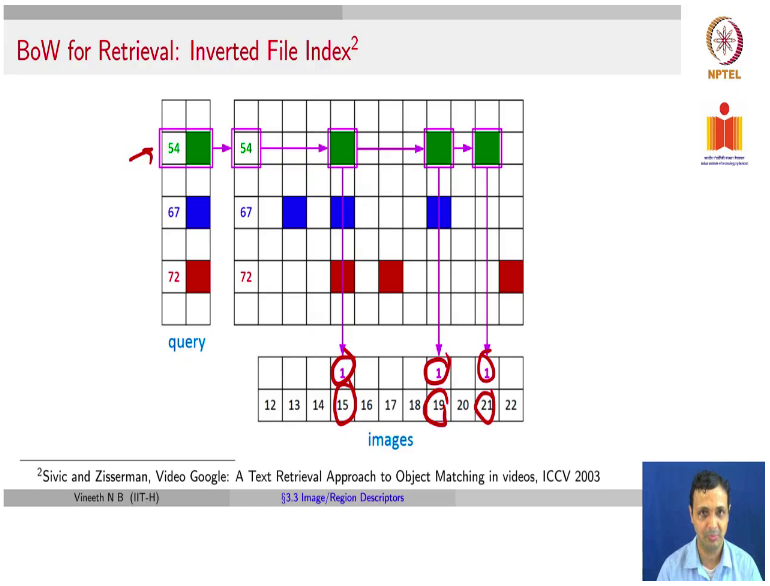
[2]Sivic and Zisserman, Video Google: A Text Retrieval Approach to Object Matching in videos, ICCV 2003

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

So let us imagine now that you have a query image, which has visual words, 54, 67 and 72, these are your visual words, remember you have a bunch of visual words and every image has a certain number of occurrences of each of these visual words. Let us assume now that you have just one occurrence of visual words, 54, 67 and 72 in a given query image. 54, 67 and 72 are just random numbers. You could have taken any other numbers. This is just to explain.

Now, if you have a repository of images that you want to retrieve from, let us say they are given by a set of numbers in your dataset. So you now try to draw this out. So you look at each of these images. Let us see the 15th image in your repository had the 72nd word, the 67th word and 54th word, whereas the 13th image has only the 67th word, the 17th image has only the 72nd word, the 19th image has the 67th and the 54th word and so on and so forth. Now what do you do?

(Refer Slide Time: 17:27)



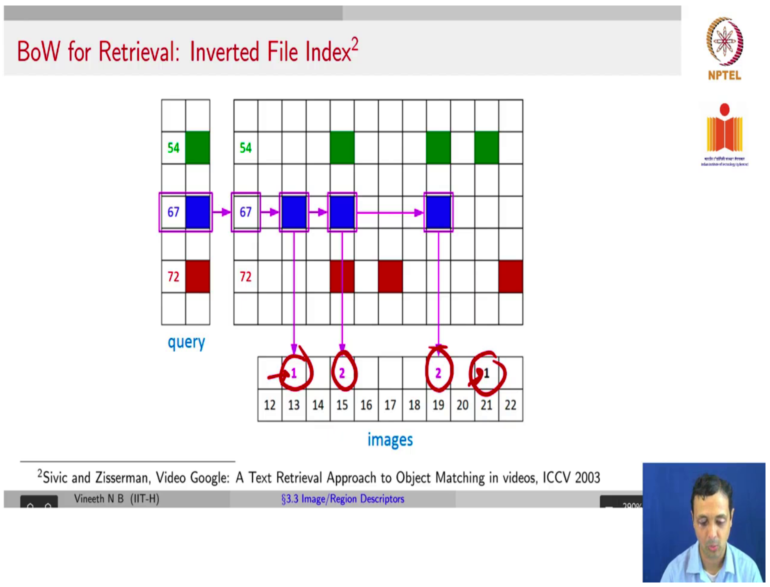
BoW for Retrieval: Inverted File Index[2]

[2]Sivic and Zisserman, Video Google: A Text Retrieval Approach to Object Matching in videos, ICCV 2003

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

So you take one of these visual words, which was occurring in your query image, and you try to see which of your query images also had that particular visual word and you add a count for that particular image. Similarly, 19 would have got a count of 1, 21 would get a count of 1 for these images. So you are trying to see, you are now trying to compare which of these repository images contain the same word which is there in your query image.

(Refer Slide Time: 18:03)



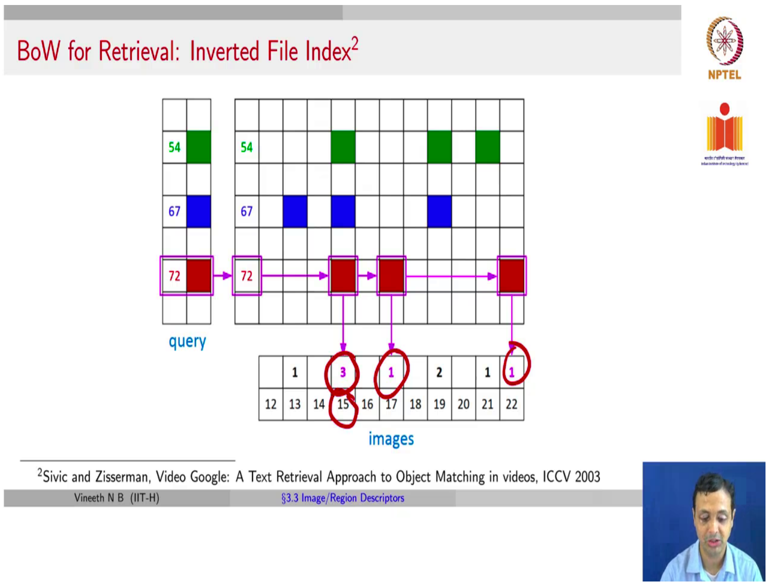
BoW for Retrieval: Inverted File Index[2]

[2]Sivic and Zisserman, Video Google: A Text Retrieval Approach to Object Matching in videos, ICCV 2003

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

Similarly, you would do this for the 67th word and you will get an increase in count for these two images, an increase in count for 13th and 21st just had a green, there was no increase in count, but the 13th image has an increase in count now.

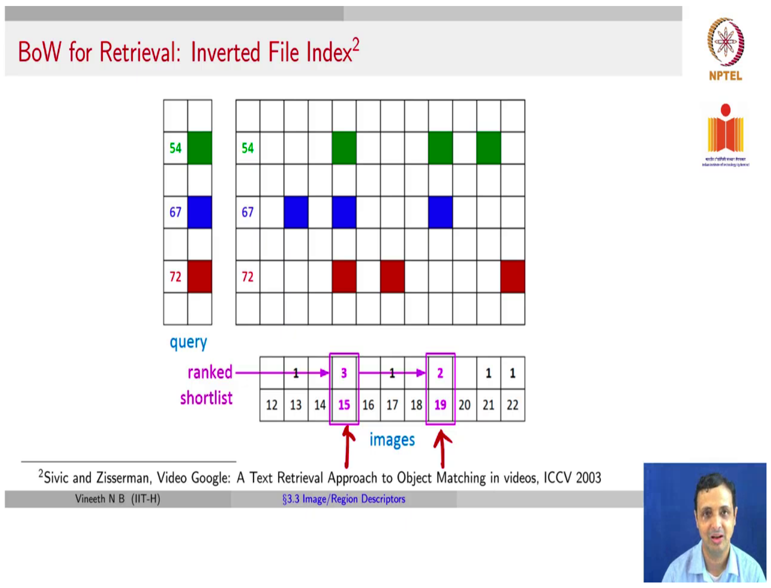
(Refer Slide Time: 18:25)



BoW for Retrieval: Inverted File Index[2]

[2]Sivic and Zisserman, Video Google: A Text Retrieval Approach to Object Matching in videos, ICCV 2003

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors



BoW for Retrieval: Inverted File Index[2]

[2]Sivic and Zisserman, Video Google: A Text Retrieval Approach to Object Matching in videos, ICCV 2003

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

Similarly, you would do this for the red visual word or the 72nd visual word. And at this time the 15th image gets one more count. The 17th image gets a new count added here. And the 22nd image gets a new count added here. So it is evident from this that the 15th image has the highest count with respect to the query image and you now rank order all your repository images based on a short list with respect to these visual words.

So you find that the 15th image has the best match. The second best match is the 19th image with two common visual words. And you repeat this process to be able to get your best match from your repository of images and this would be known as the image retrieval problem.

(Refer Slide Time: 19:16)

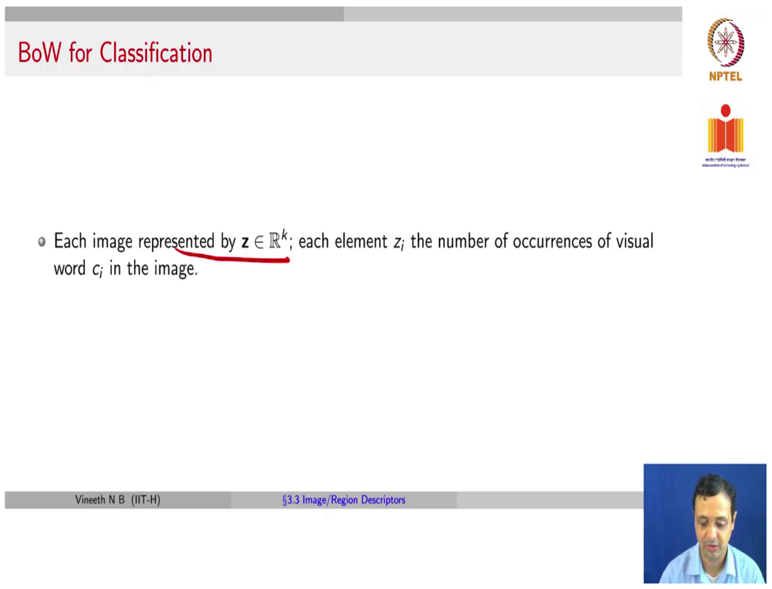
BoW for Classification

How?

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

Let us try to ask how you would extend bag-of-words to classification. Now, we talked about retrieval, but we now want to classify a given image as belonging to a particular scene. For example, in the previous image that we saw, maybe that building has a particular name and there are many monuments that you have and you want to classify an image as belonging to a particular monument. So you want to treat this now as a classification problem. How do you adapt this for classification?

(Refer Slide Time: 19:49)



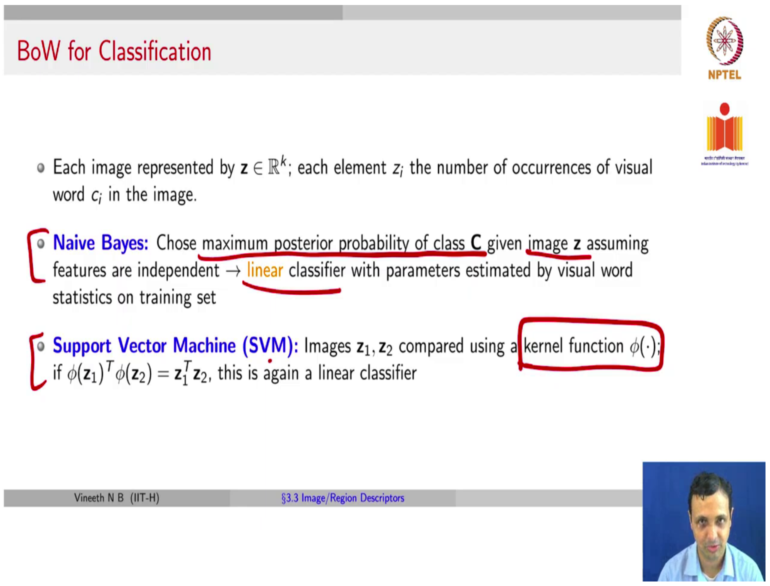
BoW for Classification

- Each image represented by $\mathbf{z} \in \mathbb{R}^k$; each element z_i the number of occurrences of visual word c_i in the image.

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

The way you would do it is once again, you represent an image by a vector z belonging to R^k, where k is the number of the visual words, very similar to how we did it for retrieval problems.

(Refer Slide Time: 20:00)



BoW for Classification

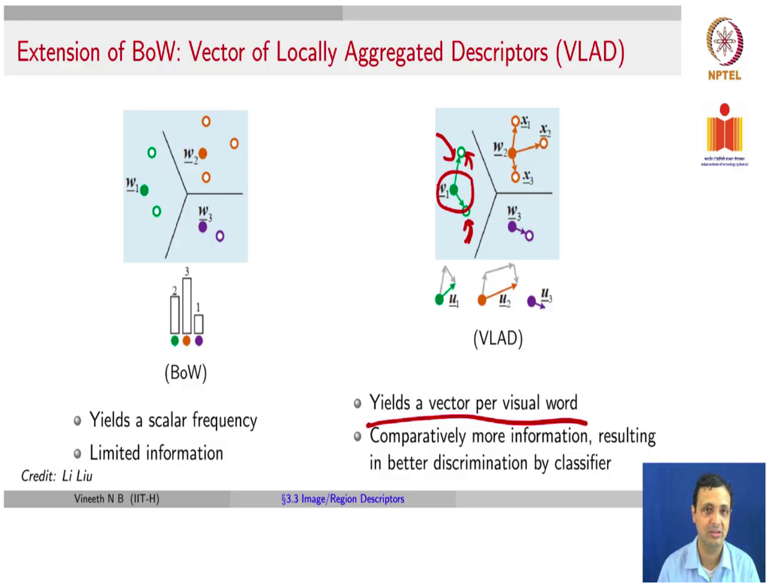
- Each image represented by $\mathbf{z} \in \mathbb{R}^k$; each element z_i the number of occurrences of visual word c_i in the image.
- **Naive Bayes:** Chose maximum posterior probability of class **C** given image **z** assuming features are independent → linear classifier with parameters estimated by visual word statistics on training set
- **Support Vector Machine (SVM):** Images $\mathbf{z}_1, \mathbf{z}_2$ compared using a kernel function $\phi(\cdot)$; if $\phi(\mathbf{z}_1)^T\phi(\mathbf{z}_2) = \mathbf{z}_1^T\mathbf{z}_2$, this is again a linear classifier

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

But once you do this now, you can use a classifier to be able to classify these, the frequency of visual words. So every image now becomes a vector representation of a set of visual words, a histogram of visual words. And you would have a set of similar images in your training data, which you would have also had a class label associated with them, because remember classification is a supervised learning problem. So now you could just use a classifier such as, say, naive bayes or support vector machines or any other classifier for that matter.

If you are using naive bayes, you would estimate the maximum posterior probability of a class C given an image z assuming features are independent, assuming the presence or the count of each visual word is independent. You would simply run a naive bayes classifier in this particular scenario. Remember naive bayes would become a linear classifier. You could do this using a support vector machine. You could use an appropriate kernel function in a support vector machine to perform the classification or for that matter you could use any other classifier too.

(Refer Slide Time: 21:06)



Extension of BoW: Vector of Locally Aggregated Descriptors (VLAD)

- Yields a scalar frequency
- Limited information

- Yields a vector per visual word
- Comparatively more information, resulting in better discrimination by classifier

Credit: Li Liu

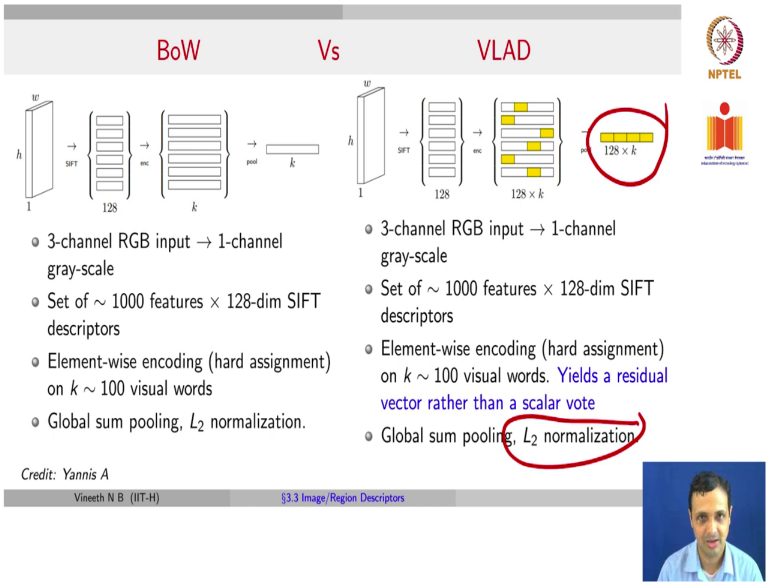
Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

An extension of the bag-of-words is known as the vector of locally aggregated descriptors or VLAD. It is very similar to bag-of-words with a small, but a significant difference. Bag-of-words as we just saw would give you a scalar frequency of how many times a particular visual word, which is obtained by clustering a lot of regions in your training images or repository of images, how many times each of those visual words occurred in a query image. So in some sense, this gives you limited information. You do not have geometry here or you do not have what were the exact regions in your query image. You are going to map them any way to a common visual word, which is the average of similar visual words in your depository.

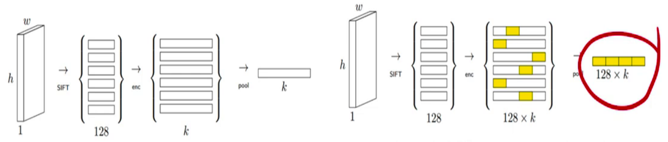
In VLAD, instead, what you do is you again have visual words, all of that remains the same, but now you have, instead of a scalar frequency, you have a vector per visual words, which looks at how far each of these features in your query image are from the visual word. So you have a visual word which you would have obtained by clustering groups of features in your training images.

Now, given a new feature in your query image, you look for how far that is from the visual word you would get a residual vector. Similarly you take another feature that is map to the same visual word and see how far that is with respect to the visual word and you add up all the residuals and you would get one vector residual, which is the sum of all residuals that get mapped to the visual word that corresponds to the visual word. So it is not a scalar frequency anymore, it is a vector of how far other features that map to this visual word are in each dimension. This gives you a little bit more information that could be more effective.

(Refer Slide Time: 23:09)



BoW Vs VLAD



- 3-channel RGB input → 1-channel gray-scale
- Set of ∼ 1000 features × 128-dim SIFT descriptors
- Element-wise encoding (hard assignment) on $k \sim 100$ visual words
- Global sum pooling, L_2 normalization.

- 3-channel RGB input → 1-channel gray-scale
- Set of ∼ 1000 features × 128-dim SIFT descriptors
- Element-wise encoding (hard assignment) on $k \sim 100$ visual words. Yields a residual vector rather than a scalar vote
- Global sum pooling, L_2 normalization.

Credit: Yannis A

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

Let us see this a bit more clearly. If we had a bag-of-words representation, given a color image, which is a three channel RGB input, you first converted to gray-scale. You could adapt bag-of-words to color two, we are just taking a simpler example here. So once you have that, you get a set of say 1,000 features and you convert that say to one dimensional. If you had 1,000 key points, you convert that to 128 dimensional SIFT descriptors for each of those 1,000 key points. Then you do an element-wise encoding to say 100 visual words, let us assume you had 100 visual words. You take each of those 1,000 features and map them to 100 visual words.

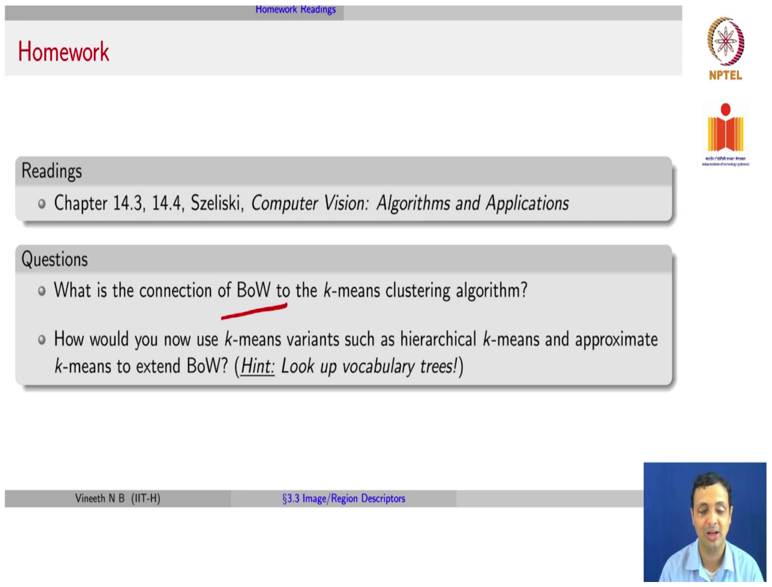
So you would have k such visual words and you would end up counting them, counting the occurrences of each of those 1,000 features with respect to one of these 100 visual words, you get a set of frequencies. Then you could do a global sum pooling and an L2 normalization of that final histogram vector to get your final bag-of-words representation.

On the other hand with VLAD, you would do the same. You would convert the three channel RGB input into a one channel gray-scale. You would once again get 1,000 features in your query image, convert that to 128 dimensional representations for each of those 1,000 features. You would again assign each of those 1,000 features to one of k visual words. But this is where the difference comes that you are now not going to get a scalar representation or a histogram, you are going to get a residual vector for each of those k visual words, which means this now is going to be a 128 cross k, into k rather representation rather than a k dimensional representation.

Because for each visual word, you are going to see for all the features that got mapped or all the key points that got mapped to a particular visual word what was the residual for each dimension for each of those 128 dimensions.

And finally, you would get a 128 into k representation, which you would L_2 normalize and use for practice. L_2 normalization is simply to ensure that the vector becomes a unit norm, L_2 norm to be one and that is the VLAD descriptor.

(Refer Slide Time: 25:39)



Homework Readings

Homework

Readings

- Chapter 14.3, 14.4, Szeliski, *Computer Vision: Algorithms and Applications*

Questions

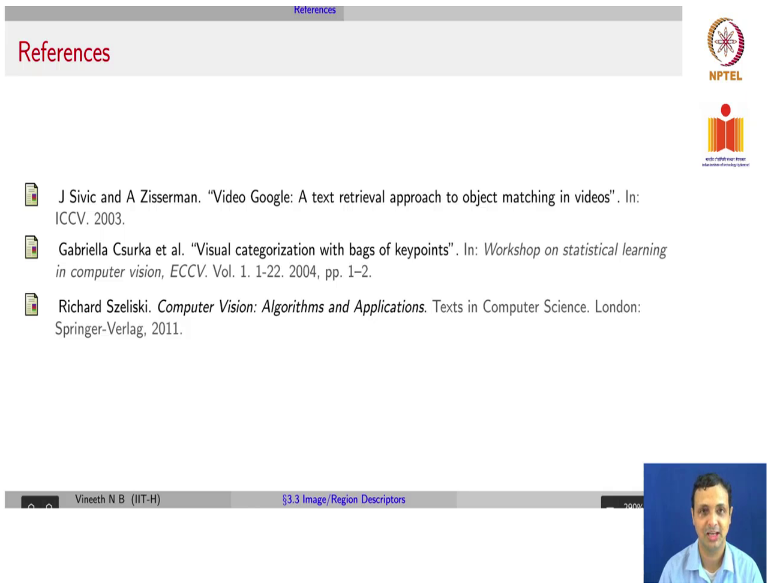
- What is the connection of BoW to the *k*-means clustering algorithm?
- How would you now use *k*-means variants such as hierarchical *k*-means and approximate *k*-means to extend BoW? (*Hint: Look up vocabulary trees!*)

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

To conclude this lecture, please read chapter 14.3 and 14.4 of Szeliski's book. And something for you to think about at the end of this lecture is how is bag-of-words connected to the k-means clustering algorithm. We already, I think, spoke about it briefly during the lecture, but think about it and answer this more carefully.

Assuming you understood that bag-of-words is connected to k-means clustering, how are extensions of k-means clustering, such as hierarchical k-means clustering or approximate k-means clustering relevant for the bag-of-words problem. Your hint here is to look up what are known as vocabulary trees.

(Refer Slide Time: 26:22)



References

J Sivic and A Zisserman. "Video Google: A text retrieval approach to object matching in videos". In: ICCV. 2003.

Gabriella Csurka et al. "Visual categorization with bags of keypoints". In: *Workshop on statistical learning in computer vision, ECCV*. Vol. 1. 1-22. 2004, pp. 1–2.

Richard Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. London: Springer-Verlag, 2011.

Vineeth N B (IIT-H) §3.3 Image/Region Descriptors

And here are some references for this lecture.