User-Centric Computing for Human-Computer Interaction

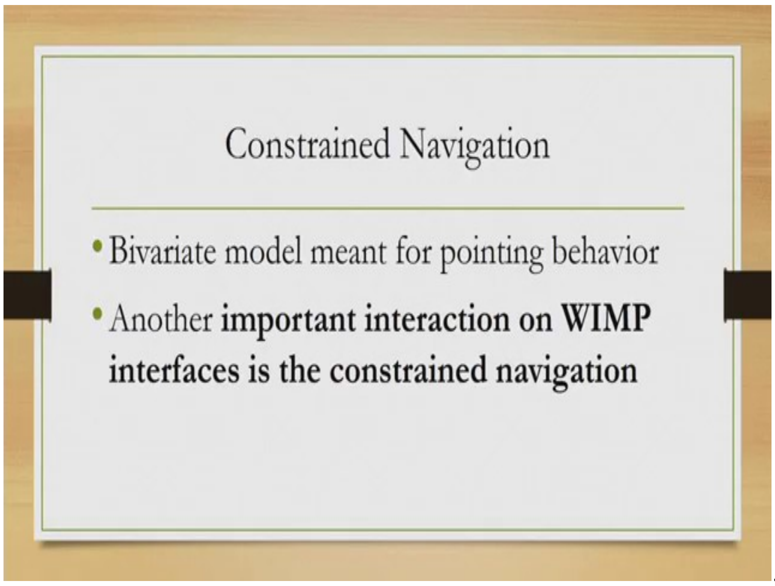
Prof. Samit Bhattacharya
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture - 19
The Steering Law for constrained navigation

Hello and welcome to lecture number 19 in the course User- Centric computing for Human- Computer Interaction. So, as is our convention before we go to the actual content, we will recap what we have learned so far. Now in this week and in previous few lectures we started our discussion on contemporary computational user models, the models that are of recent origin and used for contemporary interaction designs or interface designs.

So far we have discussed two models, one is the bivariate pointing model and the other one is trivariate pointing model. Today we are going to discuss one more model that is called the steering law which is used for modeling performance in constrained navigation.

(Refer Slide Time: 01:41)



Constrained Navigation

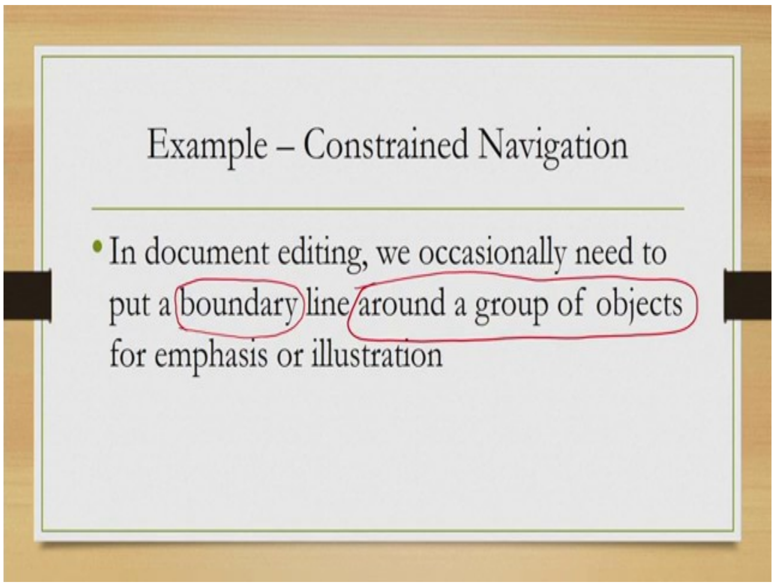
- Bivariate model meant for pointing behavior
- Another **important interaction on WIMP interfaces is the constrained navigation**

.

Now, this bivariate model or the trivariate models that we have learned in the previous lecture are used for modeling performance in pointing tasks. However, pointing is not the only activity that we perform with a WIMP interface.

In fact, another popular activity that are typically performed on WIMP interfaces are the constrained navigational tasks or activities. So, what do we mean by the term constrained navigation? Let us try to understand the concept in terms of one example.

(Refer Slide Time: 02:23)

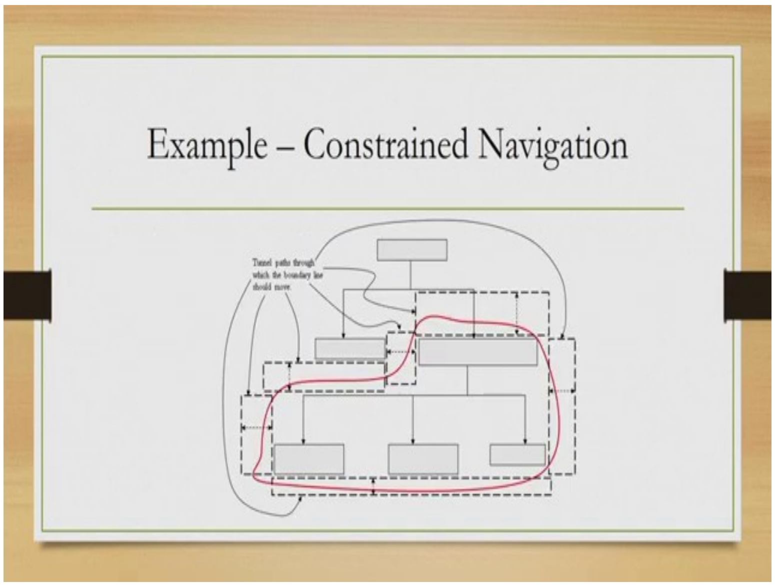


Example – Constrained Navigation

- In document editing, we occasionally need to put a boundary line around a group of objects for emphasis or illustration

Sometimes when you are editing a document you may need to put a boundary line around some objects. For example, suppose this is the text that you can see on the slide and we want to put a boundary line or a circle around one word or a line for example, suppose I want to put a circle around the word boundary in this way. Now, as you can see putting this circle around the word or if I want to put a circle around the group of word like this one is not the same as that of pointing, here we need to do it in a different way. What is the difference?

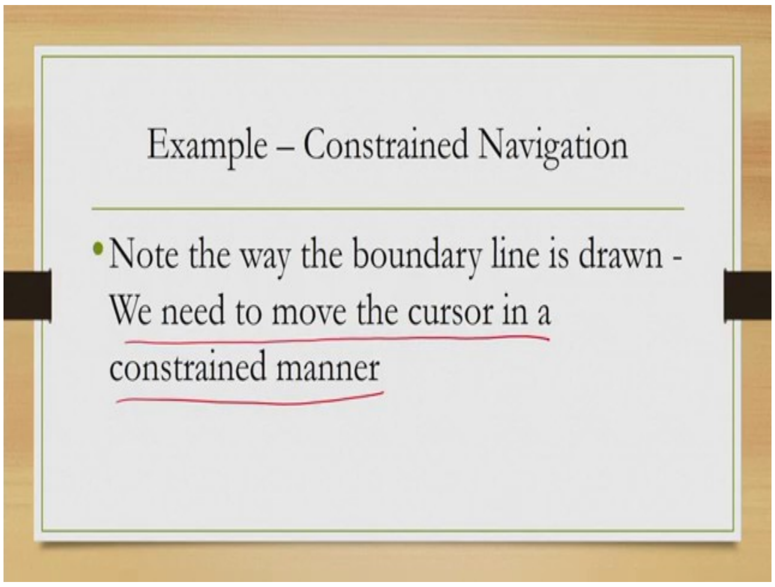
(Refer Slide Time: 03:11)



Example – Constrained Navigation

Consider this example, here the red line shows the boundary line we have put around a group of objects which is actually representative of a hierarchy. Now, but the time being ignored the dotted lines and the dotted boxes and the comments, just concentrate on the boundary line represented by the red continuous line.

(Refer Slide Time: 03:43)

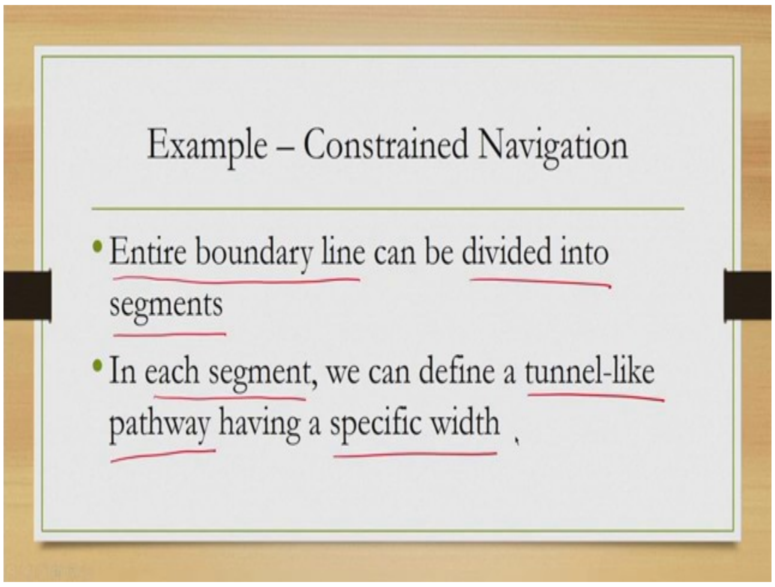


Example – Constrained Navigation

- Note the way the boundary line is drawn - We need to move the cursor in a constrained manner

So, when we have put this boundary line how we have drawn this or typically when you are asked to put a circle around a word or the circle around the group of words that we have just done before how it is done. So, essentially, we need to move the cursor in a constrained manner; that means we are not doing this in a free flowing way. So, we need to take care of many things before we actually put the line. Now, if we look closely a little bit closer into the task what we can see?

(Refer Slide Time: 04:17)

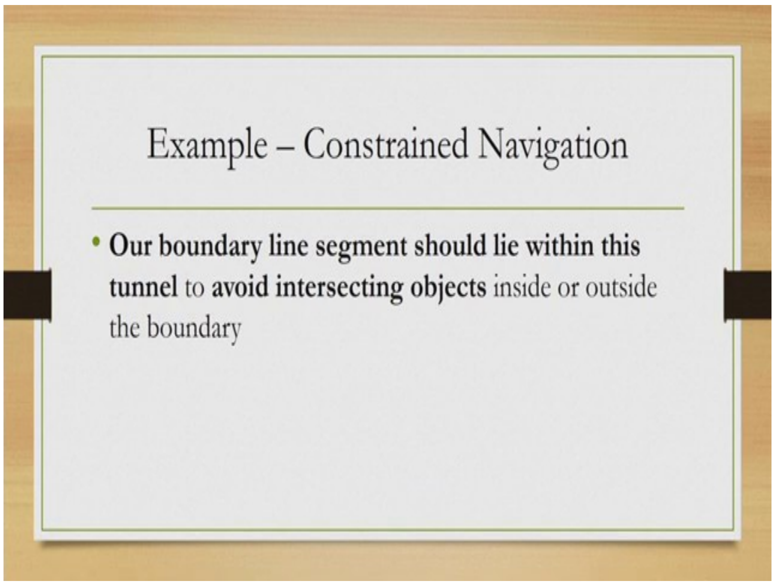


Example – Constrained Navigation

- Entire boundary line can be divided into segments
- In each segment, we can define a tunnel-like pathway having a specific width

In the given example of putting the boundary line around a group of objects what we can see is that entire line entire boundary line can be divided into segments and in each segment we can define a tunnel like pathway having a specific width. The idea is illustrated in the figure. Remember that earlier we said to ignore those dotted lines now let us concentrate on these dotted lines. These dotted lines actually indicate those tunnels. So, this is one tunnel, this is another tunnel, this one is another tunnel, here is one more tunnel, more tunnels. So, this entire red line or the boundary light passes through these tunnels.

(Refer Slide Time: 05:27)



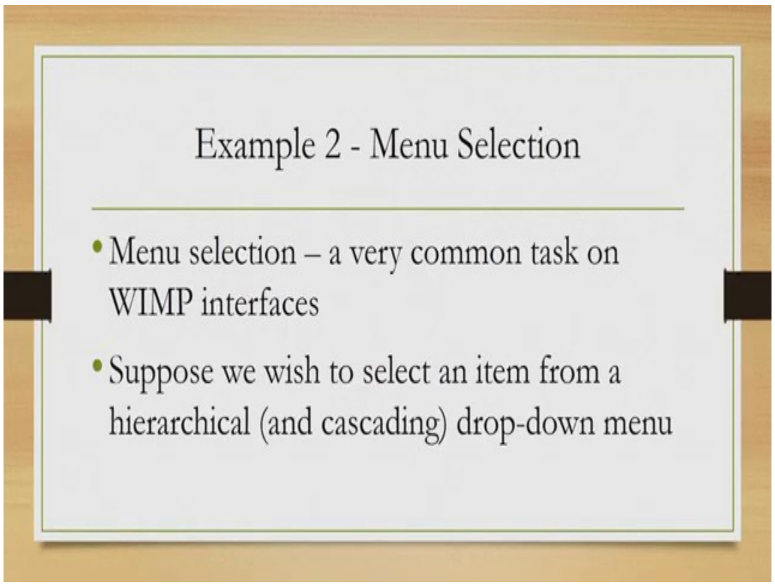
Example – Constrained Navigation

- **Our boundary line segment should lie within this tunnel** to **avoid intersecting objects** inside or outside the boundary

So, we have defined these tunnels each of these has specific width and when we have drawn the line we try to ensure that the line passes through these tunnels. Why that is important, because otherwise what will happen is that the line will cross the object boundaries and intersect those object boundaries which we do not want which we want to avoid.

So, that is the constraints in our movement behaviour that means we are constrained to move the pointer or our hand or finger in a way such that we ensure that the line that we are drawing or the trajectory that we are following remains within specified boundaries and that is the constraint. This is unlike the Fitts law modeling of pointing tasks where we do not bother about such constraints.

(Refer Slide Time: 06:11)

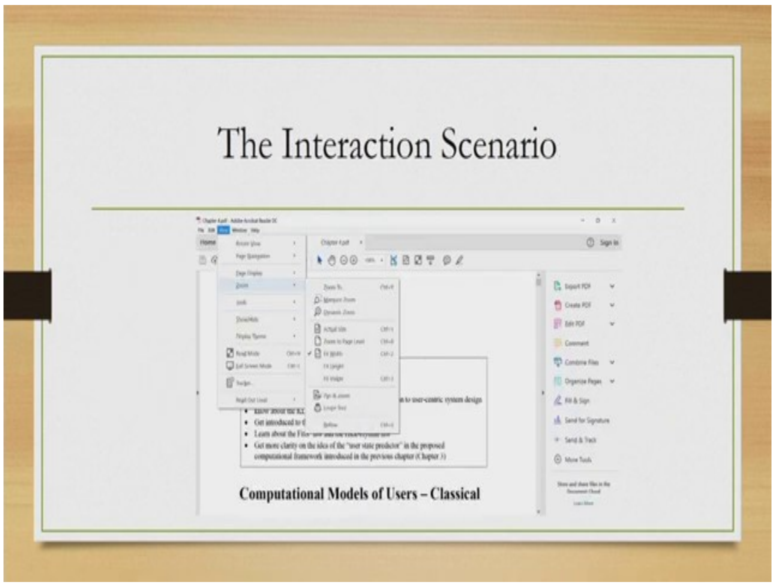


Example 2 - Menu Selection

- Menu selection – a very common task on WIMP interfaces
- Suppose we wish to select an item from a hierarchical (and cascading) drop-down menu

Now, let us see one more example. You may argue that putting such boundary lines around a group of objects may not be a very important interaction activity. So, why bother to model such activity that need not to be the case. Actually, we many a times we use such constrained movement or constrained navigation in many other more common more popular activities, one example is selection from a hierarchical menu. Now in that case we actually perform constrained navigation, let us see how consider this menu selection task.

(Refer Slide Time: 06:45)

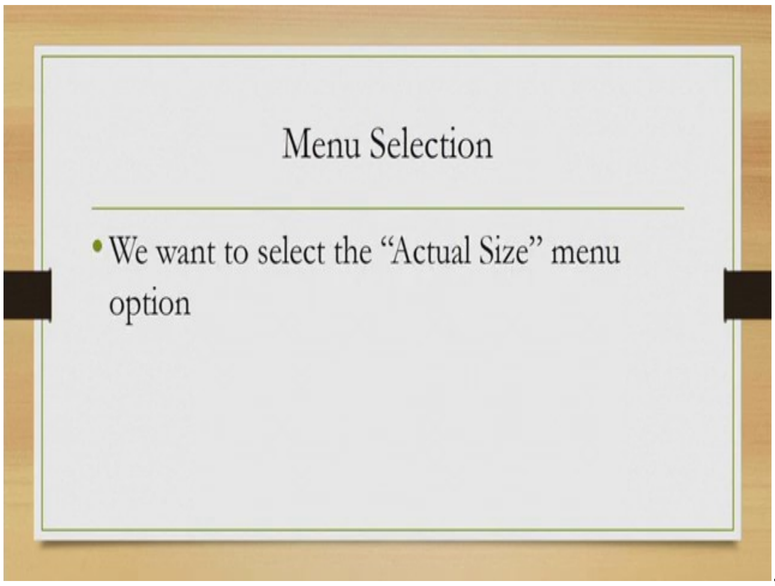


The Interaction Scenario

Computational Models of Users – Classical

So, here we have selected top level menu option here once that is selected a dropdown menu appeared. In that drop down menu, we have selected one more option then another menu appeared here and in that we want to select some option. So, this is typically what we do when we want to select a menu sub item.

(Refer Slide Time: 07:15)



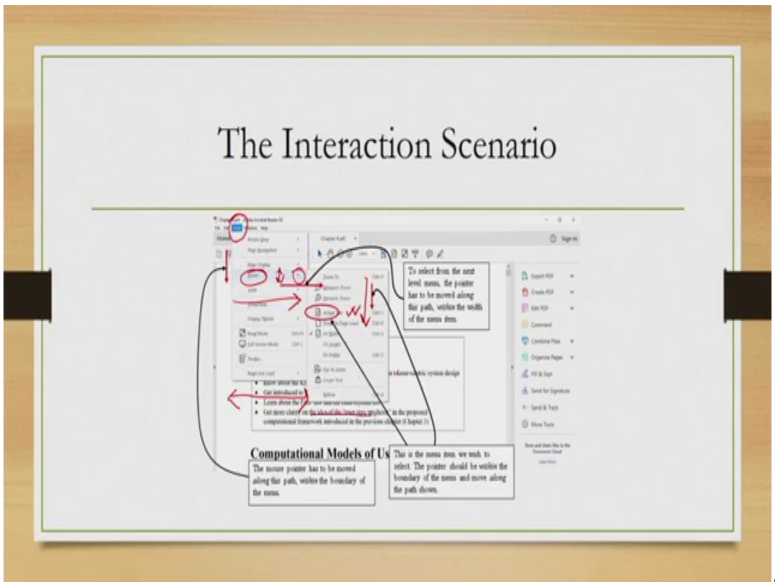
Menu Selection

- We want to select the "Actual Size" menu option

.

Now, let us assume that we want to select the actual size menu option. So, this is the final option that we want to select shown within the red boundary. And in order to do that what are the things that we need to do? Let us try to break down the overall task of selecting the actual size menu option into the sub tasks that are part of it.

(Refer Slide Time: 07:39)



The Interaction Scenario

To select from the next level menu, the pointer has to be moved along this path, within the width of the menu item.

Computational Models of Us

The mouse pointer has to be moved along this path, within the boundary of the menu.

This is the menu item we wish to select. The pointer should be within the boundary of the menu and move along the path shown.

.

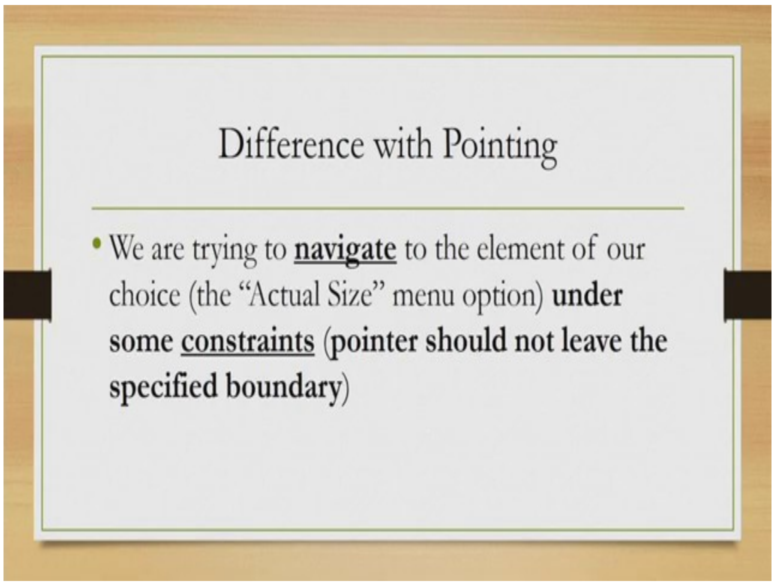
So, what we do is, essentially once we have clicked on this top -level option and the dropdown menu appeared we perform a vertical movement downwards. Here, we means that point that so, we have to answer that the pointer the or the mouse pointer moves downwards vertically along the list of sub menu items.

Now, once we have reached particular sub menu items that is the zoom item then we part from me or the mouse pointer has to be moved in a horizontal line in this way to come to this arrow symbol here. Once, it is clicked or in some cases once our pointer is on this item automatically this next level menu appears and in this next level menu we again have to make it by vertical moment of the mouse pointer to reach to the actual size menu option.

So, the tasks that we do comprises of several sub tasks. First thing is a vertical movement downwards by the cursor by the mouse pointer followed by a horizontal movement followed by another vertical movement. Now, this series of activities has to be done in a constrained manner, how? Note here that when we are moving vertically downwards we have to ensure that our mouse pointer does not exceed the boundary of this list. So, it should remain within this width.

Similarly, when we are performing this horizontal movement towards the next level menu items, we should ensure that our mouse pointer does not leave the boundary of this width of this menu item. The same thing applies in case of the second vertical movement as well. Here our mouse pointer should not cross the boundary of the second level menu item list in summary. So, when we are performing a selection task from a dropped down or any other menu organization what we have to do is basically ensure that the movement of the mouse pointer remains within the confines of the boundary of the list of sub menu items. And that is the constraint navigation task that is inherent in menu selection.

(Refer Slide Time: 10:45)

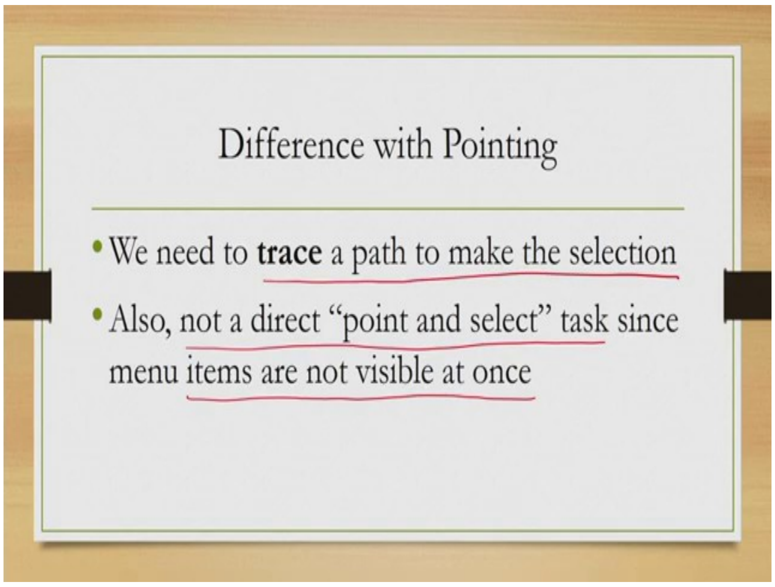


Difference with Pointing

- We are trying to **navigate** to the element of our choice (the "Actual Size" menu option) **under some constraints (pointer should not leave the specified boundary)**

So, then let us summarize the difference with pointing. So, in case of pointing we do not bother about these constraint navigation constraint path, but in case of menu selection or drawing some boundary line around one or multiple objects in a document, we have to follow a constraint path and that is one difference trace a path which is constraint.

(Refer Slide Time: 11:05)

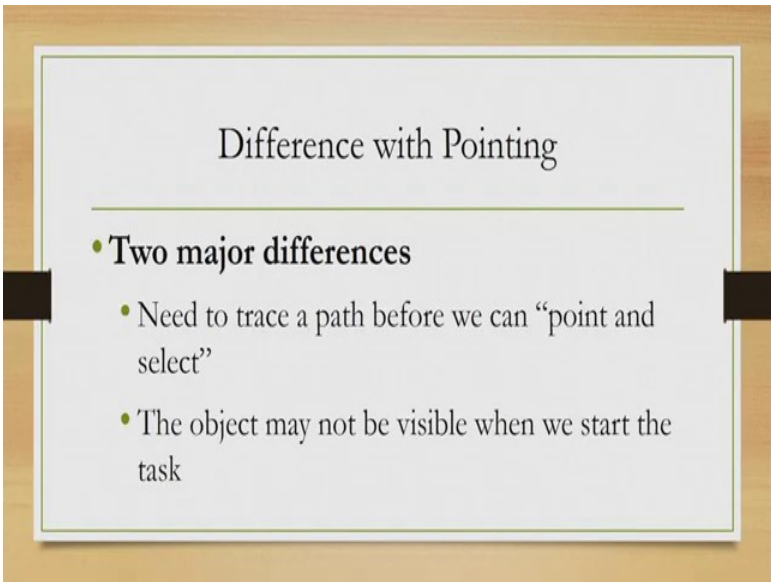


Difference with Pointing

- We need to **trace** a path to make the selection
- Also, not a direct "point and select" task since menu items are not visible at once

There is also another difference that is when we are talking of selecting a menu item, it is not that the item is already visible. In fact, during our navigation the item becomes visible as you probably have experienced during selection of a menu item from any drop down menu list. So, it is not that all the menu items are visible at once, as we keep on selecting from higher levels the lower levels keep on appearing. So, it is not a direct point and select task unlike in the behaviour model by the classical Fitts' law or the bivariate or and trivariate pointing models, but it is a task which involves items which are not visible at once.

(Refer Slide Time: 12:19)



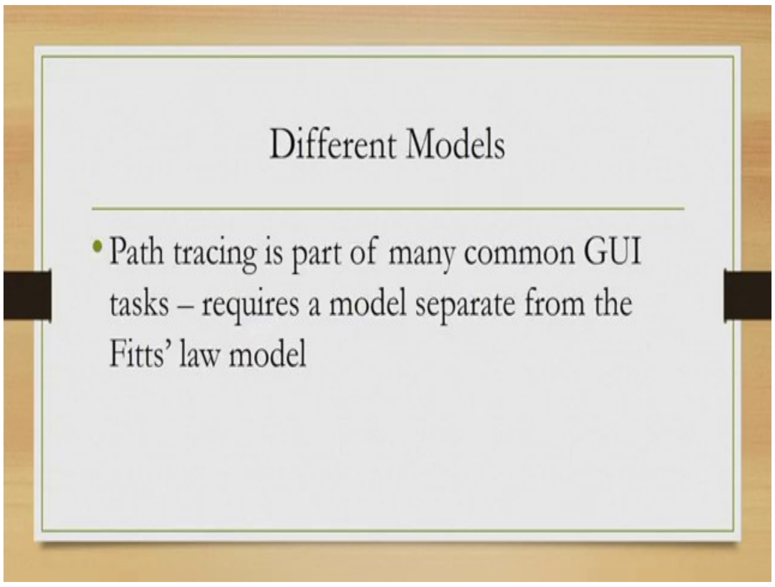
Difference with Pointing

- **Two major differences**
  - Need to trace a path before we can "point and select"
  - The object may not be visible when we start the task

So, here although we need to select point two items, but those items may not be visible at once. So, clearly in these two examples we have then seen two differences one is constraint navigation, we have to follow a path constrained by its boundary and we have to ensure that the movement of the pointer or finger or whatever we are using to point remains within that boundary. And secondly, we cannot directly point and select as the items may not be visible at once, particularly in the context of selection from drop down menus.

So, what we need is that different models, the classical Fitts' law will not be sufficient and do require models to compute the performance in the case of constraint navigation tasks.

(Refer Slide Time: 13:03)

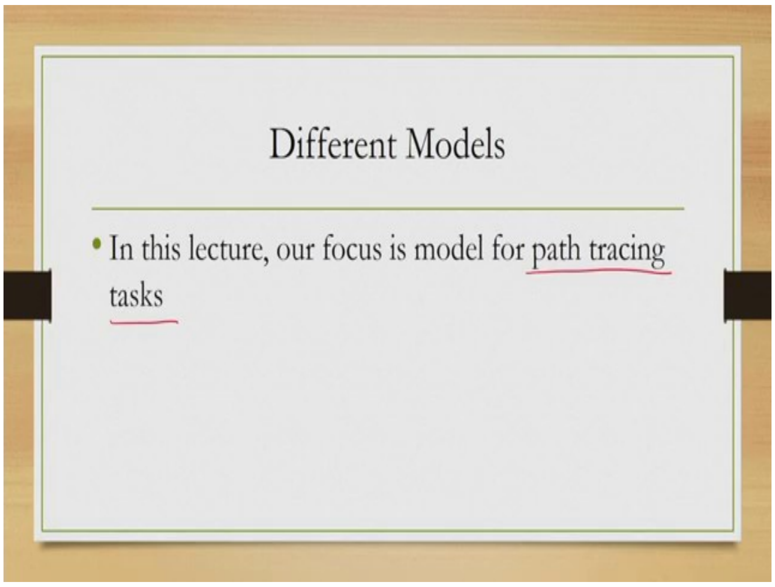


Different Models

- Path tracing is part of many common GUI tasks – requires a model separate from the Fitts' law model

Now, let us try to understand few of those models there are more than one models. And in this lecture, we will learn about one model in the next lecture there will be one more model discussed which deal with modeling performance of users during constraint navigational tasks.

(Refer Slide Time: 13:27)

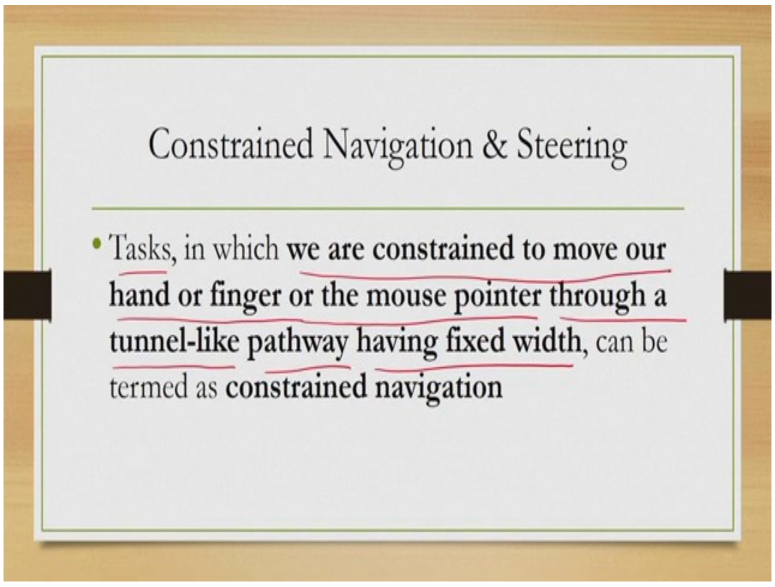


Different Models

- In this lecture, our focus is model for path tracing tasks

So, in this lecture our focus is on modeling performance during path tracing tasks. So, we need to model the performance of user while the user is performing a path tracing task like the one we have seen when we discussed about putting a boundary around a group of objects or putting a boundary around a word group of words during document creation and so on. So, we have a model to compute the performance in such tasks and we are going to discuss that model in this lecture.

(Refer Slide Time: 14:07)

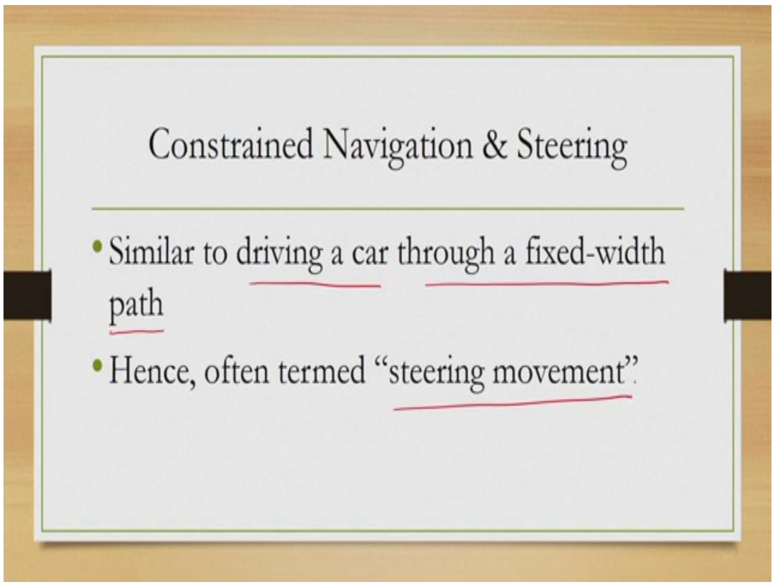


Constrained Navigation & Steering

- Tasks, in which **we are constrained to move our hand or finger or the mouse pointer through a tunnel-like pathway having fixed width**, can be termed as **constrained navigation**

So, before we start the model let us define what we mean by constrained navigation? So, earlier we have discussed it in an informal way so, here let us define it. So, constraint navigational tasks are tasks in which we are constrained to move our hand or finger or mouse pointer through a tunnel like pathway having fixed width. If these conditions are met then we can say that we are dealing with a constrained navigational task.

(Refer Slide Time: 14:49)

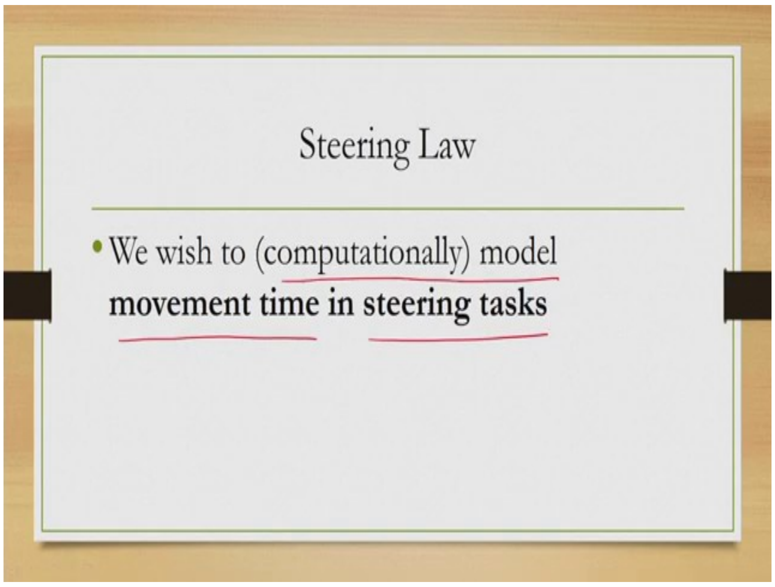


Constrained Navigation & Steering

- Similar to driving a car through a fixed-width path
- Hence, often termed "steering movement"

Now, these tasks can be equated with the problem of driving a car through a fixed- width path. So, when we drive a car through a fixed width path, we get to see similar behaviour. And accordingly this constraint navigates and is often referred to as steering movement. So, whatever we are discussing now is essentially the kind of augment which is called or popularly referred to often as steering movement and in the context of interactive systems, we are looking to develop or we are looking forward to have a model to compute the performance of such steering movements.

(Refer Slide Time: 15:41)

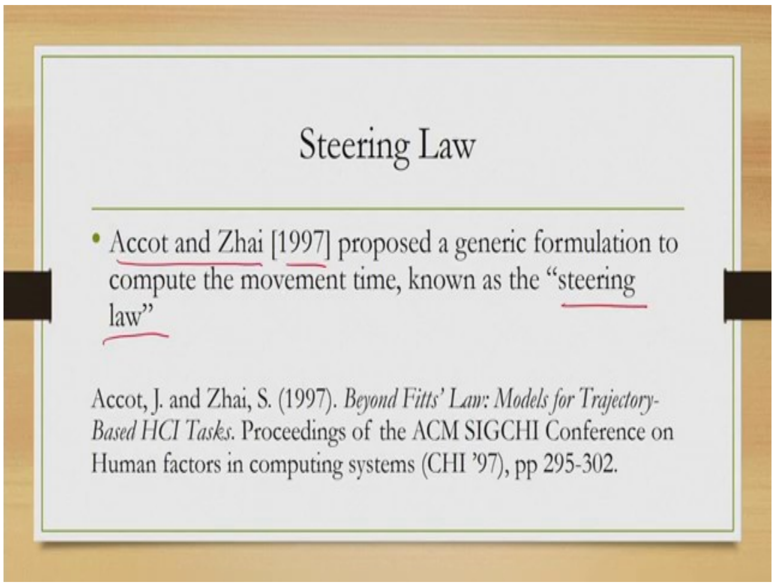


Steering Law

- We wish to (computationally) model **movement time in steering tasks**

So, our objective is to computationally model the movement time in Steering tasks. So, that is our overall objective to computationally model movement time in steering tasks. And there is one law proposed by Accot and Zhai in 1997 which helps us in modeling the steering movement time, this is called the steering law, ok.

(Refer Slide Time: 15:57)



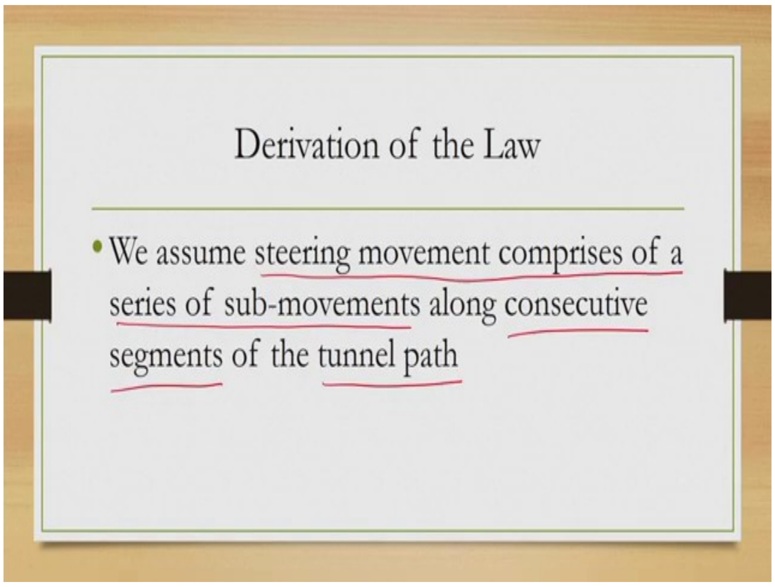
Steering Law

- Accot and Zhai [1997] proposed a generic formulation to compute the movement time, known as the "steering law"

Accot, J. and Zhai, S. (1997). *Beyond Fitts' Law: Models for Trajectory-Based HCI Tasks*. Proceedings of the ACM SIGCHI Conference on Human factors in computing systems (CHI '97), pp 295-302.

The law was published in this paper, you can refer to it if you are interested beyond Fitts' law models for trajectory based HCI tasks, it was published in ACM SIGCHI Conference 97.

(Refer Slide Time: 16:29)



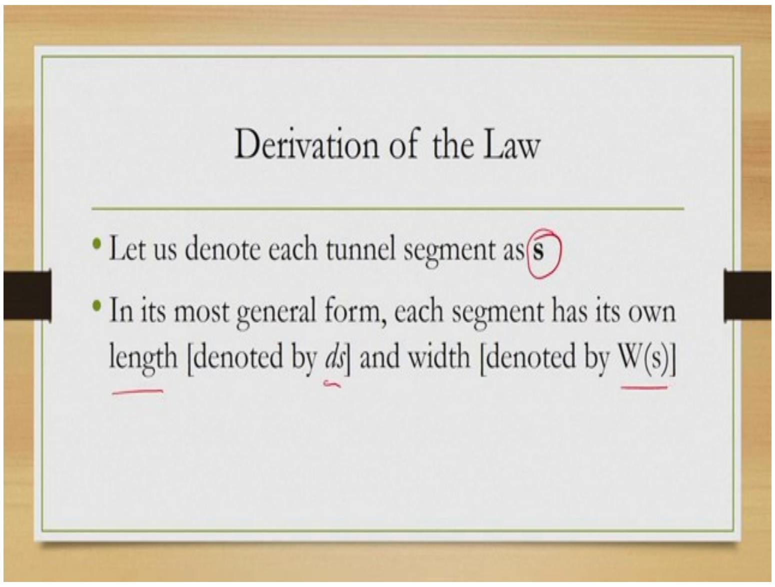
Derivation of the Law

- We assume steering movement comprises of a series of sub-movements along consecutive segments of the tunnel path

So, what the steering law assumes? While we are trying to model steering movement performance, steering movement behaviour, what we are assuming is that the steering movement comprises of a series of sub movements along consecutive segments of the tunnel path.

So, what we are assuming is that the steering movement is performed along a tunnel path and we can divide this entire movement into a series of sub movements which are consecutive and there are consequentive segments of tunnels through which these sub movements take place. So, there are segments in the tunnel.

(Refer Slide Time: 17:17)

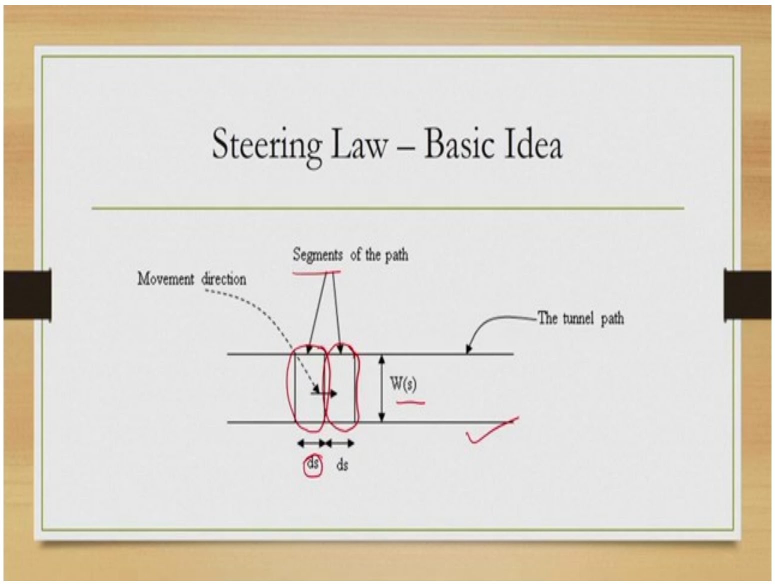


Derivation of the Law

- Let us denote each tunnel segment as s
- In its most general form, each segment has its own length [denoted by *ds*] and width [denoted by W(s)]

So, we are assuming that the tunnel can be divided into a series of segments. Let us do not each segment by s now in its most general form. So, each segment it has its own length denoted by d s and width denoted by Ws let us assume these terminologies.

(Refer Slide Time: 17:41)



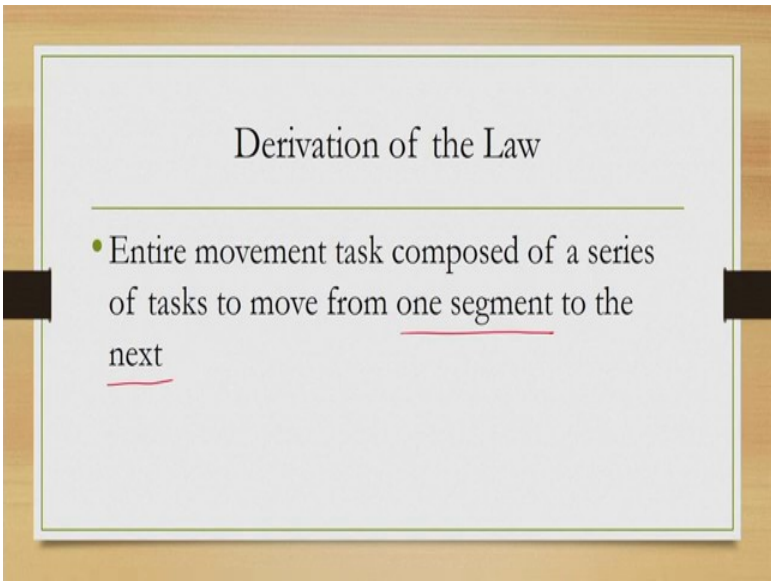
Steering Law – Basic Idea

Then we can actually depict the entire scenario of tunnel movement during constant navigation with this illustration. Here as you can see, this is the tunnel now in this tunnel, we have divided it into sub segments each of these sub segments. So, this is one sub segment this is another sub segment represented by this boundary lines each of these sub segment has a length ds and width Ws.

Now, although in this picture it may appear that Ws is same. So, it is not variable, but in general the tunnel width may be different for each sub segment that is the most general formulation of the problem. So, to recollect we are assuming that the entire tunnel path through which the constraint navigation takes place can be divided into segments. Each segment has a width and these widths need not be the same, we can represent it with Ws and that length of the tunnel path can be represented by ds.

So, if that is the case then we can actually model the performance in terms of these variables ds Ws, let us see how? So, when we are assuming that the path can be divided into sub path or segments what that assumption implies?

(Refer Slide Time: 19:19)

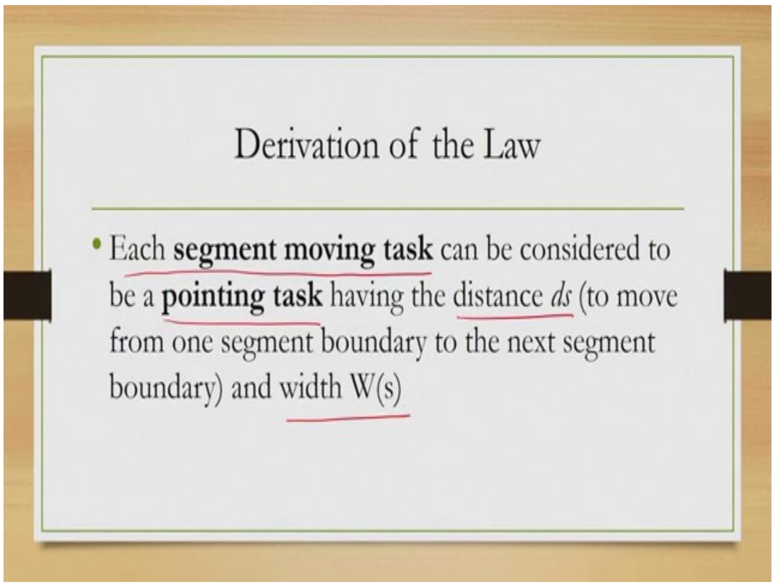


Derivation of the Law

- Entire movement task composed of a series of tasks to move from one segment to the next

It implies that entire movement task comprises of series of sub tasks from one segment to the next. So, this is important. So, when I say series of subtasks I mean sub tasks that requests the pointer to move from one segment to the next. Now, this is nothing, but simple movement task.

(Refer Slide Time: 19:41)

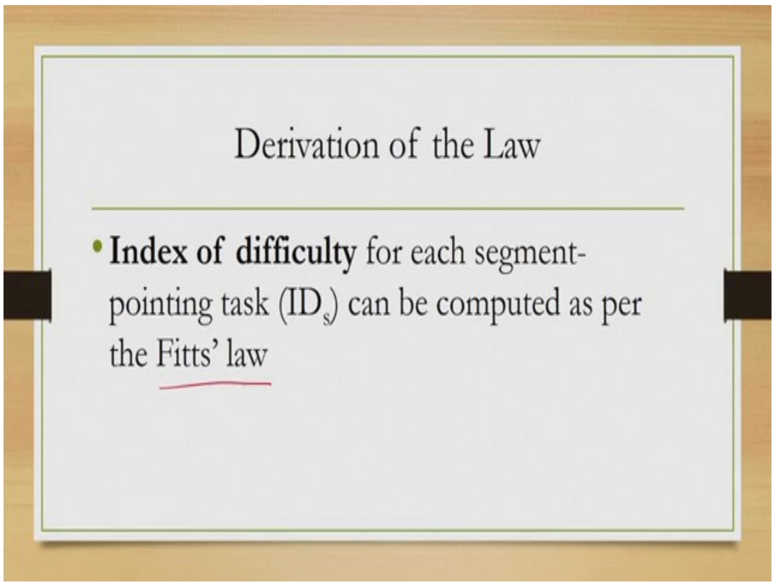


Derivation of the Law

- Each **segment moving task** can be considered to be a **pointing task** having the distance *ds* (to move from one segment boundary to the next segment boundary) and width W(s)

In that case we can consider each segment moving tasks that is movement from one segment into the next as a pointing task where the distance is ds and the width of the target is Ws then you may be thinking that we can use the Fitts' law to model these tasks.

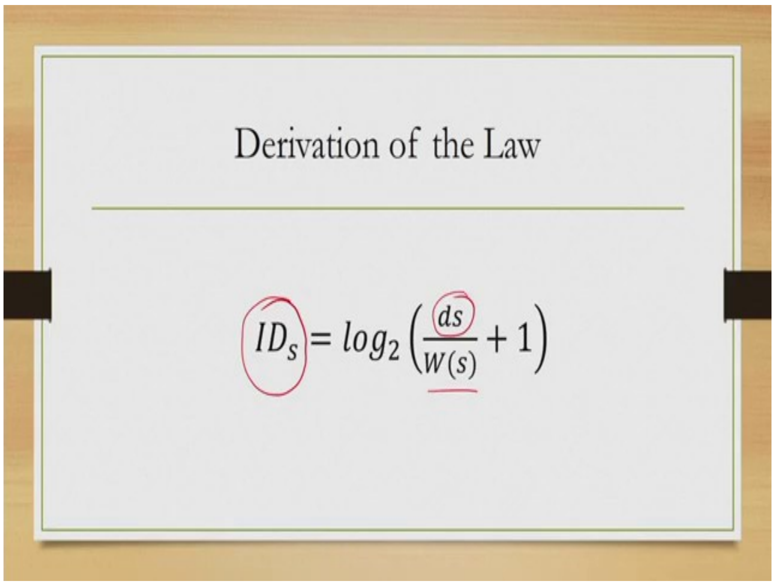
(Refer Slide Time: 20:09)



Derivation of the Law

- **Index of difficulty** for each segment-pointing task (ID_s) can be computed as per the Fitts' law

(Refer Slide Time: 20:21)

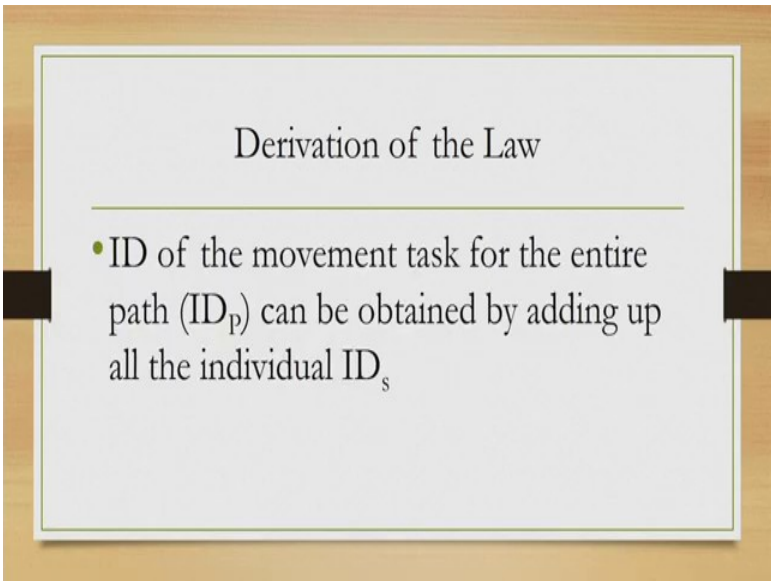


Derivation of the Law

$$ID_s = log_2\left(\frac{ds}{W(s)} + 1\right)$$

So, in that case they index of difficulty for each segment Pointing task can be computed as per the Fitts' law as may seem obvious and it will look like this mathematical expression where IDs is the index of difficulty for each segment moving task is represented by the classical form of the Fitts' law where Ws is the width of the segment ds is the distance between the segments. We can think of this distance as the distance between the midpoints of the segments to give it a generalized interpretation.

(Refer Slide Time: 20:51)

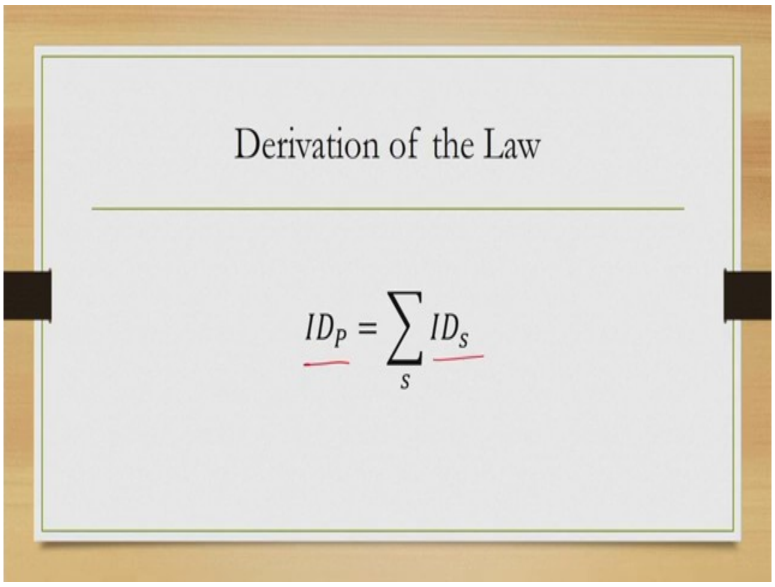


Derivation of the Law

- ID of the movement task for the entire path (ID_P) can be obtained by adding up all the individual ID_s

Then once we know the ID of individual segment movement tasks then we can sum up all the IDs for all the segments along the entire tunnel path to get the overall index of difficulty and so that we can get with the use of a summation of individual IDs as shown in this expression.

(Refer Slide Time: 21:11)

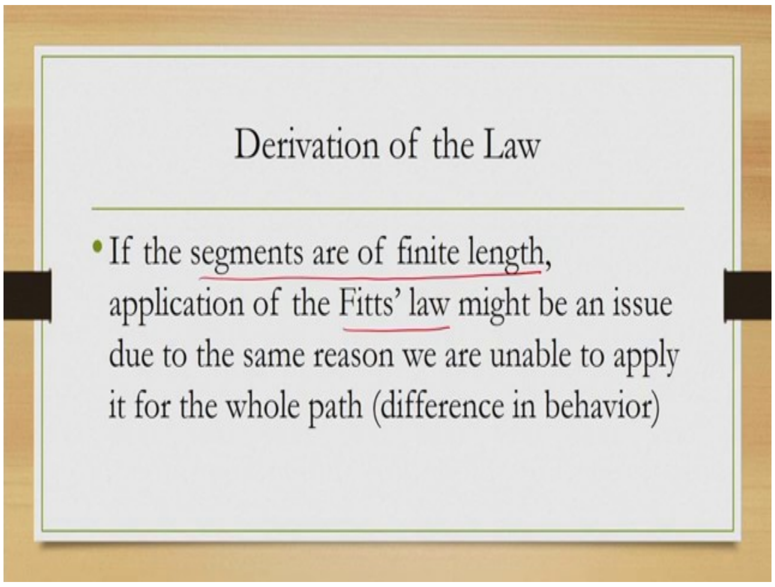


Derivation of the Law

$$ID_P = \sum_{s} ID_s$$

Here IDp is the index of difficulty for the entire path which is a summation of individual segment moving task IDs and it is summed over all the segments that are there in the tunnel, but the problem is that if the segments are of finite length then we cannot directly apply Fitts' law because of the same reason we are unable to apply it for the entire path.

(Refer Slide Time: 21:29)

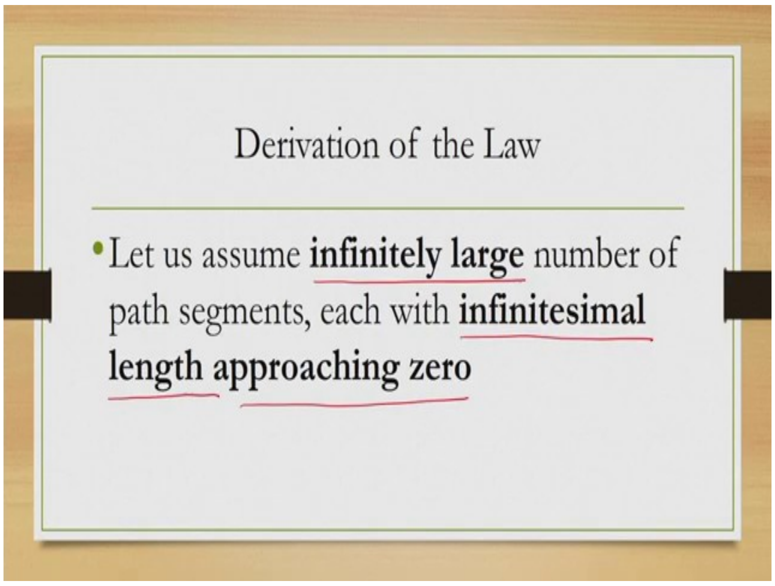


Derivation of the Law

- If the segments are of finite length, application of the Fitts' law might be an issue due to the same reason we are unable to apply it for the whole path (difference in behavior)

So, if we can apply Fitts' law for finite length tunnel segments then it should be applicable for the entire path. So, we do not need to divide it in segments. So, from the start position to the end position we can simply apply the Fitts' law; that is not possible because our behavior is different here, we are behaving in a constrained manner and constrained manner violates the assumptions behind the Fitts' law formulation that is rapid aimed and error free.

(Refer Slide Time: 22:21)

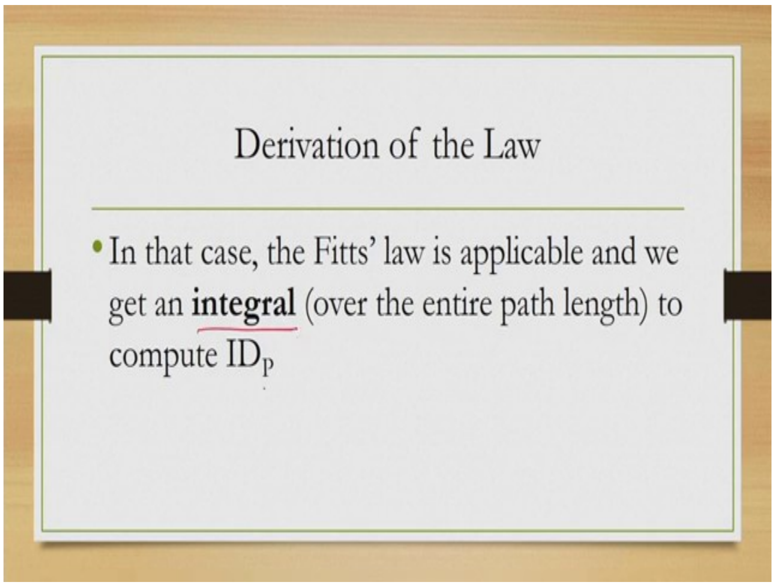


Derivation of the Law

- Let us assume **infinitely large** number of path segments, each with **infinitesimal length approaching zero**

So, we will not be able to use the earlier formulation because we are assuming that segment lengths are finite. And if that is the case then Fitts' law application is not possible. Instead what we can do is we can assume that these segments are infinitesimal length. So, we can assume that each segment has infinitesimal length approaching zero. Consequently, we are assuming that the entire tunnel path is having infinitely large number of such segments. So, each segment is of infinitesimal length and the tunnel has infinitely large number of such extremely small segments.

(Refer Slide Time: 23:05)

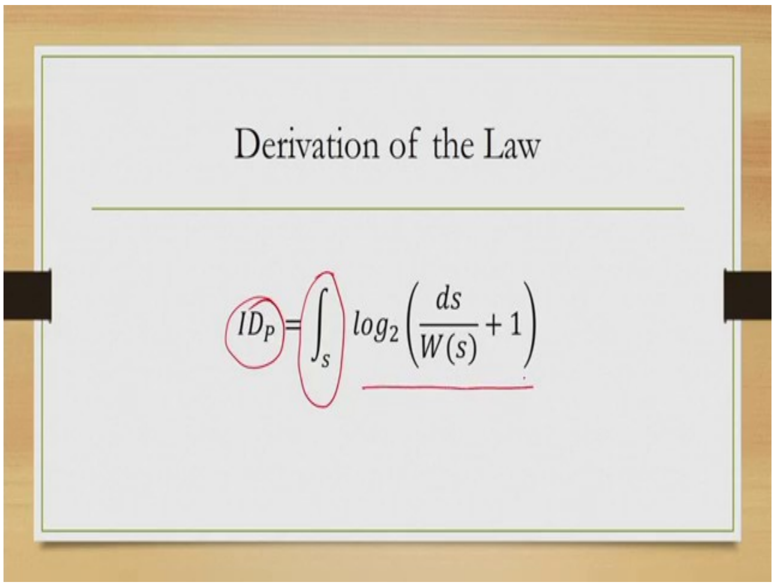


Derivation of the Law

- In that case, the Fitts' law is applicable and we get an **integral** (over the entire path length) to compute ID_P

In that case, since we are assuming very small segments. So, this constraint navigation is no longer applicable and we can apply the Fitts' law to model movement from one segment to another; however, that model will not have summation notation, instead if you may recollect from your knowledge of basic calculus when we are dealing with infinitesimally small segments or infinitesimally small numbers we go for integration over the entire range of such numbers. In this case we get an integral over the entire path to compute the index of difficulty for the entire path.

(Refer Slide Time: 23:51)

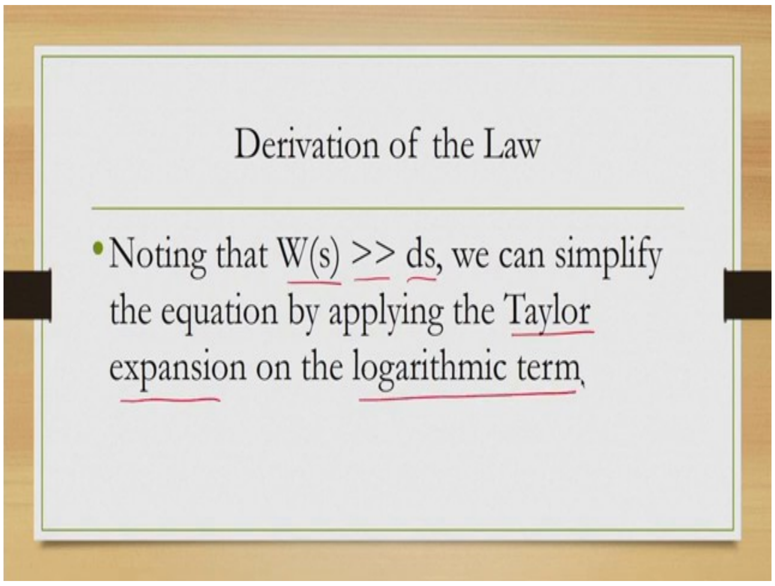


Derivation of the Law

$$ID_P = \int_s \underline{\log_2\left(\frac{ds}{W(s)} + 1\right)}$$

The formulism looks something like this equation where the IDp has the same meaning that is the index of difficulty for the entire path, but instead of summation now we have an integration notation over the entire path and then we have this expression which is similar as before.

(Refer Slide Time: 24:13)

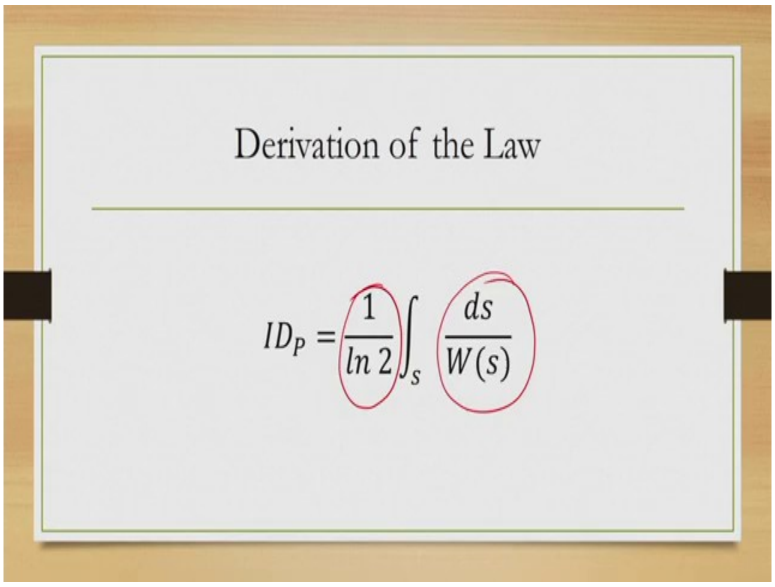


Derivation of the Law

- Noting that W(s) >> ds, we can simplify the equation by applying the Taylor expansion on the logarithmic term,

Now, if we make one simplifying assumption that is the width of each segment Ws is very very or much much greater than the distance between the segments or distance between the center points of two consecutive segments which is anyway likely to be the case, because here we are dealing with infinitesimally small segments which may be having large width then we can simplify the previous formulation by applying the Taylors expansion on the logarithmic term.

(Refer Slide Time: 24:53)

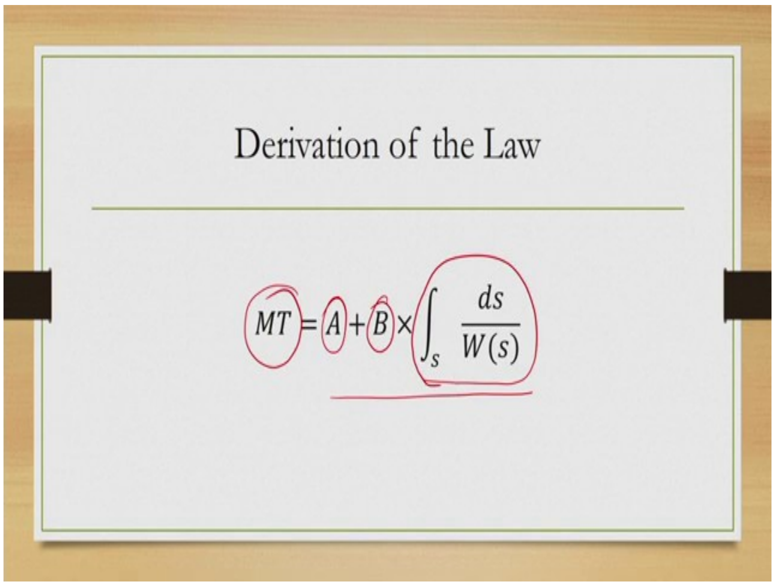


Derivation of the Law

$$ID_P = \frac{1}{ln\,2}\int_S \frac{ds}{W(s)}$$

So, if we perform that what well get is this expression where we have taken out this logarithmic term and the term is now simplified in this form where only this ratio is there. So, this is the index of difficulty for the movement during a tunnel path or this is the index of difficulty for modeling the movement during constraint navigational tasks along a tunnel path. So, we can use this index of difficulty in regular formulation of movement time as we have seen in the case of the application of the Fitts’ law.

(Refer Slide Time: 25:33)



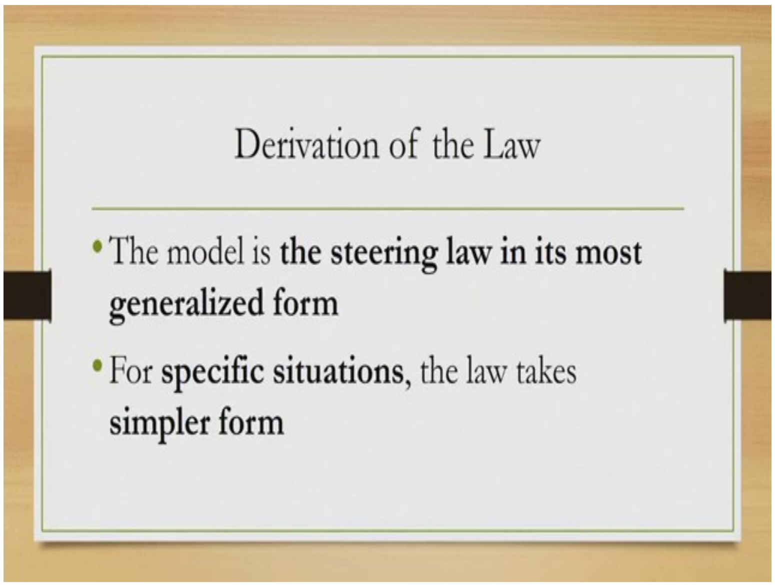
Derivation of the Law

$$MT = A + B \times \underline{\int_S \frac{ds}{W(s)}}$$

So, we can simply have this expression for computing the total movement time to perform the constraint navigational tasks as in this manner where A and B are constraints and this term is the index of difficulty as we have seen just before.

Now, note that here the B includes the constant term after Taylors expansion as well as some other constants. So, by the notion B, we are referring to both the constants together. Now, this model of movement time is the steering law model in it is most general form; however, this need not to be the case always.

(Refer Slide Time: 26:19)

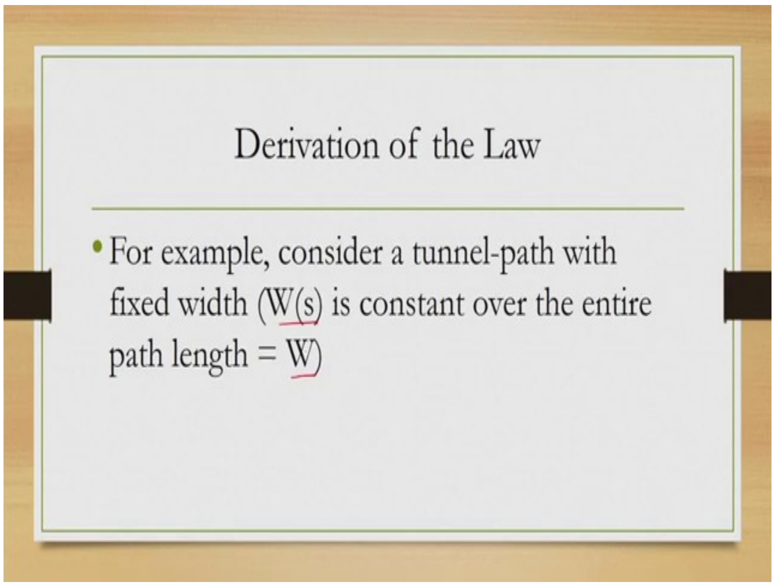


Derivation of the Law

- The model is **the steering law in its most generalized form**
- For **specific situations**, the law takes **simpler form**

In specific situations we may have a simpler form.

(Refer Slide Time: 26:23)

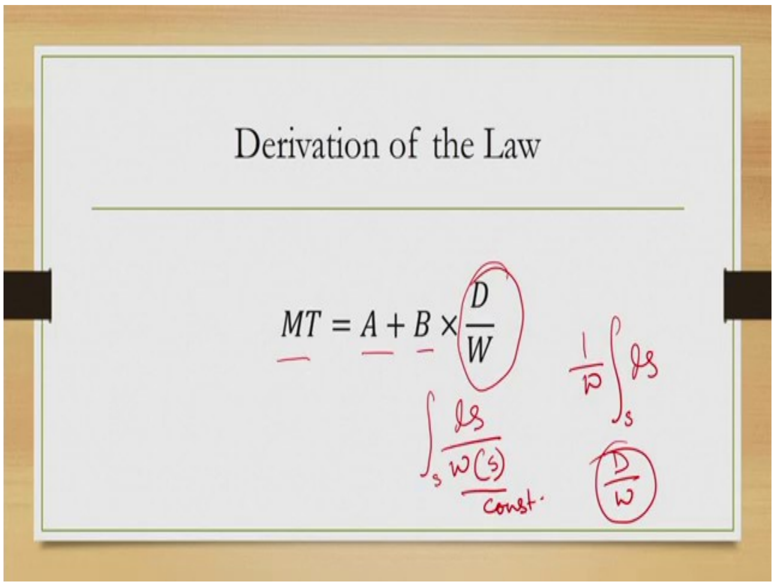


Derivation of the Law

- For example, consider a tunnel-path with fixed width (W(s) is constant over the entire path length = W)

For example assume that the tunnel width for each segment remains constant. So, each segment has a width and these widths are same for all the segments. So, in that case we have a constant tunnel width W.

(Refer Slide Time: 26:45)



Derivation of the Law

$$MT = A + B \times \frac{D}{W}$$

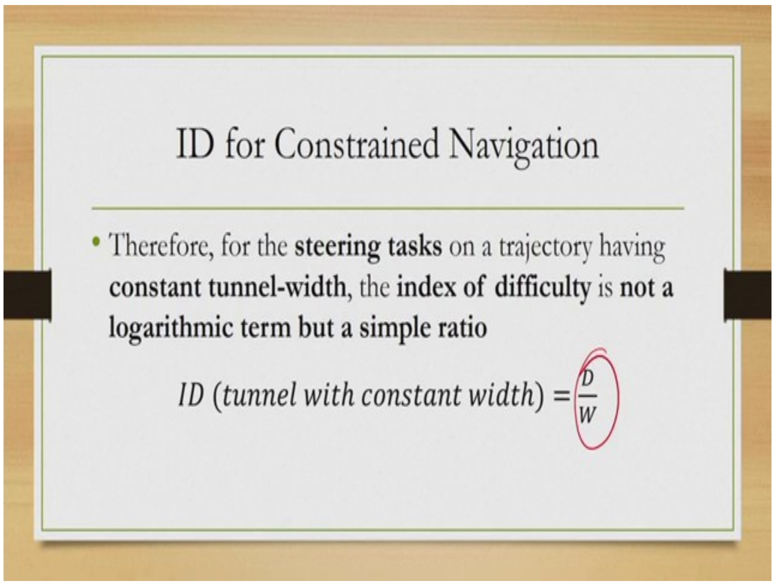
$$\int_s \frac{ds}{\underbrace{W(s)}_{const.}} \qquad \frac{1}{W}\int_s ds \qquad \frac{D}{W}$$

So, we can replace Ws with W and we may get a much simplified form which is given in this way

MT = A + B*(D/W)

So, here we can actually derive it informally that is if we have an integration over s, ds by W(s) and W is constant then this term goes out. So, we get 1/W this is followed by s integration over s ds which is nothing but the total tunnel length d. So, we get D/W as the index of difficulty for tunnel path having constant width.

(Refer Slide Time: 27:31)



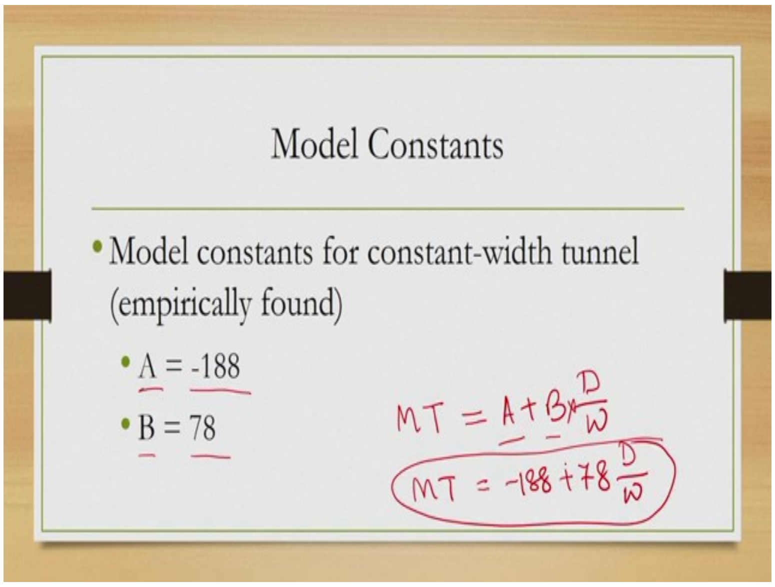
ID for Constrained Navigation

- Therefore, for the **steering tasks** on a trajectory having **constant tunnel-width,** the **index of difficulty** is **not a logarithmic term but a simple ratio**

$$ID\ (tunnel\ with\ constant\ width) = \frac{D}{W}$$

So, for tunnel path with constant width we have an index of difficulty D/W as we have just derived before.

(Refer Slide Time: 27:39)



Model Constants

- Model constants for constant-width tunnel (empirically found)
  - $\underline{A} = \underline{-188}$
  - $\underline{B} = \underline{78}$

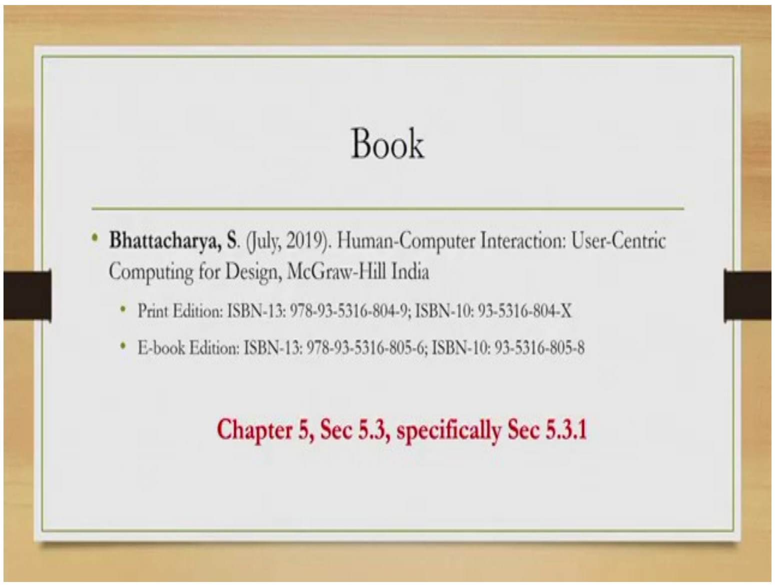
$$MT = \underline{A} + \underline{B} \times \frac{D}{W}$$

$$MT = -188 + 78\frac{D}{W}$$

Now, the model says that for computing movement time, we have to use the equation A plus B into D by W. Now, A and B as I said are constants. So, we need to estimate the constant values. Now, in literature these constant values are reported which were found after extensive empirical studies and these values are A equal to minus 188 and B equal to 78. So, you can simply replace these values and get a movement time model for constraint navigation along a tunnel path having [vocalized- noise] constant width as something like this. So, this is our final model for tunnel paths having constant width.

So, this is about modeling performance during movement of mouse pointer during constraint navigation along a tunnel path. So, whatever we have discussed so far only models the performance of a constraint navigation movement task. We have not discussed anything about menu selection which we are going to discuss in the next lecture.

(Refer Slide Time: 29:15)



Book

- **Bhattacharya, S.** (July, 2019). Human-Computer Interaction: User-Centric Computing for Design, McGraw-Hill India
  - Print Edition: ISBN-13: 978-93-5316-804-9; ISBN-10: 93-5316-804-X
  - E-book Edition: ISBN-13: 978-93-5316-805-6; ISBN-10: 93-5316-805-8

Chapter 5, Sec 5.3, specifically Sec 5.3.1

So, whatever we have discussed in today’s lecture can be found in this book and the interested reader is advised to refer to chapter 5 section 5.3 specifically section 5.3.1 to learn more about the totop.

Thank you and goodbye.