GestTB: Gestural Command Selection on Touch Bar

Arihant Parsoya
Indian Institute of Technology Bombay
Mumbai, India
parsoyaarihant@gmail.com

Venkatesh Rajamanickam
Indian Institute of Technology Bombay
Mumbai, India
venkatra@iitb.ac.in

Abstract
We present GestTB, a gestural interaction technique for command selection on the Apple MacBook touch bar (TB). GestTB dramatically expands the number of commands that can be selected from the application menu by displaying the menu on the TB. Users can select a command by performing a swipe gesture of varying swipe lengths along the length of the TB, to invoke a range of commands. GestTB also supports hierarchical commands using multi-direction gestures in horizontal direction. We present a user study that provides the design parameters for interaction using GestTB.

Author Keywords
gestural interaction; command selection; hierarchical gestures; touch bar

CCS Concepts
•**Human-centered computing** → **Gestural input;** *Touch screens;* Graphics input devices; Please use the 2012 Classifiers and see this link to embed them in the text: https://dl.acm.org/ccs/ccs_flat.cfm

Introduction
Touch bar (TB) is a touch sensitive display at the top of the keyboard. It allows the user to select commands by displaying buttons which can be customized by the user, and

change based on the application being used. Displaying commands on the TB removes the need for memorization of hotkeys (or keyboard shortcuts) to select commands, and provides quick access to commands by being placed on top of the TB. While the TB improves over hotkeys, only 13 buttons can be displayed on TB [2], which severely limits the number of commands which can be selected using TB.

In this paper, we introduce GestTB, a gestural interaction technique to select commands using the TB. GestTB provides access to the commands in the application menu by displaying it on the TB as a horizontal grid, where the user can select the command using gestural interaction. GestTB also provides support for hierarchical gestures to increase the number of commands users can select. At the first level of hierarchy, our technique supports 72 commands, which can increase exponentially with the increase in the level of hierarchy (572 in second level of hierarchy). Once the user learns the gesture for a particular command, he can perform the gesture without looking at the TB, supporting both novice and expert interactions. In this paper, we also present the design space for interaction with TB, and a user study to finalize the design parameters for GestTB.

Related Work

Hotkeys [10] is a popular method for command selection by pressing a combination of keys on the keyboard. However this technique has been underutilized, which could be a result of the complex key modifier combination [10]. Multiple techniques have been proposed to promote hotkey usage. IconHK [6] helps the user to memorize the keyboard shortcuts by displaying the keyboard shortcuts when the user clicks on the icon inside the application, HotStrokes [4] technique aids the user of hotkeys by displaying the hotkey combination as the command command is selected using gestural interaction on the trackpad.

Arranging the commands spatially leverages users spatial memory in which aids in memorization [12]. Command Maps [14] flattens the command hierarchy for quick access after the user has memorized the spatial location of the command. FastTap [7] arranges the commands in a rectangular grid, which can be accessed using a two finger tap.

Gestural Interactions

Gestural interaction techniques for command selection also leverage the use of spatial memory by mapping gestures to commands. Marking menus [9] displays hierarchical commands in a radial fashion which can be accessed using gestural interaction, Augmented Letters [13] builds on marking menus by filtering the commands based on the initials of the command name. Bezel-Tap [16] selects commands using gestural interaction from the bezel of the tablet device. Markpad [5] technique builds on Bezel-tap mapping the commands to the gestures which start from the border of the touchpad to and ends inside the the touchpad. Markpad also uses tactile cues on the touchpad, which improves the users performance while performing the gesture. HotStrokes [4] technique select the command by gesture typing the command name on the trackpad. GestKeyboard [18] recognizes discrete gestures performed on ordinary physical keyboard which can be used to activate commands.

Gestural command selection techniques have also been used in other interfaces like as smartwatches to overcome the restrictions of small displays. PageFlip [8] technique divides the watch display into three chords of a circle centered at the corner of the display. To select a command, the user performs a gesture from the corner to a chord. Depending on the chord and the length of the gesture, a particular command is selected.

Design Space

TB is designed to be an extension of the keyboard hence its width is designed to match the width of the MacBook keyboard. The exact dimension of TB is 1004pt in width and 30pt in height. TB display is primarily divided into two areas [3]: App Region and Control Strip. App Region is used to display the controls related to the application being used. The Control Strip is used for performing system level tasks such as changing volume and brightness. The length of the Control Strip can be varied depending on the controls the user adds into the control strip, and the maximum width of the Control Strip is restricted to 304pt width.

Gestural Interaction. Due to the low height of TB, gestures can be performed comfortably in two horizontal directions: left (L) and right (R). A combination of L and R gestures (example $R \Rightarrow L \Rightarrow R$) can also be used to expand the design possibilities of gestures in horizontal directions. Crossing gestural interaction can also be performed in horizontal directions. The existing implementation of NSTouch API [1] in MacOS sets the vertical coordinate of touch to the midpoint of the TB, ignoring any information about the users touch in vertical direction. Hence, the recognition of gestures in vertical directions is not possible.

Interaction Position. The function of interaction can be a function of where the interaction was performed on the TB. For example, the action of a gesture on TB can be a function of where the gesture began. Position independent interactions are also possible where the functionality of the TB is independent of the position user interacts with it.

Multi-touch Input. TB has built-in support for multi touch interaction. Due to the low height of TB, performing multi-touch gestures from same hand (like pinch) can be cumbersome to perform [3]. However, touch input from fingers of different hands can be combined to create interactions like FastTap [7] for command selection.

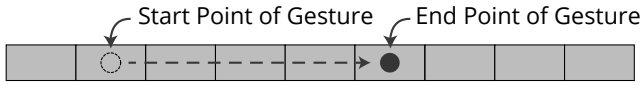

Figure 1: A unidirectional gesture on TB.

Multi-modal Interactions. The interactions of touchbar can be combined with other inputs such as keyboard or trackpad. The modifier keys in the keyboard can allow switching between different modes of the touchbar functions. For example, the function of a gesture on touchbar can change when the user presses a modifier key on keyboard.

GestTB Technique

GestTB is built for horizontal gestures on the TB. We divide the TB into smaller areas (or buttons) of equal widths, which corresponds to menus. The areas are displayed on the TB while the user is using the application. The user starts the gesture from one of the areas to access the commands in that menu, and ends the gesture in another area (Figure 1). As the user starts the gesture on a particular area, the commands corresponding to that menu area are displayed on the TB. The user can drag his fingers across the length of TB and end the gesture on the command he wants to select. The commands displayed on TB are mapped to the existing menu of the application being used. Since this interaction leverages spatial memory, this should also favor memorization [15, 12]. Once the user memorizes a gesture by repeatedly selecting a command from TB, he can perform the gesture without actively looking at the TB.

Multi Directional Gestures

When using single directional gestures (left or right gestures), limited number of commands can be mapped to the

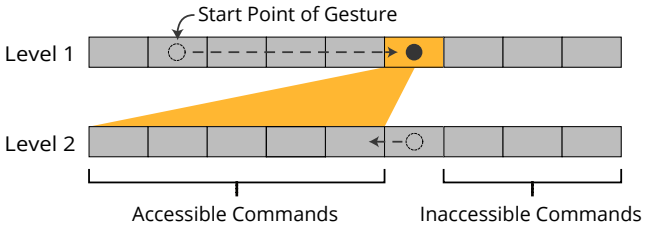

Figure 2: Accessible and inaccessible commands for multi directional gestures.

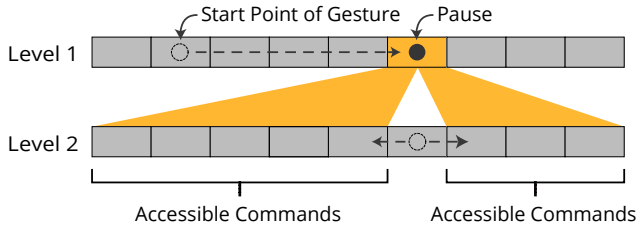

Figure 3: Accessible commands for multi directional gestures.

gestures. To increase the number of supported commands and support the use of hierarchical menus, we investigated how combination of left and right can be used on TB. Multi directional gestures can enable the support of hierarchical commands which exponentially increases the number of commands, which can be mapped to the gestures.

However, while using multi directional gestures in horizontal direction, the number of possible hierarchical command selections decreases. To illustrate this, consider the example depicted in Figure 2. When the user begins the gesture, all the commands in both left and right of the starting point are accessible to him. As the direction is changed, the command pallets is updated to the next level of hierarchy. However, due to change in direction, only the commands in the new direction (left) will be accessible to him due to direction constraints. Hence, to make the commands in previous directions also accessible, we decided to use temporal aspect of the gesture to navigate through the hierarchy. When the user wants to access commands in the old direction, he can pause for an instance at a selection area to navigate to the next level of hierarchy (Figure 3). With the support of multi directional gestures, the total, number of commands which can be mapped to these gesture combinations increase exponentially. However, as the hierarchy is increased, we expect that the time taken to select the command will increase and the memorizability of gestures will decrease.

While designing GestTB, we found that design parameters such as area size need to be found in order to determine the optimal size of area (Figure 1) for GestTB. Previous studies on touch target size [11, 17] were focused on finding the target size on the mobile devices where the target position is visible to the user and readily accessible by the hand. Unlike previous studies, TB is located above the keyboard which requires the user to re-position his hand frequently to interact with the TB. And, in most scenarios, the TB will be located in the peripheral vision of the user instead of the central vision, which might effect the accuracy of touch on the TB. Hence, to determine the size of touch area which the user can precisely locate without looking at the TB.

Experiment: Area Size Study

We determine the optimal area size for GestTB as the user interacts with the TB. The aim of the experiment is to determine how accurately the user can spatially locate (by touching) an element assuming that he remembers the spatial location of the element. We used area sizes of three different

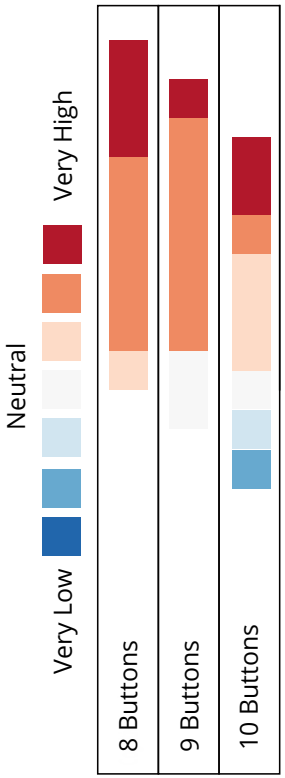

Figure 4: Users ratings on their success for different button configuration

configuration: 8, 9, and 10 buttons (corresponding to 125pt, 110pt and 100pt area widths). The area sizes were counterbalanced across the participants. To replicate this condition, we created a GUI software which displayed a graphical representation of the TB on the top of laptop screen and the TB. The task was to press the button which is highlighted on the screen. The user had to press the corresponding button on the TB. For ever task, the position of the target button was randomly generated. During the pretest, we found that users were trying to vertically align the button on the screen and the TB. Hence, we scaled down the size of the TB on the screen to prevent users from vertically aligning the buttons.

Procedure. We asked 9 participants (5 female and 4 male, one left handed, age from 22 to 29) to participate in the study. The whole experiment lasted 30 minutes. First, the participants were familiarised with the TB and asked to explore the default TB functionalities. Next, to understand the effect of learning, the experiment was conducted in two phases: (1) TRAINING phase and (2) TESTING phase. In TRAINING phase, the user was shown the target position and the position he actually touched on the TB, thus showing him the error in his task. In TESTING phase, the user was not shown any feedback about the position of his touch. The first phase was a TESTING phase followed by a TRAINING and a TESTING phases. In every phase the participants executed 50 tasks. The target position was generated using a uniform random generator along the complete length of the TB.

Apparatus. We used a 16 inch MacBook Pro with 2.2 GHz Intel Core i7 processor. The size of the TB was 1005x30 points. A Mac OS application was built using Cocoa, which can record the users tap position on the TB.

	(1)TESTING	(2)TRAINING	(3)TESTING
8 buttons	98.22	97.11	98.00
9 buttons	95.11	95.11	96.44
10 buttons	97.11	95.56	95.11

Table 1: Accuracy of different button configuration for different TESTING and TRAINING phases

Results and Discussion

Accuracy. The main result of the experiment is that the accuracy of for 8, 9 and 10 button configurations without any training were 98.21%, 95.11% and 97.11% respectively. After a TRAINING session, the task accuracy did not improve consistently across all the button configuration. The accuracy of different buttons at different phases is shown in Table 1.

Surprisingly, the accuracy for 10 button is higher than 9 button configuration. During the experiment, we found that the users were frequently looking at the TB while executing the task for 10 button configuration. The users had to count the number of buttons from the left or right end to map the target button on the TB. This is also corroborated by the fact the average time to execute a task for 10 button ($\mu = 1.26, \sigma = 0.69$) was higher than 9 button ($\mu = 1.10, \sigma = 0.46$) configuration.

Questionnaire. At the end of the session, the participants were asked how successful they were in executing the task for different button arrangements. The Likert scale for this questionnaire were set from 1 (very low) to 7 (very high). Their results are shown in Figure 4. In addition to this scale, the participants were asked if they used the bezel of the laptop to guide their fingers. The questionnaire reveals that none (yes=0, no=7, maybe=1) of the participant used the bezel of the laptop to guide their fingers. They were also

asked to state other visual clues or strategies they used to complete the task. We found that most participants divided the TB into left and right halves, and mapped the buttons to their fingers while executing the task. One of the participants reported that *"I divided up the buttons into groups of 3/4/5 and used a specific hand to press buttons in each group"*. As the number of buttons increased, the participants found it harder to map the fingers with the buttons while executing the task. For number of buttons greater than 8, the users would require to map their thumbs to the buttons which is inconvenient while using the TB.

Time. We also wanted to see the time it takes to execute a task for different button sizes. We found that time taken to execute a particular task for 8, 9 and 10 button configurations were ($\mu = 1.03, \sigma = 0.42$), ($\mu = 1.10, \sigma = 0.46$) and ($\mu = 1.26, \sigma = 0.69$) seconds. T-test reveal that the there was statistically significant ($p < 0.00$) increase in the time taken to execute the task as the number of buttons was increased. We believe that the homing time and time to touch the TB would be constant across different button configuration as the target button was randomly generated. Hence, the increase in time across the buttons would be due to increase in mental preparation before executing the task. It could also be due to the increase in the number of buttons the user had to count if he executed the task by counting the number of buttons from the ends of TB.

Conclusion and Future Work

We have presented a first phase of an ongoing research. We proposed GestTB, a gestural interaction technique for command selection on the touch bar. GestTB divides the TB into smaller areas which can be used to select commands using gestures. We conducted user study to determine the optimal size of area for our technique. As the number of buttons increases, the user can avail large number of commands. However, the users accuracy decreases as the number of buttons are increased. After accounting for users accuracy and the utility of the technique, we believe that 9 button configuration is the optimal number of buttons as it has 95.11% target selection accuracy. With 9 button configuration, GestTB allows for 72 and 576 commands in first and second level of hierarchy. In the next phase of our research, we will evaluate GestTB technique through a user study.

While our technique supports a large number of commands, it limits the number of commands that can be represented in a particular level of hierarchy to 9. We would like to work on ways to enable access to more number of commands in GestTB. One way to solve this problem is to allow the user to configure the application menu based on his needs. This would allow the user to put most relevant controls on the TB for easy access.

In future studies, we would like to investigate the effect of tactile feedback as the user performs the gestures. As observed in previous techniques [5], tactile feedback while performing the gesture improves the gesture accuracy. A textured film can be used to give tactile feedback as the user enters a new partition area while performing a gesture.

Acknowledgments

We thank Anirudha Joshi for his valuable inputs to the experimental design for the evaluation of GestTB. We are greatly indebted to the participants of the user study.

REFERENCES

[1] Apple. 2019a. NSTouch - AppKit. (2019). https://developer.apple.com/documentation/appkit/nstouch.

[2] Apple. 2019b. Touch Bar Overview. (2019). https://developer.apple.com/design/human-interface-guidelines/macos/touch-bar/touch-bar-visual-design/.

[3] Apple. 2019c. Touch Bar Overview. (2019). https://developer.apple.com/design/human-interface-guidelines/macos/touch-bar/touch-bar-overview/.

[4] Wenzhe Cui, Jingjie Zheng, Blaine Lewis, Daniel Vogel, and Xiaojun Bi. 2019. HotStrokes: Word-Gesture Shortcuts on a Trackpad. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, Article 165, 13 pages. DOI: http://dx.doi.org/10.1145/3290605.3300395

[5] Bruno Fruchard, Eric Lecolinet, and Olivier Chapuis. 2017. MarkPad: Augmenting Touchpads for Command Selection. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 5630–5642. DOI: http://dx.doi.org/10.1145/3025453.3025486

[6] Emmanouil Giannisakis, Gilles Bailly, Sylvain Malacria, and Fanny Chevalier. 2017. IconHK: Using Toolbar Button Icons to Communicate Keyboard Shortcuts. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 4715–4726. DOI: http://dx.doi.org/10.1145/3025453.3025595

[7] Carl Gutwin, Andy Cockburn, Joey Scarr, Sylvain Malacria, and Scott C. Olson. 2014. Faster Command Selection on Tablets with FastTap. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. Association for Computing Machinery, New York, NY, USA, 2617–2626. DOI: http://dx.doi.org/10.1145/2556288.2557136

[8] Teng Han, Jiannan Li, Khalad Hasan, Keisuke Nakamura, Randy Gomez, Ravin Balakrishnan, and Pourang Irani. 2018. PageFlip: Leveraging Page-Flipping Gestures for Efficient Command and Value Selection on Smartwatches. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 529, 12 pages. DOI: http://dx.doi.org/10.1145/3173574.3174103

[9] Gordon Kurtenbach and William Buxton. 1994. User Learning and Performance with Marking Menus. In *Conference Companion on Human Factors in Computing Systems (CHI '94)*. ACM, New York, NY, USA, 218–. DOI: http://dx.doi.org/10.1145/259963.260376

[10] David M. Lane, H. Albert Napier, S. Camille Peres, and Aniko Sandor. 2005. Hidden Costs of Graphical User Interfaces: Failure to Make the Transition from Menus and Icon Toolbars to Keyboard Shortcuts. *Int. J. Hum. Comput. Interaction* 18 (2005), 133–144.

[11] Pekka Parhi, Amy K. Karlson, and Benjamin B. Bederson. 2006. Target Size Study for One-handed Thumb Use on Small Touchscreen Devices. In *Proceedings of the 8th Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '06)*. ACM, New York, NY, USA, 203–210. DOI: http://dx.doi.org/10.1145/1152215.1152260

[12] Simon T. Perrault, Eric Lecolinet, Yoann Pascal Bourse, Shengdong Zhao, and Yves Guiard. 2015. Physical Loci: Leveraging Spatial, Object and Semantic Memory for Command Selection. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 299–308. DOI: http://dx.doi.org/10.1145/2702123.2702126

[13] Quentin Roy, Sylvain Malacria, Yves Guiard, Eric Lecolinet, and James Eagan. 2013. Augmented Letters: Mnemonic Gesture-Based Shortcuts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. Association for Computing Machinery, New York, NY, USA, 2325–2328. DOI: http://dx.doi.org/10.1145/2470654.2481321

[14] Joey Scarr, Andy Cockburn, Carl Gutwin, and Andrea Bunt. 2012. Improving Command Selection with CommandMaps. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 257–266. DOI: http://dx.doi.org/10.1145/2207676.2207713

[15] Joey Scarr, Andy Cockburn, Carl Gutwin, and Sylvain Malacria. 2013. Testing the Robustness and Performance of Spatially Consistent Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 3139–3148. DOI: http://dx.doi.org/10.1145/2470654.2466430

[16] Marcos Serrano, Eric Lecolinet, and Yves Guiard. 2013. Bezel-Tap Gestures: Quick Activation of Commands from Sleep Mode on Tablets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. Association for Computing Machinery, New York, NY, USA, 3027–3036. DOI: http://dx.doi.org/10.1145/2470654.2481421

[17] Daniel Vogel and Patrick Baudisch. 2007. Shift: A Technique for Operating Pen-based Interfaces Using Touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 657–666. DOI: http://dx.doi.org/10.1145/1240624.1240727

[18] Haimo Zhang and Yang Li. 2014. GestKeyboard: Enabling Gesture-Based Interaction on Ordinary Physical Keyboard. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems (CHI EA '14)*. Association for Computing Machinery, New York, NY, USA, 195–196. DOI: http://dx.doi.org/10.1145/2559206.2579485