November 06, 2017

Drawing flowcharts on touch-enabled devices

Wei Li

Erika Lu

Drawing flowcharts on touch-enabled devices

ABSTRACT

Users benefit from being able to draw flowcharts directly on touch-enabled devices without the hassle of dropdown menus for shape selection or operations to adjust shapes and sizes of drawn objects. This disclosure automatically creates shape elements that match a user's intention based on the drawing positions, shapes, sizes of strokes, etc. Beautified versions of a user's drawing strokes are generated and used to replace the strokes. Using the techniques of this disclosure, the touch device intelligently distinguishes between drawing positions, shapes, and handwriting. The techniques distinguish shape from handwriting text and eliminates interruptions to user workflow due to the need to switch between shape mode and text mode.

KEYWORDS

- Touchscreen
- Gesture UI
- Flowchart
- Handwriting recognition
- Interactive Whiteboard
- Mode switching
- Shape recognition

BACKGROUND

A flowchart is a graphical or symbolic representation of a process. Each step in the process is represented by a respective symbol and includes a short textual description of the process step. The flowchart symbols are linked together with arrows that indicate the process flow direction.

Computer applications that enable users to draw flowcharts and have several advantages over hand drawn charts. Such applications generate professional quality diagram and make easy various operations such as editing, collaborating, sharing, presenting, importing the flowchart into various reports, etc. Typical user interaction in such computer applications requires users to utilize dropdown menus to select various options. For example, to draw a connector in a flowchart, a user may be required to click on a 'line' button in a stencil depicted in a side menu bar and then drag the line to the desired location in the drawing canvas.

The user also may need to label different shapes and connectors in a flowchart. Each time the user creates, edits, or links different components in the flowchart, the user needs to select the correct category/mode, e.g., "text editor," "insert shapes," "insert connectors," etc. from a dropdown menu. Such mode switches interrupt the user's workflow. In applications that support freeform writing, modifying the writing using a keyboard is an undesirable user experience due to workflow interruption.

DESCRIPTION

Users benefit from being able to draw flowcharts directly on touch-enabled devices, e.g., tablets, smart phones, touch screen laptops, interactive whiteboards, etc., without the hassle of dropdown menus for shape selection or operations to adjust shapes and sizes of drawn objects. It is also useful to enable users to edit freeform (e.g., hand-drawn) text with gestures.

The techniques of this disclosure can be implemented on touch-enabled devices, e.g., tablets, smartphones, laptops with touch-screen technology, interactive whiteboards, etc. Freeform drawing is performed on touch-enabled devices using a finger or pen or stylus-based interface. Techniques described herein intelligently distinguish shapes from handwriting and enable users to generate drawings without interruption.

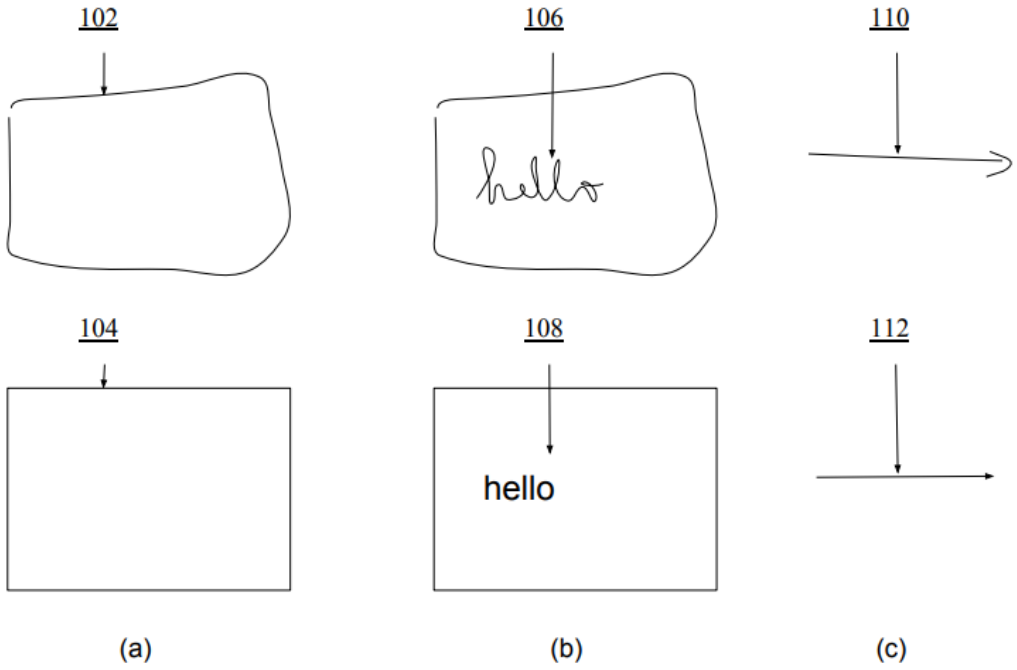

Fig. 1: Freehand drawing and writing interpreted and beautified: (a) Shape; (b) Shape with text; (c) Arrow

Fig. 1 illustrates interpretation of hand drawn shapes and textual labels. In an example of using freehand drawing to create a flowchart illustrated in Fig. 1(a), the hand drawn shape (102) is interpreted as a rectangle (104). Different shapes and arrows can have an optional text label. A label can be created by directly writing the text on the target shape. For example, the handwritten text ‘hello’ (106) is interpreted as text using handwriting recognition techniques and is automatically converted to printed format (108) as illustrated in Fig. 1(b).

Users represent connections between shapes with unidirectional or bidirectional arrows, e.g., by drawing arrows at the desired place and have the intended orientation. A hand drawn arrow is recognized and is replaced by a beautified arrow that connects two objects at the two ends of the arrow. Fig, 1(c) illustrates a hand-drawn connector (110) beautified, e.g., made

straighter, thicker, oriented symmetrical relative to the shapes, etc., into a unidirectional arrow (112) that is used to connect two shapes.

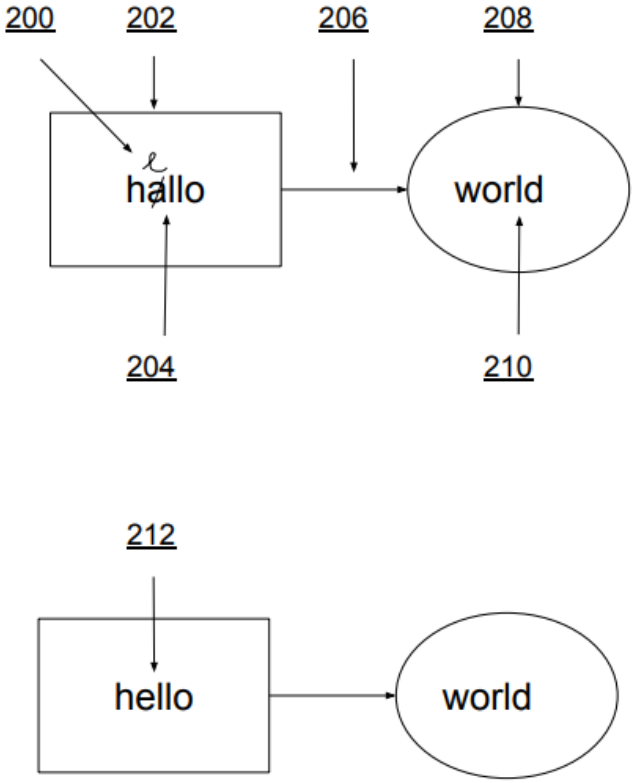

Fig. 2: Editing handwriting while creating a flowchart

Fig. 2 illustrates different gestures recognized and transformed into drawing elements. It also illustrates editing freeform drawing, e.g., handwriting. As shown in Fig. 2, a rectangle (202) and an oval (208) are connected using an arrow (206). The shapes are labeled ‘hallo’ (204) and ‘world’ (210). The user edits the word ‘hallo’ by drawing a line across the ‘a’ in ‘hallo’ (strikethrough), and writing ‘e’ on top, similar to traditional writing. The editing gesture is recognized and the word ‘hallo’ is modified to ‘hello’ (212). The user can edit text in various ways, e.g., replacing a letter of a word, deleting a letter, inserting letters or words, etc.

The order of drawing shapes and writing text is interchangeable. A user can create a shape followed by adding a label to it or the user can write a label followed by drawing a shape to enclose the label.

The techniques described herein intelligently distinguish shapes from handwriting. This enables users to continue drawing without workflow interruptions, even for drawings such as flowcharts that typically include a mix of shapes and texts. For example, without such techniques, users would need to switch between shape mode and text mode to enter corresponding data which introduces interruptions in the user workflow.

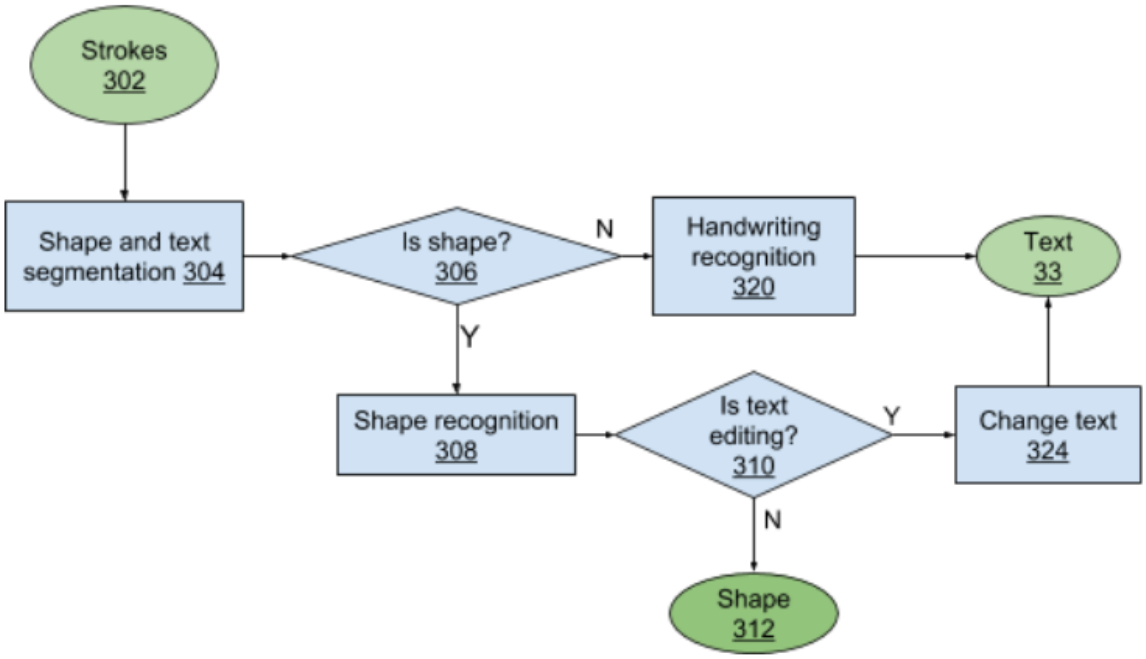

Fig. 3: Distinguishing shapes and text

Fig. 3 illustrates an example process to distinguish shapes and text. User strokes (302) are analyzed to distinguish between shape and text (304). This distinguishing step is also responsible for segmenting contiguous strokes into groups. Each group contains a single shape or a single string of text. The segmentation may also partition connected strokes to deal with the case when

users use one stroke to draw multiple shapes or to draw and write without lifting stylus or finger. Segmented stroke groups are sent to handwriting recognition or shape recognition depending on the segmentation result (306). A recognized shape (308) first goes through a test on whether it is a text editing gesture (310). If the gesture is a text editing gesture, it is used to modify existing text (324) to provide text (330). Otherwise, a shape is created (312). If the strokes are not a shape, handwriting recognition (320) is performed to provide text (330).

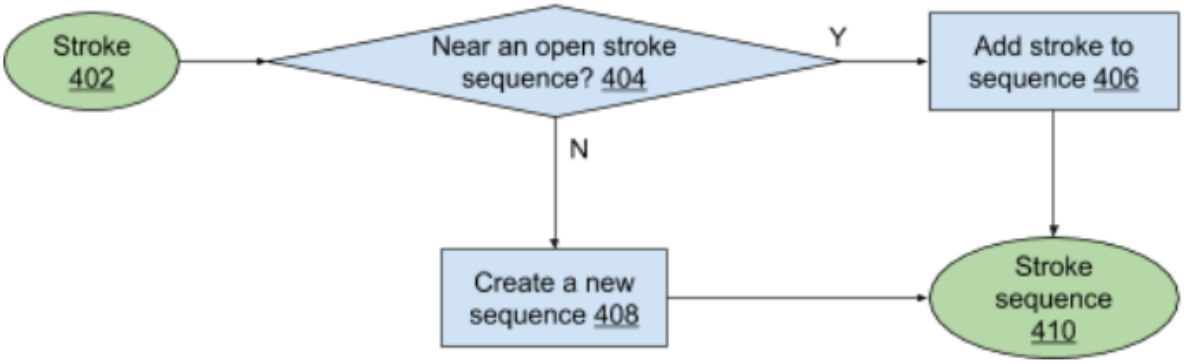

Fig. 4: Identifying stroke sequences

Fig. 4 illustrates an example process to group input strokes into stroke sequences, per techniques of this disclosure. A stroke sequence is composed of one to many strokes. A stroke sequence is created as 'open' initially. A new stroke (402) is added (406) to an open stroke sequence (410) if they are near each other spatially (404). Otherwise, a new stroke sequence is created (408). A stroke sequence becomes closed if no stroke is added to the sequence within a given timeout period.

A stroke sequence can also be closed early if it is determined that existing strokes in the sequence are unlikely to evolve to other shape or text. Each closed stroke sequence is recognized as a single shape, or a string of text. The grouping of strokes is based on the position and timing of input strokes. Naturally, the strokes forming a shape are within a vicinity. Similarly,

handwritten strokes are developed horizontally from left to right (or right to left for right-to-left languages).

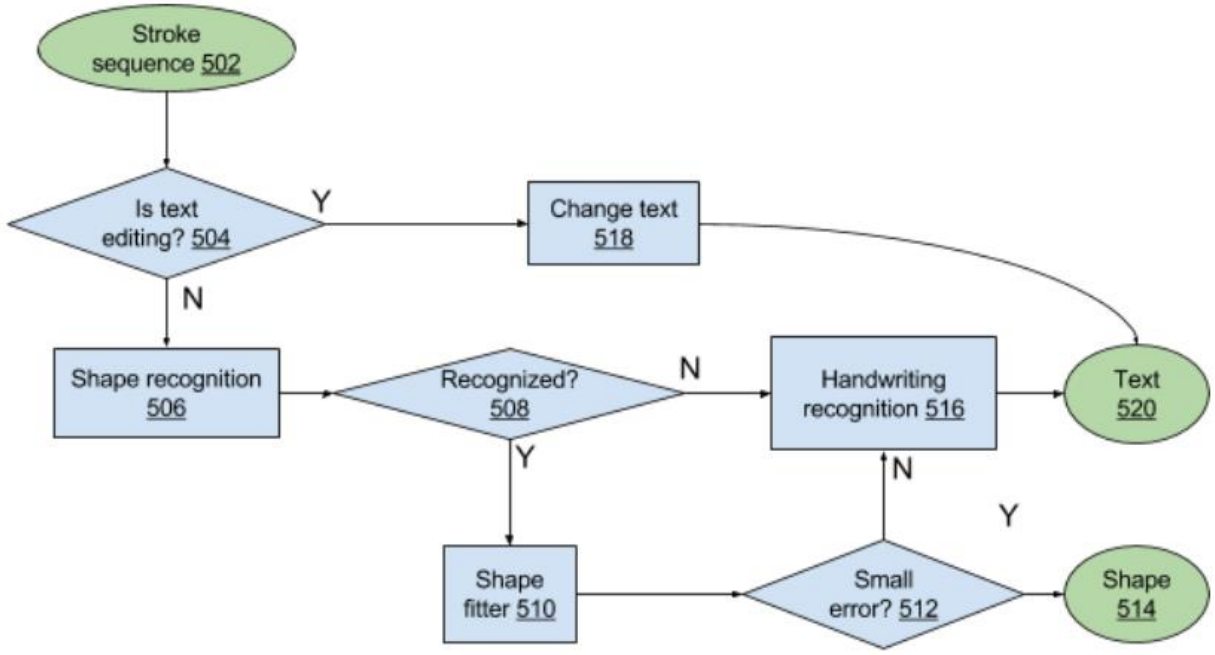

Fig. 5: Identifying shapes and text from stroke sequence

In an example scenario using a closed stroke sequence as segmentation, the techniques identify shapes and text from stroke sequences using the process illustrated in Fig. 5. First it is determined (504) whether a stroke sequence (502) is a text editing gesture. If the stroke sequence is a text editing gesture, text is edited (518) to provide result text (520). The described techniques utilize single stroke text editing gestures such that those can be quickly and reliably detected. Non-editing strokes are sent to a shape recognizer (506). In some implementations, the shape recognizer functionality may reside on the device and therefore have very short response times.

Continuing with this example scenario, next it is determined whether the shape is recognized (508). If the shape is recognized, a shape fitter (510) is applied and the output produced by the shape fitter is checked for errors (512). Geometric computation is performed to

fit hand drawn strokes to known shapes and measure fitting errors. If the fitting error is less than a defined tolerance threshold, a shape is created (514). For strokes that are not recognized or matched to know shapes or where the fitting error is large, handwriting recognition is performed (516) to provide result text (520).

Handwriting recognition is requested whenever a new stroke is added to the stroke sequence. A stroke sequence is terminated immediately if the status of existing strokes cannot change. Some early termination rules for stroke sequences within this framework are as follows:

- A stroke sequence is recognized as a shape that can't be part of another shape and shape can't be part of any glyph.
- When the total fitted line segments are more than the maximum number of line segments in shapes that are supported by the described techniques.

In this described example scenario using a closed stroke sequence as a segmentation for flowcharting handwriting recognition is implemented when shape recognition fails. Alternately, stroke sequences may be analyzed by two recognizers in parallel. Lead time for analysis may be reduced by selecting the output from one of the recognizers, depending on the results.

In some implementations of the described techniques the following implementations may be made. If handwriting editing is detected, then all pending strokes are processed as text. If a stroke is written over an existing shape, and both the start and the endpoint fall within the shape, then process the stroke as text. If an ellipse drawn is smaller than a predefined threshold then a wait period is utilized before "closing" the ellipse, e.g., to allow the user to add more strokes. Detected triangle shapes are closed only when the confidence score for triangle is above a certain threshold to avoid letters being mistaken for triangles. Similarly, a separate confidence threshold is used for a star shape to ensure that it can be easily drawn.

In situations in which certain implementations discussed herein may collect or use personal information about users (e.g., user data, information about a user's social network, user's location and time at the location, user's biometric information, user's activities and demographic information), users are provided with one or more opportunities to control whether information is collected, whether the personal information is stored, whether the personal information is used, and how the information is collected about the user, stored and used. That is, the systems and methods discussed herein collect, store and/or use user personal information specifically upon receiving explicit authorization from the relevant users to do so. For example, a user is provided with control over whether programs or features collect user information about that particular user or other users relevant to the program or feature. Each user for which personal information is to be collected is presented with one or more options to allow control over the information collection relevant to that user, to provide permission or authorization as to whether the information is collected and as to which portions of the information are to be collected. For example, users can be provided with one or more such control options over a communication network. In addition, certain data may be treated in one or more ways before it is stored or used so that personally identifiable information is removed. As one example, a user's identity may be treated so that no personally identifiable information can be determined. As another example, a user's geographic location may be generalized to a larger region so that the user's particular location cannot be determined.

CONCLUSION

The techniques of this disclosure provide a simple and seamless process of creating and editing flowcharts and drawings on touch-enabled devices using a finger, stylus, etc. Using techniques such as shape recognition and fitting, hand drawn gestures, e.g., strokes, sizes,

positions, etc., are interpreted to create shape elements that match a user's intention. The techniques allow users to seamlessly edit drawings by gestures such as text strikethrough, insertion of text using the caret symbol, drawing connectors, etc. The techniques eliminate the need to use the keyboard or to change between modes, e.g., a text mode and a shape mode, in a drawing software application. The concept and techniques in this disclosure are applicable to any system that involves freehand drawing using a touch or gesture interface. It is extensible to the 3D space as well. For example, it is a natural fit for annotations in virtual reality and augmented reality.