thumb orientation. These would preferably correspond to "Space" and "BackSpace" key regions for the right and left thumbs, respectively. Such minimal depressions can tactilely guide users' hands back to home row of the key layout without requiring users to look down at the surface and without seriously disrupting finger chord slides and manipulations on the surface.

[0280] The positions of key regions off home row can be marked by other types of tactile indicators. Simply roughening the surface at key regions does not work well. Though humans easily differentiate textures when sliding fingers over them, most textures cannot be noticed during quick taps on a textured region. Only relatively abrupt edges or protrusions can be sensed by the users' fingertips under typing conditions. Therefore, a small raised dot like a Braille dot is formed on top of the surface at the center of each key region. The user receives feedback on the accuracy of their typing strokes from where on the fingertip a dot is felt. This feedback can be used to correct finger aim during future keypresses. Since single finger slides are ignored by the chord motion recognizer, the user can also slide a finger around the surface in tactile search of a particular key region's dot and then tap the key region when the dot is found, all without looking at the surface. Each dot should be just: large enough to be felt during tapping but not so large as to impede chord slides across the surface. Even if the dots are not large enough to impede sliding, they can still corrupt proximity and fingertip centroid measurements by raising the fingertip flesh near the dot off the surface thus locally separating the flesh from the underlying proximity sensing electrode. Therefore, in the preferred embodiment, the portion of each dot above the surface dielectric is made of a conductive material. This improves capacitive coupling between the raised fingertip flesh and the underlying electrodes.

[0281] FIG. **42** shows the steps within the keypress detection loop. Step **750** retrieves from the current identified path data **250** any paths which were recently created due to hand part touchdown or the surface. Decision diamond **752** checks whether the path proximity reached a keypress proximity thresh for the first time during the current sensor array scan. If the proximity has not reached the threshold yet or has already exceeded it previously, control returns to step **750** to try keypress detection on the next recent path. If the path just crossed the keypress proximity threshold decision diamond **754** checks whether the contact path has been identified as a finger rather than a palm. To give the users the freedom rest the palms anywhere on the surface, palm presses should not normally cause keypresses, and are therefore ignored. Assuming the path is a finger, decision diamond **756** checks whether the hand the identified finger comes from is currently performing a chord slide gesture or writing via the pen grip hand configuration. Asynchronous finger presses are ignored once these activities have started, as also indicated in step **660** of FIG. **40**A. Assuming such hand activities are not ongoing, decision diamond **757** proceeds with debounce tests which check that the finger has touched the surface for at least two sensor array scan cycles and that it had been off the surface for several scan cycles before touching down. The path tracking module (FIG. **22**) facilitates such liftoff debouncing by reactivating in step **334** a finger's old path if the finger lifts off and quickly touches back down over the same spot. Upon reactivation the time stamp of the last liftoff by the old path must be preserved for comparison with the time stamp of the new touchdown.

[0282] If all of these tests are passed, step **758** looks up the current path position ($P_x[n],P_y[n]$), and step **760** finds the key region whose reference position is closest to the fingertip centroid. Decision diamond **762** checks that the nearest region is within a reasonable distance of the finger, and if not causes the finger press to be ignored. Assuming a key region is close to the finger, step **764** creates a keypress element data structure containing the path, index identifier and finger identity, the closest key region, and a time stamp indicating when the finger crossed the keypress proximity threshold. Step **766** then appends this element data structure to the tail of a FIFO keypress queue. This accomplished, processing returns to step **750** to process or wait for touchdowns by other fingers.

[0283] The keypress queue effectively orders finger touchdowns by when they pass the keypress transmitted to the host. However, an element's key symbol is not assured transmission of the host once in the keypress queue. Any of a number of conditions such as being part of a synchronized subset of pressing fingers can cause it to be deleted from the queue before being transmitted to the host. In this sense the keypress queue should be considered a keypress candidate queue. Unlike the ordered lists of finger touchdowns and releases maintained for each hand separately in the synchronization detector, the keypress queue includes and orders the finger touchdowns from both hands.

[0284] FIG. **43**A shows the steps within the keypress acceptance and transmission loop. Step **770** picks the element at the head of the keypress queue, which represents the oldest finger touchdown which has neither been deleted from the queue as an invalid keypress candidate nor transmitted its associated key symbol. Decision diamond **772** checks whether the path is still identified as a finger. While waiting in the queue path proximity could have increased so much that the identification system decides the path is actually from a palm heel, in which case step **778** deletes the keypress element without transmitting to the host and step **770** advances processing to the next element. Decision diamond **774** also invalidates the element if its press happened synchronously with other fingers of the same hand. Thus decision diamond **774** follows through on deletion command steps **601**, **612**, **615**, **620** of the synchronization detection process (FIG. **39**). Decision diamond **776** invalidates the keypress if too much lateral finger motion has occurred since touchdown, even if that lateral finger motion has not yet caused a chord slide to start. Because users may be touch typing on the surface, several millimeters of lateral motion are allowed to accommodate glancing fingertip motions which often occur when quickly reaching for keys. This is much more glancing tap motion than is tolerated by touchpads which employ a single finger slide for mouse cursor manipulation and a single finger tap for key or mouse button click emulation.

[0285] Decision diamond **780** checks whether the finger whose touchdown created the keypress element has since lifted off the surface. If so, decision diamond **782** checks whether it was lifted off soon enough to qualify as a normal key tap. If so, step **784** transmits the associated key symbol to the host and step **778** deletes it from the head of the queue. Note that a keypress is always deleted from the queue upon