Designing IR gesture-sensing systems

Alan Sy, Silicon Laboratories - June 09, 2011

Touchless user interfaces are an emerging technology for embedded electronics as developers seek to provide innovative control methods and more intuitive ways for end users to interact with electronics products. Active IR (infrared) proximity-motion-sensing technology can solve this human-interface design challenge.

Thanks to the advent of highly integrated proximity/ambient-light sensors, implementing motion sensing using IR technology is now easier. The two primary methods used to enable gesture sensing are position-based and phase-based sensing. Position-based gesture sensing involves finding gestures based on the calculated location of an object. Phase-based gesture sensing is based on the timing of the changes in signals to determine the direction of an object’s motion. Both technologies are complementary enablers of IR gesturing applications, such as page turning for e-readers, scrolling on tablet PCs, and navigating GUIs (graphical user interfaces) in industrial-control systems.

Hardware considerations

Although touchless-interface applications primarily involve gestures made by a human hand, gesture-recognition concepts can also apply to other targets, such as a user’s cheek. Application and system constraints dictate IR gesture-sensing range requirements. Object reflectance is the main measurable component, and a hand is the most common detectable object. A hand can achieve gesture sensing up to 15 cm away from the proximity sensor. Fingers, with dramatically lower reflectance than hands, can achieve gesture sensing at a range of less than 1 cm for thumb-scroll applications.
The general guideline for designing a gesture-sensing system with multiple LEDs (light-emitting diodes) is to ensure that there is no “dead spot” in the middle of the detectable area. When a target is placed above the system and is not detected, the target is in a reflectivity dead spot. To avoid dead spots, the LEDs must be placed such that the emitted IR light can reflect off the target and onto the sensor from the desired detection range (Figure 1). The most likely area for a dead spot is directly above the sensor, between the two LEDs. The two LEDs are placed as close to the edge of the target as possible to provide feedback in the middle while maintaining enough distance between the LEDs so that the target can be detected when the finger or hand moves left or right.

The location and reflectance of the target in relation to the system are also important. Note that the proximity sensor in Figure 1 is located under the palm of the hand and in the middle of the finger. The fingers are poor focal points for hand-detection systems because light can slip between the fingers. The shape of the fingers also results in unpredictable measurements. For a finger-detection system, the tip of the finger is curved and reflects less light than the middle of the finger.

**Position-based gesture sensing**

The position-based motion-sensing algorithm involves three primary steps. The first step is the conversion of the raw data inputs to usable distance data. The second step uses this distance data to estimate the position of the target object, and the third step checks the timing of the movement of the position data to determine whether any gestures have occurred.

The proximity sensor outputs a value for the amount of IR light that the IR LEDs reflect. These values increase as an object or a hand moves closer to the system and reflects more light. Assume that a hand is the defined target for detection. The system can estimate how far away the hand is based on characterization of the PS (proximity-sensing) feedback for a hand at certain distances. For example, a hand approximately 10 cm away yields a PS measurement of 5000 counts, so subsequent PS measurements of 5000 counts mean that a similarly reflective hand is approximately 10 cm away from the system. Taking multiple data points at varying distances from the system helps you interpolate between these points and creates a piecewise equation for the conversion from raw PS counts to a distance estimation.
Each LED in a system with multiple LEDs has a different PS feedback for each hand distance, so each LED will need an independent counts-to-distance equation. For a two-LED system, each LED must be characterized with a target suspended over the midpoint between the LED and the sensing device (Figure 2). When Target 1 is suspended over the sensing device and LED\(_1\), the measured feedback will correlate to a distance, \(D_1\), above the system. The same is true for Target 2, LED\(_2\), and \(D_2\).

The next step is to estimate the target’s position using the distance data and the formula for the intersection of two circles. An LED’s light output is conical; for this approximation, however, it is considered to be spherical. With the LEDs on the same axis, the intersection of these two spheres can be considered using equations for the intersection of two circles.

When a target is suspended over the middle of a system, \(D_1\) and \(D_2\) are the estimates of the distance from points \(P_1\) and \(P_2\) to the target above the system (Figure 3). Think of \(D_1\) and \(D_2\) as the radii of two circles; the intersection of these two circles is the location of the target.

Figure 4 is an expanded version of Figure 3, in which the measurements A and B label the location of the target along the axis between points \(P_1\) and \(P_2\). The distance measurements \(D_1\) and \(D_2\) have been renamed \(R_1\) and \(R_2\) to indicate that they are now considered radii. The value of A is the location of the object along the axis between \(P_1\) and \(P_2\). A negative value is possible, indicating that the target is on the left side of \(P_1\). The distance to the target is a function of these variables, as the following equations show: \(D = A + B\), and \(A = (R_1^2 - R_2^2 + D^2)/2 \times D\).
With the positioning algorithm in place, keeping track of timing allows the system to search for and acknowledge gestures. The entry and exit positions are the most important considerations for hand-swiping gestures. A left swipe occurs if the hand enters the right side with a high A value and exits the left side with a low A value. This scenario assumes that the entry and exit transpired within a defined time window. If the position stays steadily in the middle area for a set period, this gesture can be considered a pause gesture.

The system must keep track of the time stamps for the entry, exit, and current positions of the target in the detectable area. You can easily recognize most gestures with this timing and position information. Timing will need to be custom-tuned for each application and each system designer’s preference.

**Phase-based gesture sensing**

With phase-based gesture sensing, the location of the target is never calculated. This method involves looking solely at the raw data from the proximity measurements and identifying the timing of the changes in feedback for each LED. The maximum feedback point for each LED occurs when a hand is directly above that LED. If a hand is swiped across two LEDs, the direction of the swipe can be determined by looking at which LED’s feedback rose first.

When a hand is swiped left over a three-LED system, it crosses over \( D_2 \), then \( D_3 \), and then \( D_1 \) (**Figure 5**). The sensing algorithm recognizes the rise in feedback for \( D_2 \) and records the time stamp for this rise. The algorithm then detects the same rises for \( D_3 \) and \( D_1 \) with a later time stamp than the one before it. The algorithm also can detect the return of each LED’s measurement to the no-detection state and can record a time stamp for this event. In this case, \( D_2 \) returns first to a normal state, then \( D_3 \), and then \( D_1 \).
For up and down gestures, $D_1$ and $D_2$ rise and fall simultaneously, with $D_3$ coming either before or after $D_1$ and $D_2$ for the up or down gesture. If a hand approaches the system directly from above and then retracts to indicate a “select” gesture, all three channels rise and fall at once.

**Figure 6** shows the signal responses of the right, left, down, and up gestures, which appear as ADC counts versus time. The green line represents PS measurements using $D_1$, the purple line represents PS data from $D_2$, and the yellow line shows data from $D_3$. For a right swipe, $D_1$ spikes first, followed by $D_3$ and then $D_2$. For the up and down swipes, $D_1$ and $D_2$ spike simultaneously because the hand crosses these LEDs at the same time when swiping up or down.

### Advantages and drawbacks

The position-based method can offer information on the location of the target, enabling ratiometric control of systems. For example, to scroll several pages through a book, you could suspend your hand over the right side of the detectable area rather than making several right-swipe gestures.

The main drawback of the position-based algorithm is the accuracy of the position calculations. The positioning algorithm assumes a spherical output from the LEDs, but in practice LED output is more conical than spherical. The algorithm also assumes uniform light intensity across the entire output of the LED, but the light intensity decays away from the normal. Another issue is that this algorithm does not account for the target’s shape. A uniquely shaped target causes inconsistencies with the positioning output. For example, the system cannot tell the difference between a hand and a wrist, so it is less accurate when detecting any gestures involving movement that puts the wrist in the area of detection. The positioning algorithm is adequate for low-resolution systems that need only a $3 \times 3$ grid of detection, but the algorithm is not suitable for pointing applications. In short, this algorithm’s output is not an ideal touchscreen replacement.

The phase-based method provides a robust way of detecting gestures in applications that do not require position information. Each gesture can be detected on either the entry or the exit from the detectable area, and the entry and exit can be double-checked to provide much higher certainty for each observed gesture.

The drawback of this method is that it provides no positioning information, meaning that it offers a more limited number of gestures that can be implemented than does the position-based method. The phase-based method can tell only the direction of entry and exit from the detectable area, so it does not recognize any movement in the middle of the detectable area.

### Combining both methods

Both position- and phase-based methods of gesture sensing can be implemented in concert to help
mask and mitigate each method’s inherent deficiencies. The position-based algorithm can provide some positional information for ratiometric control, and the phase-based algorithm can detect most gestures. These two algorithms together provide a robust approach for gesture-sensing applications. This dual-method approach requires more code space to implement and requires additional CPU cycles to process both algorithms. For a growing number of sophisticated human-interface applications, however, it may be well worth the computational trade-off to enable the next generation of gesture sensing.