Chapter 1

Scoring of Gestures

1.1 Angle Based Comparison Function

1.2 Component Based Comparison Function

The second way by which a score is assigned to a user attempted gesture will be discussed in this section.

The idea behind this method is to take a given hand and decompose it into smaller component that can be scored individually. Then these components scores will be combined to arrive at the cumulative score for the entire hand. Each finger is seen as a component. After considering all of the gestures being tested in this project, I realized that each finger can be in one of three main kinds of poses. All of the fingers except for the thumb are only seen in some variations of being straight, or being curved. The thumb, however, has its own special kind of pose, which deals with connecting to other fingers. For example in some of the gestures the thumb is touching the pinky; in some gestures it is touching the middle finger. Therefore, the third possible pose is represented by a finger name, such as "pinky" or "index" etc., and it represents the finger the thumb is touching in the gesture being analyzed. The way the algorithm is designed, it makes sense to assign this dynamic third pose to the thumb only. Figure 1.1 and Figure 1.2 shows one of the gestures used in this project, namely gesture9Left, to illustrate what is meant by the different poses different components of the hand can take. As we can see, all of the fingers are straight; the ring finger is curved; and the thumb is touching the ring finger. This "pose signature" of this gesture as it is used in code is shown in Figure 1.3. The pose signature for a certain gesture is the same regardless of whether left or right hand is being used. That is why the code sample shows two case statements for gesture9Left and gesture9Right.

The comparison function for the component based scoring of hand gestures is shown in Figure 1.4. It returns a number 0-100 just like the angle based comparison function to indicate the score for the hand being graded. This function gets the fingers for the hand and goes through and finds the individual grades for each finger. Then it combines the into a cumulative grade by weighing the fingers equally.

To give a clearer idea about how the fingers actually get graded, the gradeFinger() function is shown in Figure **??**. This function relies on three helper functions which calculate the straightness and curvedness of fingers and a function which returns the score for the thumb. These helper functions are shown in Figure 1.6; getThumbScore() the more complicated of the three. The way a score is calculated for a thumb and the finger it is supposed to be touching is by finding the distance between the tip bone of the thumb and any of the three outermost bones on the finger. The smallest distance is chosen as the tip of the thumb might be closer to any three of the distal, intermediate or proximal bones. This distance is scaled down

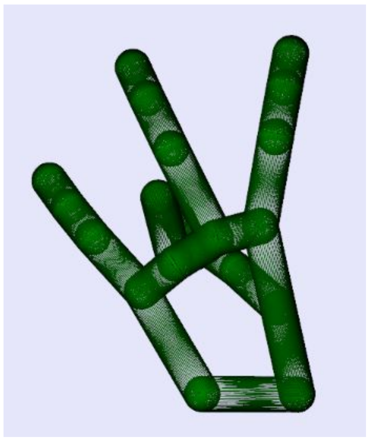

FIGURE 1.1: Gesture showing different finger poses.

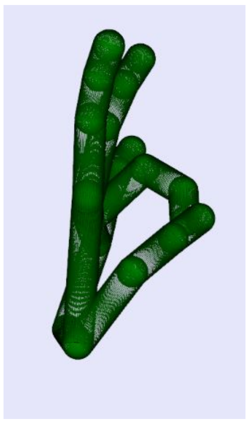

FIGURE 1.2: Same gesture after a 90 degree rotation.

```
case "gesture9Left":
case "gesture9Right":
   fingerPoseMap.put("index", "straight");
   fingerPoseMap.put("middle", "straight");
   fingerPoseMap.put("ring", "curved");
   fingerPoseMap.put("pinky", "straight");
   fingerPoseMap.put("thumb", "ring");
   break;
```

FIGURE 1.3: In the component based scroing, each gesture type gets a certain mapping for the kinds of poses fingers are expected to be in for that gesture.

by the smallest bone length multiplied by a scaling factor. Like the other two functions, straightnessOfFinger() and curvednessOfFinger(), the score that is returned is snapped to be between 0-1.

```java
public static int compare(Hand h, String gestureType) {
   FingerList fingerList = h.fingers();
   //make sure you have five fingers
   if (fingerList.count() == 5) {
      //calculate grades for each finger
      HashMap<String, Double> grades =
          getFingersGradedMap(getFingerHashMap(fingerList),
          getFingerPoseMap(gestureType));
      //grade for whole hand
      double totalGrade = cumulativeGrade(grades);
      //score 0-100
      return (int) (totalGrade * 100.0);
   }
   return -1;
}
```

FIGURE 1.4

```java
private static double gradeFinger(HashMap<String, Finger> fingerMap, Finger f,
    String pose) {
   if (pose.equals("straight")) {
      return straightnessOfFinger(f);
   } else if (pose.equals("curved")) {
      return curvednessOfFinger(f);
   }
   //thumb is not touching any finger
   else if (pose.equals("thumb")) {
      return straightnessOfFinger(f);
   }
   //thumb touching other fingers
   else {
      Finger theFingerThumbTouches = fingerMap.get(pose);
      return getThumbScore(f, theFingerThumbTouches);
   }
}
```

FIGURE 1.5: Given a finger a certain pose, this function returns a grade (0-1) for that finger.

```java
private static double straightnessOfFinger(Finger f) {
   //best case = 0; worst case is: 90+90+90 = 270.
   double sumOfAngles = getSumOfThreeAnglesBetweenFingerBones(f);
   sumOfAngles = sumOfAngles - 30;//offset by 30 degrees
   double score = sumOfAngles / 270;//closer to 0 means a better score
   score = 1 - score;//conventional scale: 0 = bad, 1 = good.
   return snapScore0to1(score);
}
private static double curvednessOfFinger(Finger f) {
   //best case is: 90+90+90 = 270; adjusted bestcase = 210; worst case = 0;
   double sumOfAngles = getSumOfThreeAnglesBetweenFingerBones(f);
   double score = sumOfAngles / 210;//closer to 1 means a better score
   return snapScore0to1(score);
}
private static double getThumbScore(Finger thumb, Finger otherFinger) {
   //bones in thumb and finger
   HashMap<String, Bone> thumbMap = getHashMapOfBonesFromFinger(thumb);
   HashMap<String, Bone> fingerMap = getHashMapOfBonesFromFinger(otherFinger);
   //get center point of thumb's tip bone
   Vector thumbTip = thumbMap.get("distal").center();
   //finger bones
   Bone d = fingerMap.get("distal");
   Bone i = fingerMap.get("intermediate");
   Bone p = fingerMap.get("proximal");
   //length of bones
   float smallestBoneLength = (Math.min(Math.min(d.length(), i.length()),
       p.length()));
   //distances from thumb tip to finger bones
   float d1 = thumbTip.distanceTo(d.center());
   float d2 = thumbTip.distanceTo(i.center());
   float d3 = thumbTip.distanceTo(p.center());
   double minDistance = (double) (Math.min(Math.min(d1, d2), d3));
   //scale and score
   double distanceScaledByBoneLength = minDistance / (smallestBoneLength * 3);
   double score = 1 - distanceScaledByBoneLength;
   return snapScore0to1(score);
}
```

FIGURE 1.6: The three helper functions used to calculate grades for a finger in one the three main kinds of poses.