Localization in Electronic Fabric

Robbie Hott

May 3, 2006

1 Introduction

With the development of smaller and more powerful sensors, and the continuing work on integrating electronic wires into clothing fabrics, combining the two becomes a strong means to produce a garment that is versatile but also reprogrammable and useful in many applications, including the military. Basically, this paper intends to set forth an algorithm to set up a sensor network attached to a shirt and allow each node to discover its position in relation to the body of the wearer, in the case that disaster strikes the wearer or problems occur. Furthering this algorithm, the shirt could also be used for entertainment purposes if the sensor nodes are equipped with lights or color changing panels. The goal is also to implement this algorithm as a simulation to show its power, usability, and accuracy.

2 Related Work

There have been a lot of work into putting computers into wearable means. Some earlier work [2] describes the potential of wearing computers, such as sewn in keypads, dresses that light up due to conductive fabrics, and jackets that play music by depressing the keypad sewn into the fabric. Post and Orth opened the way for the paper in [2] with their work in 1997 to weave metallic yarns into fabric [7]. They describe running thin silk threads wrapped in copper foil, similar to telephone wire. This paper makes the claim that resistors, capacitors, and coils can all be sewn out of fabric, but other components, such as LEDs, crystals, piezo transducers, and others, must be attached by soldering directly to the metallic yarn. However, with sensor nodes now becoming smaller and capable of more (such as giving off light energy, we can sew more into the fabric itself.

Some of the main work in this field has been in the medical technology department, making monitors and other healthcare essentials wearable. Pentland in [6] describes glasses that act as a memory aid and human assistant to those prone to memory loss, as well as a sociometer that can be worn to gather data about a persons daily interactions, which can help out those with depression or mania. Ottenbacher in [5] developed a Bluetooth based ECG system that could be worn as a t-shirt. Chen, Wei, Cohen, Ding, Tokinoya, and Takeda design and develop a vest to be worn that can tell if a wearer falls, to help with older patients prone to osteoporosis, can process heart beat, respiration rhythm from ECG, walking cadence, and body gesture from acceleration in [1]. Each of these works, although great leaps for healthcare, are also bulky and require extra components. With more smaller sensors that can be reprogrammed becoming popular, having clothing that can discover where it fits into the human shape and having the ability to reprogram the sensors to detect different key possible problems could also revolutionize this field.

3 Background

Post and Orth have shown that weaving conductive metals safely in clothing is possible and feasible [7]. Using that, a shirt can be sewn with the conductive fabric in a grid-like pattern throughout. At each junction point, a node can be sewn in to create a grid of nodes. These nodes are hardwired together, however they must have the capability to send data wirelessly to an outside PDA or computer to allow for outside-power-driven computations. Since the shirt is designed, the nodes can be hardcoded with their grid number (coordinates) such as $(2, -56)$ or $(14, 60)$. The best way to do this is shown in Figure 1. Now we are ready to set up the algorithm.

3.1 Challenges

This problem poses many challenges that must be overcome. The main obstacle I must overcome is the inaccessibility of a large testbed of nodes to test this algorithm on, and moreso, not having the ability to design a true prototype shirt. This will be overcome by using a simulation of the shirt, however, future studies would require the use of a prototype. Other obstacles include dealing with battery life and dead node issues, which could happen as nodes become unusable, but this algorithm does not address those issues, and I will leave them for future studies in this area.

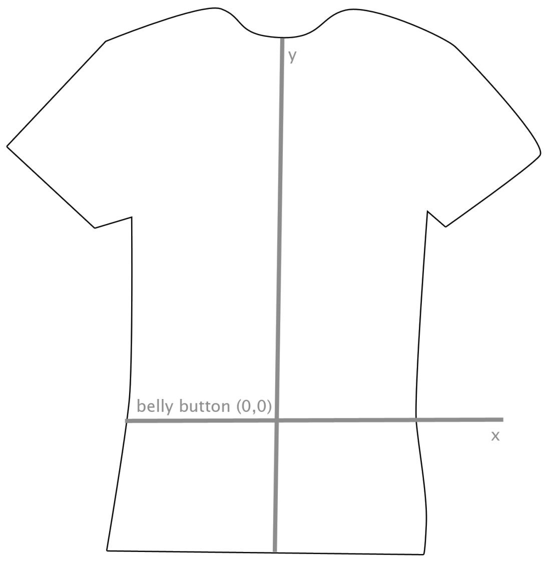

Figure 1: Coordinate system for the t-shirt.

The algorithm does, however, address the issue of accuracy, which is a large challenge in the design of a shirt and node structure of this magnitude. It addresses this challenge by using an outside source of computation power, a PDA, to gather data, match reference points to human form, and compute approximate hop distance using multiple reference points and data provided by the sensors.

4 Algorithm

The first thing that we must consider for the correct placement and communication of the nodes is an algorithm to set up the location of each node in relation to the other nodes as well as its placement on the body. This algorithm has some initial assumptions. Since the nodes are sown into the fabric, we can hard wire them with their placement on the shirt, for example column 3, row 60. This placement will help determine distance between nodes, since shirts can stretch depending on body size. Now, consider the algorithm as follows:

- First, the wearer must have a PDA available to provide power to help the shirt to do computations.
- The wearer must press certain reference points on the shirt:
  1. Inside collar bones on both sides of the neck
  2. Top end of each shoulder blade
  3. Belly Button
  4. Top of the hips
- Each sensor under the reference points must then report to the PDA that it is a reference point.
- The PDA matches those points up to a generic human body model to decide which point is where on the body. It also measures the distance between the points on the neck and the points on the hips and averages the mean hop length (it knows how many hops are between nodes because the nodes can send their shirt coordinates).
- The PDA then sends hop length to each of the original sensors, also telling them which reference point they are.
- This data is then sent through the fabric in all directions. The data includes the hop distance, number of hops up/down from the source, number of hops right/left from the source, and classification of the source. Each node would receive and save the information, increment or decrement appropriately the number of hops in the correct direction to include itself in that number, then pass the information along to the nodes around it, using the following scheme:
  1. If the node is the source, it will send it in all directions.
  2. If the data comes in from the bottom, the node will increment the number of hops up and send it out along the wires at the top, left, and right.
  3. If the data comes in from the top, the node will decrement the number of hops up (going down) and send it out along the wires at the bottom, left, and right.
  4. If the data comes in from the left, the node will increment the number of hops left and the data will be passed on to the right.
  5. If the data comes in from the right, the node will decrement the number of hops left (going right) and the data will be passed on to the left.

  See Figure 2 for an example.

With this algorithm, each sensor can send its number of hops from the reference points to a receiver that wants to examine a patient wearing the shirt or an officer wanting to check where and how bad a wound is, along with the hop distance. That receiver can then piece together the information, using Pythagorean Theorem and Euclidean distance formulas, to pinpoint exactly where on the body the sensor is. Similarly the sensors over the heart could be reprogrammed

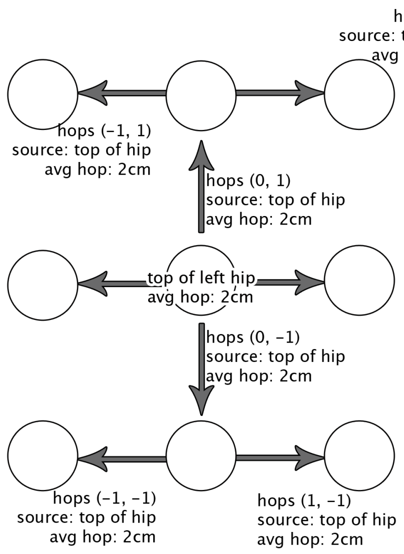

Figure 2: Sample of the algorithm in action.

to measure heartbeat and report when there is a problem with the heart.

4.1 Pseudocode

For a further discussion of the algorithm, I will discuss the pseudocode of the code that each node will run. I only give the pseudocode so that it can be easily rewritten in any language a programmer chooses. In the discussion of the Simulation, I will discuss an actual version of code that was used.

Figure 3 describes the main algorithm that will run in each of the nodes. If an event occurs, it will run one of the other algorithms. If there is pressure to set up this node, the node will run the code in Figure 10 in order to deal with that information as described above. Upon network data to disperse where the node is in relation to other nodes and the body underneath, the code

```
function run() {
  while(1) {
    if (network_data.event)
      pass_info(network_data.data);
    if (pressure.event
             && !pressure.happened_before())
      disperse_info(pressure.data);
    if (other.event)
      run_programmable(other.data);
  }
}
```

Figure 3: Pseudocode of the main node algorithm

in Figure 11 is run. The user may also specify code that is run in certain other events, and the `run_programmable()` function should take in the event data, decide which function is necessary to deal with that data, and call that function with the data.

5 Performance Metrics

There are some performance constraints that this algorithm must maintain in order to be effective on a t-shirt. They include power consumption, speed, and accuracy.

5.1 Power Consumption

For the algorithm to be effective, the nodes must consume as little power as possible in order to allow the shirt to be used many times and for there to be enough power to maintain the sensors throughout wearing. Power consumption of the communication, wired and wireless, processing, and listening costs will be factored in per node and per garment in order to calculate what the algorithm uses, the minimal needed, and how this algorithm can be optimized to use less power.

5.2 Speed

The algorithm must be fast in figuring out sensor placement on the shirt. If it takes ten minutes or more, the applications it is useful for drops. Logging the speed of communication and computation for our simulation and fitting that into the communication scheme in the shirt will help to get an evaluation for the speed it will take the shirt to initialize. Again I would like to look at minimizing the time necessary for initialization, and set upper bounds on the time required.

5.3 Accuracy

Accuracy is the most important metric I will use to evaluate this algorithm. If the algorithm is very inaccurate, then it will not be as useful. A theoretical bound for accuracy is perfection, so I will try to achieve perfection, however due to differing body shapes, I will only get an approximation. Therefore, measuring how far off from the true body shape and node hop distance the approximation is, will be key. To do that, I plan to run my simulation and compare the algorithm's approximation to the actual original body shape, using easily identifiable points on the body.

6 Simulation

I wrote a simulation for this algorithm in C++ that would test the accuracy of this algorithm. It contained a 2-dimensional array of structs containing data about the hop distance, direct distance to each reference point, hops up, and hops left. The algorithm set the two points for the collar bone, calculated the hop distance based on a real measurement, and then saved that to the two nodes and propagated the information through the nodes in the array. The algorithm also took in the position of the belly button, and had the capability to take two more positions.

The simulation was tested using two different test beds. The first set was a theoretical set, which contained 100 nodes across by 136 nodes down. This is simply theoretical since 13,600 nodes on one shirt would just be too expensive, but it allows us to see the accuracy because nodes are 0.21 inches apart. The second set of tests was more realistic, however, it still enforced 546 nodes. It spread the nodes out to 1 inch apart, leaving 21 nodes wide by 26 nodes down.

Preliminary results from the simpler test bed can be seen in Figures 4, 5, and 6. This is the base data that the simulation dumps to disk, showing the distance each node is from a reference point.

7 Results

The results of this simulation deal solely with accuracy, and they are a bit surprising. The simulation shows that the algorithm incurs a large compounded cost for rounding when it selects the nodes for reference points. Figure 7 depicts this problem. For the theoretical test bed, when the collar bone points are selected, those points fall between nodes 27 and 28, and 73 and 74. The true hop distance for this test bed is 0.21 inches, however using this scheme and picking any combination of these nodes, we get hop distances ranging from 0.2127 inches, 0.2174 inches, and 0.2222 inches. Even these minute differences

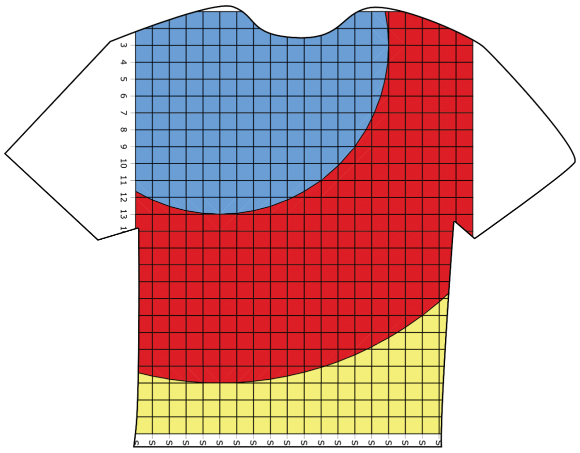

Figure 4: Simulation distance from the right collar bone to the rest of the body.

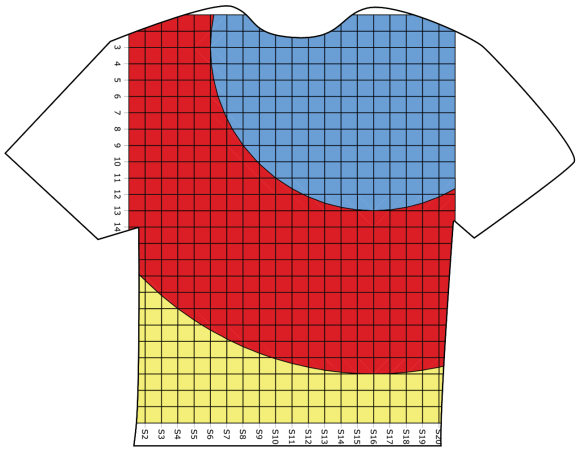

Figure 5: Simulation distance from the left collar bone to the rest of the body.

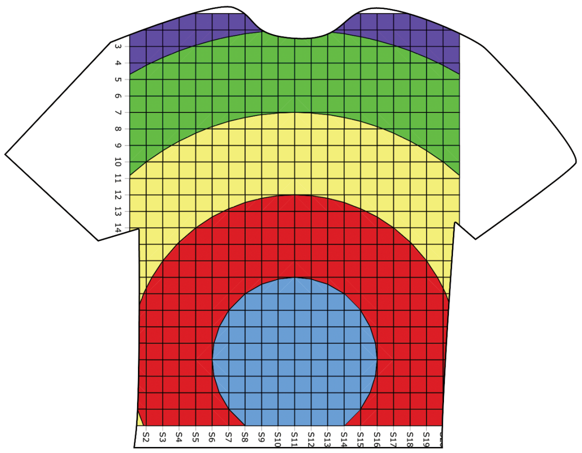

Figure 6: Simulation distance from the belly button to the rest of the body.

can incur a high cost if we must reference them from across the shirt. In fact, at the opposite end of the shirt, our data could be off by up to 2 inches, as shown in Figure 9.

The more realistic case of 1 inch hops can be much worse. If the collar bone falls on the middle of the hop, we must round up or round down. In this case, the collar bones fell at 15.5 and 5.5. General rounding schemes tell us to round up at 0.5, and in this case, that actually turns an accurate hop distance of 1 inch. However, if we round in or out, as Figure 8 shows, the distance we get to a reference point could be up to 4 inches off.

How can we deal with this compounded inaccuracy? The first scheme would be to add more reference points. Then, after calculating the distance between the collar bones and getting the hop distance, we can propagate that through the shirt and only give references to close reference points. That way, we would be referencing points that were 5-6, maybe 10 inches away, keeping the error down to an inch or less.

A second way to deal with this inaccuracy is to add more reference points and use them in the calculation of the distance. We can do this by selecting multiple 1-2 inch pairs to calculate the distance between and average, add a calculation between the hips, or use a point such as the belly button and discovering the distance of each side of the triangle. This gives a more accurate hop distance, but still leaves us with the inaccuracy of choosing the nodes.

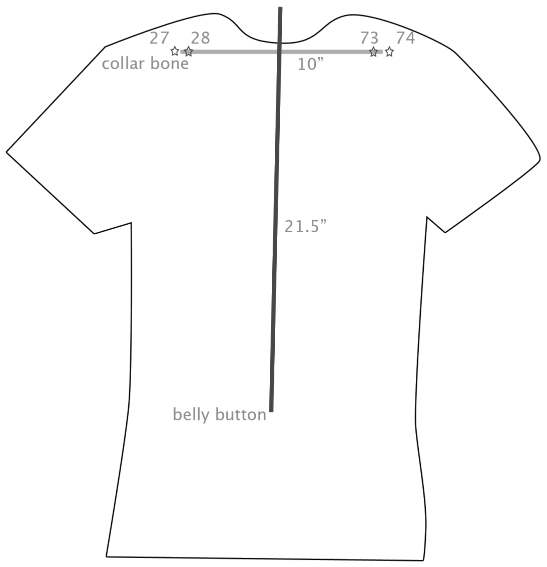

Figure 7: Actual collar bone points fall between two nodes on each end. The algorithm must decide which two nodes to use as the reference points, introducing some error.

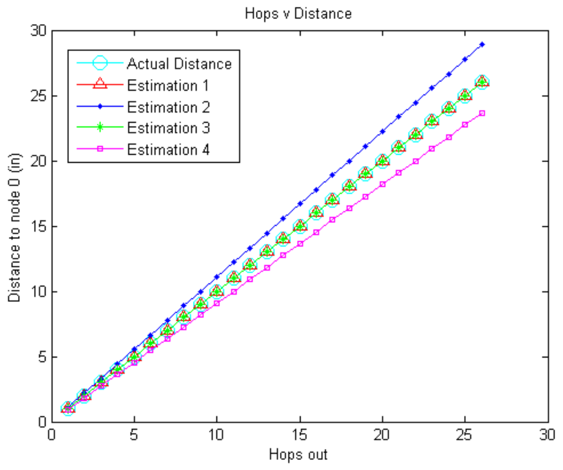

Figure 8: Distance calculated per hop with the different estimation schemes. The further away from the one point, the worse the distances are away from the actual calculated distance.

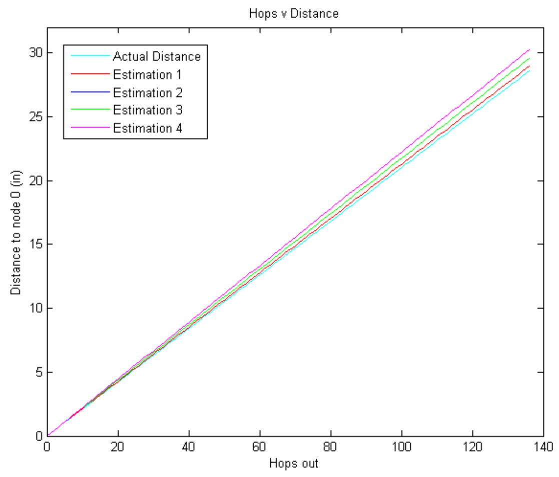

Figure 9: Distance calculated per hop with the different estimation schemes. The further away from the one point, the worse the distances are away from the actual calculated distance.

8 Future Work

This paper leaves a lot of open questions. Now that we have an algorithm, which while not completely accurate, is decent at small distances, we must also consider testing the power efficiency in the communication and the speed at which the algorithm sets up. There is a problem with testing this: the technology to create this kind of shirt is not readily available. I would propose that when the technology becomes available, or a more accurate simulation of the technology is available, to test this algorithm for speed and power consumption. It has a high cost of setup, especially in the means of communication, but it should prove very efficient because after the initial setup, no communication must ensue until a new wearer dons the shirt. The speed depends completely on the speed of the nodes' communication devices and the wire connecting them through the fabric of the shirt. All of these questions should be addressed when they can be given an accurate picture of the technology the nodes will be using.

9 Conclusion

The algorithm presented in this paper is one way to set up an electronic shirt to localize all its nodes to the position of the wearer underneath. It is inaccurate over large distances, but I would argue that for small distances, around 5 inches, it is accurate enough for most applications. Using directions (degrees), we can just go to the closest reference point, which should be no more than 4-5 inches away. If a doctor can be told where a soldier is bleeding on the battlefield in a general area, even if it is 1 inch off, it can be helpful in notifying the doctor and the severity and size of the wound, giving the doctor an estimate of how long the soldier can wait before the doctor arrives.

Therefore, this algorithm provides great potential for the future, and a good step towards localization of nodes in electronic shirts. It will be interesting to see how this step and future steps will be utilized once the technology becomes even more readily available to do computing and high-level sensing with sensors and computers woven into fabrics.

Bibliography

1. Chen, et al. "Development of a Scalable Healthcare Monitoring Platform." CIT 2004.

2. Cooper, Orth, and Post. "Fabric Computing Interfaces." CHI 98. April 1998.

3. Marculescu. "E-Textiles: Toward Computational Clothing." IEEE Pervasive Computing. 2003.

4. Martin, Jones, Edmison, and Shenoy. "Towards a design framework for wearable electronic textiles. "

5. Ottenbacher, et al. "Integration of a Bluetooth Based ECG System into Clothing." ISWC 2004.

6. Pentland. "Healthwear: Medical Technology Becomes Wearable." IEEE Pervasive Computing, May 2004.

7. Post and Orth. "Smart Fabric, or 'Wearable Clothing'." 1997.

Appendix

```
function disperse_info(pressure_info) {
    packet = (x-coord, y-coord, pressure_info)
    send(packet, PDA)
    wait(receive(data, PDA))
    my_information.add(data)

    hopsN = 0
    hopsE = 0
    packet2 = (hopsN, hopsE, data.hop_distance, data.node_name)
    send(packet2, N_connector)
    send(packet2, S_connector)
    send(packet2, E_connector)
    send(packet2, W_connector)
}
```

Figure 10: Pseudocode of the function to send, receive, and deal with data from the PDA

```
function pass_info(network_data) {
  if (network_data.input == S_connector) {
    receive(data, S_connector)
    data.hopsN++
    my_information.add(distance from me to data.node_name:
         hopsN up by hopsE right with hop dist data.hop_distance )

    send(data, N_connector)
    send(data, E_connector)
    send(data, W_connector)
  }
  if (network_data.input == N_connector) {
    receive(data, N_connector)
    data.hopsN--
    my_information.add(distance from me to data.node_name:
         hopsN up by hopsE right with hop dist data.hop_distance )

    send(data, S_connector)
    send(data, E_connector)
    send(data, W_connector)
  }
  if (network_data.input == W_connector) {
    receive(data, W_connector)
    data.hopsE++
    my_information.add(distance from me to data.node_name:
         hopsN up by hopsE right with hop dist data.hop_distance )

    send(data, E_connector)
  }
  if (network_data.input == E_connector) {
    receive(data, E_connector)
    data.hopsW++
    my_information.add(distance from me to data.node_name:
         hopsN up by hopsE right with hop dist data.hop_distance )

    send(data, W_connector)
  }
}
```

Figure 11: Pseudocode of function to send data throughout the network of sensors and update information.