**Context-Aware Computing for Bridge Inspection: NARRATIVE**

>> What? What is context-aware computing?

>> Why? Why do we need context-aware computing in bridge inspections?

* Where is context-aware computing being used in society?
* What improvements does context-aware computing bring to bridge inspection?

>> How? How is context-aware computing implement in bridge inspection?

* Explain the context-aware platform.
* Position and orientation being tracked 🡪 Converted to visual truncated cone 🡪 Identify elements in context 🡪 Streamline queries and pre-decide what data is preferential 🡪 Communicate bi-directionally based on streamlined data

Lock: Position is measured by GPS if above deck, non-GPS systems if below deck. Non-GPS positioning systems are based on a Cartesian coordinate system. However, GPS is NOT! Also, the elements are stored as point clouds (again a Cartesian coordinate system). It is easy to convert one Cartesian system to another but GPS to Cartesian conversion is tricky. One way to do it is Vincenty’s forward/backward pass algorithm but that requires that we know the GPS coordinates of the origin and two (three?) points on each axis. However, this might not be possible to collect. For example, in case the origin is inside the deck its not physically possible to measure its GPS coordinates. Or in case the points required to be noted are in a GPS denied environment, then its not possible to determine their coordinates.

Key: Improvised algorithm converts GPS to Cartesian coordinates based on *any* 4 calibration points whose GPS and Cartesian coordinates are knows. Empirical experiments conducted and uncertainty in the results of conversion algorithm shown to be comparable to 1) results of GPS data, 2) resolution of RTK-GPS and 3) insignificant to the resolution of the GPS systems proposed to be used on field.

>> Case Study: Telegraph Road Bridge

First issue: How do we store element data?

* Element list is based on MDOT’s standardized bridge inspection reports.
* List of elements classified as Deck Elements, Superstructure Elements, Sub-structure Elements and Approach Elements.
* Geometric data is stored as a point cloud data 🡪 FEM Point cloud or laser scanned point cloud data or photo scanned point cloud data 🡪 Only constraint is that 1) it should represent all elements in the element list and 2) it should be geometrically representative.
* Lock: How do we know which points belong to which element?
* Key: Mapping algorithm. Algorithm is based on AutoCAD dwgs of Telegraph Road Bridge where each element is drawn as a simple geometric shape (boxes, spheres, cylinders, triangular prisms). Information recorded as Element ID, Element Name, Element Shape, Bounding Parameters. Mapping algorithm iterates through the element list and assigns each element a sub-set of the point cloud by running containment test algorithm on the entire point cloud and the specific element geometry.

Second Step: Track inspector’s position and orientation

Inspector’s position is tracked by two different technologies – 1) GPS above deck and 2) UWB below deck. Orientation by magnetic tracker.

Third Step: Compute the parameters of the Visual Truncated Cone.

Fourth Step: On query, iterate through point cloud and find out which point is within spatial context and which element it corresponds to. Display list of elements contained in this list. Prioritize based on % of element contained.

Fifth Step: Use the prioritized list to set up required queries, pre-sort data, set up communication streams, etc. Explain concepts but implementation is to be done as Further Work.

Lock: Naturally, these position and orientation technologies are not 100% accurate. So, the accuracy of the algorithm actually identifying the objects in context is questionable. Is there some way to quantify this accuracy/uncertainty?

Key: Simulate a typical inspection routine. The routine has 128 typical inspection position and orientation combinations. For each position and orientation combination, we do one base case scenario run (without any position/orientation error) and 100 scenario runs (with the cases having position/orientation errors included based on sampling error from a typical GPS/UWB position error distribution). For each of the 100 runs, we compare the list of elements identified with the corresponding list of the elements identified in the base case. This gives us an idea of the average number of elements correctly identified, the average number of elements missed, the average number of elements wrongly identified for each of the 128 position-orientation combinations. We draw conclusions on the accuracy of the algorithm (for the GPS/UWB units being proposed) and the position-orientation combinations most susceptible to uncertainty in identifying the elements. We then have a qualitative (if possible quantative?) discussion on ideas to counter this error in identification by using higher accuracy instruments (downside is that the payload will be more) and by balancing the power structure between letting the context-aware computing algorithm identify the elements in context vs allowing the inspector a degree of authority to override the algorithm and pick alternative elements as in his/her context.

Finally, we discuss the pros, cons and concerns of using context-aware computing based inspection routines vs traditional fully manual inspection routines. We then draw conclusions on what/how to approach and investigate, in the near future, the appropriate components in the system to improve the accuracy of the context-aware computing algorithm and the positioning/orientation sensor combinations.