Sonardyne Interview Prep

DEMONSTRATE THE FOLLOWING:

Why do you want to work for us?

I really admire the history of Sonardyne, in 1971 when John Patridge decided he wanted to make underwater operations safer and more efficient I believe that was a life changing goal as well as motivational for me because the idea resonates with me. When I decided on an autonomous surveillance drone for my masters it was because I wanted to create something that would make lives better for people.

Something that I believe everyone should have access to because it has the potential to enable us to achieve more out of our lives with the help of technology. The ability to apply facial recognition on an autonomous drone can be applied to a lot of things. It could be hands free service that allows even incapacitated people to see things from a different perspective or do things they would not normally be able to do.

I remember when i was researching similar papers I saw potential use cases such as being able to aid blind people in walking by essentially acting as their eyes and relaying signals to them on directions that are safe to walk towards and when to wait. With the help of haptic feedback and or sounds.

Knowledge or Experience of:

SLAM / Path planning

* MSc using pose to get object location and waypoints to direct a Thorvald robot in a simulated grape field and enable autonomous navigation.
* Robot localisation (Odometry, Map based and GNSS)
  + Odometry – Dead reckoning (localisation based on the previous calculated pose and velocity estimation starting from 0, robot’s initial pose)
    - Odometry variants – Odometry can be visual, or laser-based . In visual odometry the velocity is estimates from the matching consecutive camera images (VPS on Tello drone which I have experience with).
    - All odometry solutions suffer from inherent drift problems. Statistical estimators like Kalman filters can reduce the drift but cannot completely eliminate it. Fusing information from multiple odometric sensors has a similar effect.
    - Kalman filter - a statistical estimator which filters the noisy measurement data, models measurement uncertainty and propagates that over time to exploit the knowledge about typical state changes (motion models.) It can fuse observations from multiple sources. It has a two step iterative procedure: predict where it propagates the state using a motion model and update: where it incorporates the measurements. The output is an averaged state together with its uncertainty.
    - Kalman Filter for robot localisation: The state corresponds to the robot’s pose. Observations are measurements from odometry, IMU, etc. Motion model is based on robot’s kinematics (e.g. differential drive)
  + Map based localisation – this uses a map of an environment as a references. The map can be an occupancy map (created with laser, 3D Cameras), visual landmarks or just geo-tagged images.

The sensor reading (laser, image) is then matched against the reference map: result is the most likely location which that reading originated from. An oppupancy map represents free and occupied parts of the environment and it is built from the 2D laser sensors, using SLAM (gmapping).

* Robot Navigation
* Can be split into Global planner, local planner and a recovery behaviour. A global planner is a type of high level navigation that finds the path from a start point to a goal using geometry. A local planner is a type of low level navigation that acts on the global plan and avoids obstacles using sensors(kinematics). A Recovery behaviour is only invoked if a robot gets stuck.
  + Global planner detailed (examples of global planner/navigation algorithms) - Computed before the robot starts moving Generates a series of waypoints for the local planner to follow.
  + Built-in types (BaseGlobalPlanner):
    - carrot\_planner - takes a goal point and attempts to move the robot as close to it as possible, even when that goal point is in an obstacle
    - navfn - uses a navigation function to compute a path for a robot
    - global\_planner - a fast, improved version of navfn
* Cost map: occupancy grid that represents safe places for a robot to be in. a global costmap is an amalgamation of a full static map + known obstacles.

Navigation algorithms

* A global planner, assuming a circular robot will operate on a grid costmap to find a minimum cost plan using a search algorithm. An example of a search algorithm would be Dijkstra which is utilised by thee navfn. The global planner is an improved version implementing A\* which incorporates heuristics.
* A\*
* Djikstra

Computer vision

* Image classification, pneumonia detection, skin lesion detection, face detection, object detection

Robotic related sensors and motion controllers

* Lidar and camera for computer vision on Thorvald and camera on tello drone, qibullet motion controller to program pepper bot to be able to point at the location of a card on a table based on a set of x,y coordinates from computer vision which are converted to grid format. Introduction to motion controllers in ROS

Software Development methodologies

* Agile – a software development methodology that is closely associated with object-oriented programming which utilises Languages such as Smalltalk, Lisp, Java and C#.
  + It is an iterative approach to project management and software development that helps teams deliver value to their customers faster. An agile team delivers work in small but consumable increments.
  + Scrum is an agile methodology that is adaptable. Fast, flexible and an effective agile framework designed to deliver value to the customer throughout the development of a project. The five principles of scrum are: Commitment, courage, focus, openness and respect.
  + Pair programming is an agile development technique originating from Extreme programming in which two developers team on one computer. The two people work together to design, code and test user stories. (A user story being an informal, general explanation of a software feature written from the perspective of the end user or customer. The purpose of a user story is to articulate how a piece of work will deliver a particular value back to the customer.) A common implementation of pair programming calls the programmer at the keyboard the driver, while the other is called the navigator. The navigator focuses on the overall direction of the programming. The collaboration between developers can be done in person or remotely.
* Spiral - a systems development lifecycle method used for risk management that combines the iterative development process model with elements of the Waterfall model. The spiral model is often used by software engineers and is favoured for large, expensive and complicated projects. Examples of spiral usage can be seen in Microsoft that used it to develop the early versions of Windows. Game development is another industry that uses spiral model to develop games.

Waterfall – a software development methodology that is a sequential development process that flows like a waterfall through all phases of a project. (Requirements, system design, implementation, testing, delivery/ development and Maintenance) with each phase completely wrapping up before the next phase begins. Can be slow and costly due to tis rigid structure and tight controls.

Devops

A mix of a development methodology and a set of practices that support an organisational culture. DevOps deployment centres on organisational change that enhances collaboration between departments responsible for different segments of the development life cycle, such as development, quality assurance and operations.

Key phases being:

* Plan
* Code
* Build
* Test
* Release
* Deploy
* Operate
* Monitor
* plan

Diagram

Description automatically generated

Pros:

* Is focused on improving time to market.
* lowering the failure rates of new releases
* shortening lead time between fixes
* minimizing disruption while maximising reliability and uptime.

To achieve this DevOps organisations aims to automate continuous deployment to ensure everything happens smooth and reliably. Cons:

Some customers don’t want continuous updates to their system

Some industries have regulations that require extensive testing before a project can move onto the operations phase. If different department use different environments, undetected issues can slip into production. Some quality attributes require human interaction, which slows down the delivery pipeline

Some quality attributes require r

Id fieereent departments

**Unit testing** - Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation.

**Integration testing** - Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. Integration testing is conducted to evaluate the compliance of a system or component with specified functional requirements.

**System testing** - System testing, also referred to as system-level testing or system integration testing, is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application.

**Acceptance testing** - Acceptance testing is a quality assurance (QA) process that determines to what degree an application meets end users' approval. Depending on the organization, acceptance testing might take the form of beta testing, application testing, field testing or end-user testing

**Black box texting** - A method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance.

**White box testing** - White box testing is a form of application testing that provides the tester with complete knowledge of the application being tested, including access to source code and design documents. This in-depth visibility makes it possible for white box testing to identify issues that are invisible to grey and black box testing.

**Field testing** – Field testing is a product development process for collecting usage data from potential customers before launch. This test type focuses on the unguided, natural, contextual usage of a product.

**End user testing/ User acceptance testing (UAT)** - User acceptance testing (UAT), also called application testing or end-user testing, is a phase of software development in which the software is tested in the real world by its intended audience.

**Object oriented vs Procedural Programming vs Functional**

**Procedural programming** is the use of code in a stepwise procedure to develop applications.

Text

Description automatically generated

For example, to develop a simple Bank Account App procedurally:

* Creating an account for an individual (account)
* Getting an account to deposit or withdraw funds (getAccount, deposit, withdraw)
* Transferring funds between two different accounts (transfer)
* Recording all changes that occur with all accounts (accounts)

**Object oriented programming** is the use of self-contained code objects to develop applications. In JavaScript, this can be achieved by creating a blueprint (class) for manufacturing objects.

Text

Description automatically generated

A **class** (BankAccount) encapsulates a set of properties (**constructor function**) and behavior (**class functions**deposit**,**withdraw**, and**transfer) that can be used to instantiate an **object**of specific values (i.e: let john = new BankAccount(“John”)). This is typically used to model real-world objects.

**Functional Programming** is the use of **pure** high-order functions to develop applications. This involves a focus on creating code that avoids changing state and mutating data. JavaScript methods such as map, filter, find, and reduce are built such that any data received by a program, functionally coded, will not mutate the original data. Also, the use of **closures**&**currying** makes functional programs simpler to implement and easier to read.

Text

Description automatically generated

Here, the data (origin) is not mutated during the creation of new accounts. Rather, a copy of it is created (slice) and merged with a new account (**spread operator**) into a new array (accounts). Accessing an account is also done without mutation (filter). Performance of transactions are done by creating an updated copy (map of accounts).

Unlike OOP, in functional programming there are no objects whose states are continually being updated (i.e.: no John or Joe objects). Rather, copies of an account or the list of accounts are copied, updated, and returned.