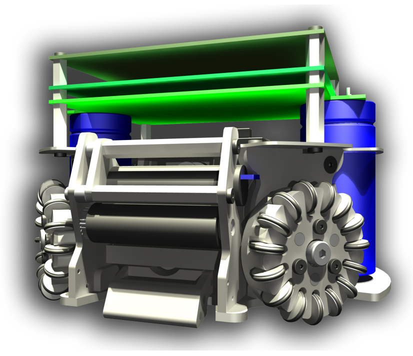
RoboJackets 2010 Team Description Paper

Alex Cunningham *, Andrew Bardagjy, Stefan Posey, Ben Johnson, Stuart Donnan, Philip Rogers, and Stoian Borissov
* primary contact: alexgc@gatech.edu

RoboJackets - www.robojackets.org
Georgia Institute of Technology - Atlanta, Georgia, USA.



Abstract. For the 2010 RoboCup SSL season, Georgia Tech RoboJackets RoboCup SSL team has redesigned much of the system from our previous year of international competition (2008) with both a significant software upgrade and a new fleet of robots. Software has improved dramatically to incorporate a robust behavior-driven play system and an optimal pass planning system. The 2010 robot fleet includes many incremental improvements to address deficiencies in the previous design, as well as the addition of a chipper and wheel encoders. This document describes our overall system, with a focus on the improved software system and new mechanical design.

1 System and Team Overview

The robotic system is composed of three main subsystems: electrical, mechanical, and software. Each component has a small team of individuals working under a leader; the system leaders and the team leader communicate their findings and progress to one another to ensure a successful system. Each team sets its own goals and priorities with the team leader overseeing the entire process. As the team grows, this hierarchy has allowed the team to react accordingly. All team members are encouraged to participate at their own pace. Additionally, some effort is made to recruit and train new members to ensure the team's future.

2 Software

The new version of the software adds new passing capabilities using an optimization-based approach. The previous system supported moving, shooting, and could play the rules of soccer, but did not handle multirobot activity, such as passing, in a reliable manner. This year, as in the previous year, we use the shared vision system for field sensing using standardized patterns. For 2010, we will extend the current system to implement a robust offensive passing system where the robots acquire ball control and make a coordinated series of passes until it can shoot on goal. We employ a multi-stage planner that creates initial plans with an *analytical planner* and then performs *nonlinear constrained optimization* over the positions of robots to create an improved plan that mixed low-level controllers can execute.

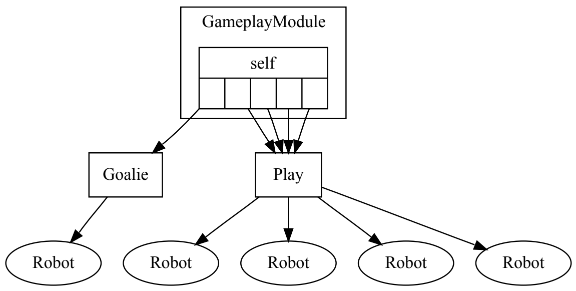


Fig. 1: A tree diagram showing a play structure of robots, where the five robots on the team are divided into single goalie, and four field robots executing under a single play.

The underlying structure of the software system is a streamlined variation on the *Skills, Tactics and Plays* (STP) framework, with more general plays and behaviors, in which a "play" is a top-level structure that the strategic-level gameplay module can switch between depending on the state of the game, and "behaviors," which encapsulate the functionality of either one robot or a set of robots at once. A sample partitionoing of robots into behaviors by a top-level play is shown in Figure 1. This accomplishes the same goals as the STP framework, but without strict divisions between "Tactics' and "Skills." During the last competition GT RoboCup SSL participated in, the system provided sufficient means to manage a variety of scenarios required by the rules, but was less robust in general fluid gameplay due to the inability to robustly collaborate between robots. The offensive system at the time consisted of a pair of robots that chase the ball around the field and try to make shots. This system, while simple, is effective against weaker opponents, but proved no match for teams with robust passing systems.

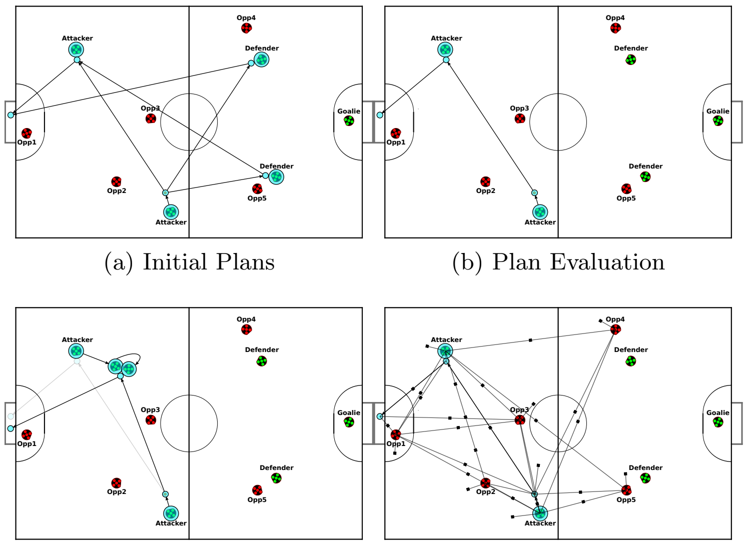
There are a variety of means of approaching optimization in the context of high-dimensional systems, and this paper will focus on the relationship between optimizing robot trajectories in an optimal controls sense with recovering robot trajectories from measurements taken on the environment. The *Simultaneous Localization and Mapping* (SLAM) problem is a canonical problem in the field, where an optimization algorithm must find the most likely position of a robot in an environment based on a series of measurements of the environment. Much of the work in the field has focused on approaching this problem using Kalman or particle filters [1–3] to represent the state of the system, however we will focus on solutions that provide the entire trajectory of the robot, otherwise known as the complete SLAM problem.

The basic *Smoothing and Mapping* (SAM) approach [4] to the problem uses the information matrix to encode the relationship between the variables, which is as an exact dual to the covariance matrix used in Kalman filters, but the key in the SAM approach is that we do not need to keep keep the whole information matrix, rather, we can store the square root of the information matrix, which remains sparse during the course of optimization, a feature that we can exploit to perform much more efficient optimization. The resulting operation used in solving SLAM problems becomes a matter of developing fast, multi-variable optimization algorithms, and there has been much work in this area to improve the underlying optimization techniques, particularly from graph theory [5–7]. In particular, there are methods for incremental SAM [8], which are especially useful when approaching the development of real-time systems.

In order to perform robust, efficient passing, we have decided on a particular structure for the offense play. The basic concept is that if there is a shot available or a sequence of passes to a shot (henceforth known as a "shooting solution"), then the system will construct the sequence of paths for robots and kicks, run a nonlinear optimization technique to improve the path and ensure viability, and then execute it.

The resulting system structure for the offensive passing algorithm has the following structure, shown in Figure 2.

1. **Analytic Planner** Given the positions of all the robots and the ball, it is possible to determine if there is a shot solution available with a simple weighted graph search. This can be a simple heuristic for shot viability, or a more sophisticated system as necessary.
2. **Plan Evaluation** We take a set of plans and sort by quality on a number of metrics to create a good initial estimate for a plan.
3. **Optimization** With an initial estimate of a plan, the optimization engine will be able to create a graphical model of the plan and optimize for an optimal sequence of actions. The particular optimization engine is a nonlinear constrained optimization algorithm primarily designed for Simultaneous Localization and Mapping, but we will exploit the duality between this planning problem (create an optimal path from constraints) and SLAM (recover a previously traveled path given measurements).
4. **Execution** Given a viable plan, we choose among a variety of motion planning and behavioral approaches can be used to execute the actions parameterized by the plan.



(a) Initial Plans (b) Plan Evaluation

(c) Optimization (d) Result

Fig. 2: The plan optimization process, starting with a rough evaluation of plans and then optimization of a small set to generate a single optimized plan.

The optimization algorithm comes from the GTSAM library [9] developed in the Georgia Tech Robotics and Intelligent Machines (RIM) center with Frank Dellaert, and derives from Sequential Quadratic Programming. This algorithm is robust to both nonlinear systems and constraints, and has been reformulated to work in a graphical model environment, which allows for an intuitive construction of stochastic switched hybrid systems.

2.1 Analytic Planning

The analytic planner creates an initial set of pass plans, which consist of a sequence of passes ending with a shot on goal given the robots on the field. We do a search over these possible plans and construct a cost function for pass solutions that penalizes passes that are too long or that are clearly blocked by either an opponent or another teammate. Special care is taken to not penalize solutions that have a possibility of being good passes at future levels of the planner. Using this cost initial function, we sort possible passes and select a subset of the best passes to send to the optimization phase.

The cost function used resembles the evaluation step in the STP planner, though it differs because our pass structure is created before any robot has the ball, and includes all possible passes, allowing us to optimize the passer and the receivers, as well as the ball path.

2.2 Optimization

Once a plan has been selected for further evaluation, we use a constrained nonlinear optimization approach to find an improved solution. A key observation in the development of the optimization technique is the dual nature between optimizing a robot control system and SAM. In the SAM problem, the basic goal is to recover a robot trajectory based on a set of constraints from the robot's sensor systems and motion models after the robot has moved through the environment. In the controls case, the goal is to determine an optimal trajectory based on a set of constraints, such as those on robot motion or obstacle avoidance. In both cases, the goal is to recover a robot trajectory from a series of constraints, and they primarily differ in what sort of constraints are applied. Rather than using the formulation of optimal control, we will use a graphical model approach to assemble a full cost function over the environment.

There are two steps to effective optimization of a soccer plan:

– *Factor Graph Construction*: we assemble a cost function to optimize using a piecewise factor graph approach usually used in SAM solutions. This allows development by means of combining a large number of constraint functions that individually relate only a small number of variables.
– *Nonlinear Optimization*: after the general cost function is assembled, we use Sequential Quadratic Programming (SQP) to reduce the a system with an arbitrary smooth nonlinear cost function and constraint set to a quadratic programming problem suitable for solving via Multi-frontal QR factorization.

The formulation of actual factors in this approach, as it derives from the need to define an error function in SAM, has the form of a function $h(x)$ dependent on the state of the system and a measurement z, where in the SAM case, the $h(x)$ is the generative measurement model and the z is the actual value of the system, but in our system, $h(x)$ describes the current state with relation to a cost function, and z is the optimal value. Expressed in general form, these factors combine as in (1) to form a nonlinear least squares problem, where x forms the set of state variables, z_k is the optimal value for the constraint, and R_k is a weighting matrix on the cost function.

$$x^* = argmin(\frac{1}{2}\sum_k \|h_k(x_k) - z_k\|^2_{R_k}) \tag{1}$$

With this error-function approach, especially as it allows us to place relationships between variables in the system, we can build the cost function to define an optimal solution. One of the basic cases where we assemble these cost functions is to optimize for shorter passes between robots, as well as smaller robot movements, which we express using an optimal distance where $z = 0$, and $h(x, y) = \|x - y\|$. By connecting all of nodes with constraints such as these, we can simultaneously optimize the movement of robots and passes to find a series of movements and passes that minimizes all of the distances. In this unconstrained optimization case, we can approach the actual optimization using nonlinear unconstrained optimization techniques such as Levenberg-Marquardt or Gauss-Newton. Because of the least-squares formulation of a quadratic cost function, we can approximate the Hessian of the overall cost function using only the Jacobian of $h(x)$. The factor can be approximated through a linearization of the measurement function, which produces the Quadratic Programming problem shown in (2).

$$x^* = argmin(\frac{1}{2}\sum_k \|H_{x_k}x + h(x_{k_0}) - z\|^2_{R_k}) \tag{2}$$

Some other cases where factors define a means to improve a plan:

– Maximizing the distance of pass trajectories from other robots to prevent pass interception.
– Maximizing the distance between robot paths and opponent robots to avoid collisions and improve movement speed.
– Minimizing the angle between the facing of a robot upon receiving a pass and making the next pass so as to minimize the time necessary aiming.

While this approach works for basic scenarios, such as making the robots move and pass the smallest possible distance, we also need to express hard constraints on the motion and passing of robots, such as staying on the field and not driving through other robots. These constraints are not just cost functions to be minimized, but rather requirements that must be met for any solution to be feasible. We can extend the previous expression to allow the incorporation of these hard constraints by redefining the optimization problem as a Lagrangian optimization problem, as in (3), where we minimize the cost function over the states in the system and the Lagrange multiplier cofactors λ.

$$x^* = argmin(\frac{1}{2}\sum_k \|h_k(x_k) - z_k\|^2_{R_k} - \lambda^T g(x)) \tag{3}$$

We can express the constraints in the form of $g(x)$, where $g(x)$ defines a vector-valued function where the constraints are considered fulfilled if $g(x) \leq 0$, which allows both equality constraints, such as ensuring passes arrive at the same time and location as the robot receiving the pass, or inequality constraints, such as maintaining bounds on the positions, velocities and acceleration of the robots to keep them on the field and driving with viable commands. The addition of these constraints requires a more sophisticated solution method, and in our case, we use Sequential Quadratic Programming (SQP) to reduce the full nonlinear constrained problem into a series of quadratic programming subproblems with linear constraints, which can then be solved with a mixture of direct elimination of the constrained variables, and unconstrained optimization on the remaining variables [10].

2.3 Control

In order translate the plans to robots in the actual system, we have a number of low-level controllers that execute commands. We can switch between various control approaches for particular applications, which is necessary to avoid using suboptimal performance in some situations. There are several means of issuing commands to the robots to execute movements across the field:

RRT Planning For situations when we need a robot to move around other robots, we can use a simple RRT-based system to find a path. This does, however, have the problem that the resulting path is unpredictable, so it cannot be used well with optimization-based approaches.

Direct Path Commands We can instruct a robot to execute a specific path, which allows for more control by the higher-level modules in the system.

The low-level path execution uses a controller that drives the robots to their destinations as fast as possible with the ideal trapezoidal velocity profile. Upon starting movement, the robot begins at maximum acceleration until it reaches maximum velocity, maintains the maximum velocity while following the path, and then slows at its maximum deceleration rate to stop at its goal. The bounds used in this model derive from empirical tests of robot performance.

3 Mechanical

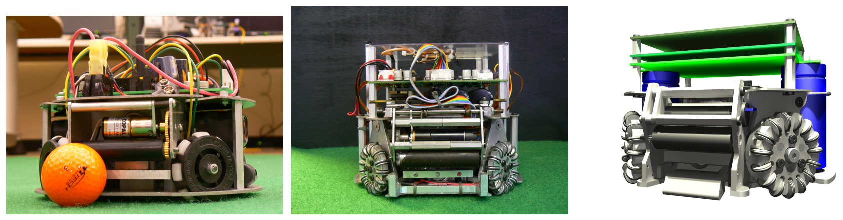


Fig. 3: 2007 Robot (left) vs. 2008 Robot (center) vs. 2010 Robot CAD Rendering (right)

While our previous robots participated successfully in spirited competition, we are developing a new team with increased capabilities for the 2010 season. In particular, the 2010 robots boast a chipper and wheel encoders. Lessons learned in competition and testing of the 2008 fleet motivated us to place special emphasis on increased manufacturability and reliability. Testing and validation of new components uses the previous systems as benchmarks for comparison and follows similar methods as those outlined in our 2008 Team Description Paper [11]. The new team will be no larger than 179mm in diameter and 149mm tall.

3.1 Reliability, Manufacturability, and Maintainability

A number of improvements are planned to enhance manufacturability, reliability, and maintainability of the 2010 team. In 2008, due to extreme cost cutting, set screws were used, throughout the vehicles to attach gears to shafts.

As expected, this became an unfortunate reliability concern. In the 2010 design, custom pinion shafts are used eliminating the need for set screws. In addition, the use of pinion shafts makes the drive train more compact leaving more room for ball manipulation apparatus.

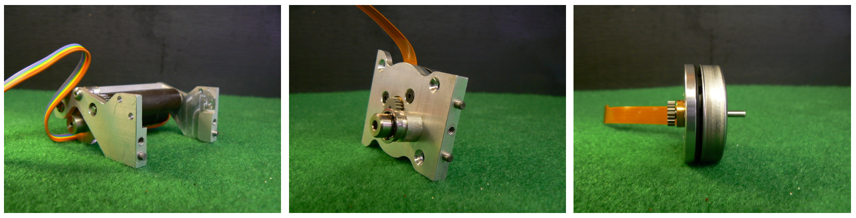


Fig. 4: From right to left, (a) dribbler, (b) drive module, and (c) modified drive motor

To further increase the reliability and maintainability of the new team, the drive modules are now held in place with two locating pins and a single screw on the bottom and one screw on the top. The dribbler is attached to the bottom plate with a similar screw and locating pin configuration. Both assemblies are shown in Figure 4. These changes have decreased the number of screws in each robot by over half and have increased the locational accuracy of all components.

3.2 Omni Wheel

Our latest designs address a few key problems with the old omniwheel design. The previous designs suffered from excessive carpet fiber buildup which could be only mitigated by a complex cleaning procedure. In addition, refinements were made to accommodate the new drive modules.

To address carpet buildup, the edges of the rollers are now tapered. Additionally, the formerly sharp edges of the wheel bodies were filleted to provide a smoother interaction with the carpet. To simplify cleaning procedures the new fleet uses a small dowel pin as an axel for each roller. It should also be noted that the new rollers are compatible with both 2008 and 2010 wheel bodies.

3.3 Drive Module

To make room for the chipper and wheel encoders, the drive modules had to be completely redesigned. Each assembly is composed of a single plate which supports both the motor and wheel and a plate to support the encoder. Just as in previous years, we use Maxon EC45 pancake brushless motors. The gear ratio was decreased slightly from 5:1 to 9:2 allowing for a slight increase in speed. In addition, the gear teeth were made much larger making the drive train less susceptible to fibers and and other foreign matter. This design change is achieved through an internal gear mounted to the rear of each omni wheel which mates with a custom pinion shaft installed in the motors. This shaft repaces the stock shaft and protrudes out the rear of the motor connecting to an encoder. A modified drive motor is shown in Figure 4.

3.4 Dribbler

The dribbler is the assembly which controls the ball during gameplay. The mechanism utilizes a custom steel pinion shaft covered with silicone rubber tubing to increase adhesion. Just as in 2008, a Maxon EC16 brushless motor mated to a GP16A planetary gearhead spins the dribbler shaft through a simple 1:1.4 gearbox. The use of ball bearings rather than bushings increase efficiency and a larger tooth size decreases susceptibility to foreign object contamination. Additionally, the assembly incorporates a break-beam ball sensor. The dribbler's ball coverage is no more than 19%.

3.5 Kicker

The robot's kicker is the primary method of both scoring and passing. A solenoid is mounted inside the robot and a large amount of current is discharged into it from a capacitor bank. In order to accommodate a more powerful solenoid, large off-the-shelf solenoids have been purchased and have had their casing removed to fit dimensional restrictions. The changes will allow for more powerful kicks while still allowing the kicker system to fit within the same size restrictions. A more robust kicker boot has also been implemented which, unlike the 2008 kicker boot, is designed so that it does not bend under the large forces felt while kicking the ball. The kicker plunger will be made of a front aluminium component and a rear steel component. The use of aluminium reduces energy losses due to the solenoid's magnetic field pulling back on the plunger after it has been fired.

3.6 Chipper

The chipper allows the robots to pass the ball by chipping it into the air. It can shoot the ball over opponents and adds greater flexibility to planning algorithms while reducing chances of interception. The chipper boot is located underneath the kicker boot. When a robot is in possession of the ball, the dribbler rolls the ball up against the chipper boot. The chipper is powered by a solenoid which is mounted above the kicker solenoid. In order to transfer from the solenoid to the chipper, two arms are used that act like levers. When the solenoid fires and pulls the plunger back, the arms rotate and move the chipper boot forward, chipping the ball up. Currently two different chipper systems are being tested. The results will help determine the correct arm length, solenoid power, chipping angle. To reduce mechanical losses from friction, future chipper system designs will implement pancake solenoids which will fire in the same plane as the chipper boot, removing the need for any mechanical interface.

4 Electrical

The electronics system is broken into the following two broad subsystems:

Controller Circuitry contains the radio link to the computers on the sidelines, and all motor control functionality.
Kicker Circuitry drives the kicker and chipper solenoids.

In comparison to the previous design revision, we added a chipper solenoid, but the charger and other switching mechanisms have remained largely the same.

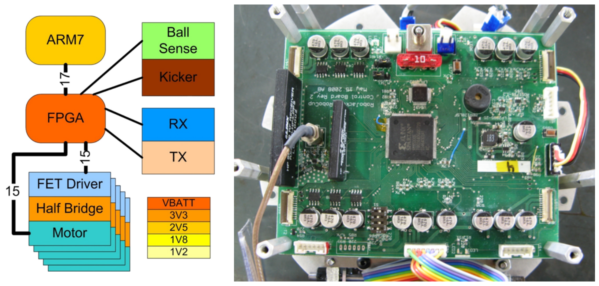


Fig. 5: Block diagram for the controller board (left), and a photograph of the actual board.

4.1 Controller

The primary functionality of the controller subsystem is to translate motor speeds sent by the computer on the sidelines to actual motor values. Though the controller is currently "dumb" we hope in the future to enable more intelligence on the robot.

The requirements of the electrical system derive from the requirements of the drive and dribbler motors. Each motor has three phases connected in a wye configuration and three hall effect sensors to establish rotor position. To drive a brushless motor the rotor position is used to determine which coils should be energized. A coil is energized with a half bridge. Each half bridge is composed of an N and P channel FET driven by a Microchip FET driver (TC4428). There are three half bridges per motor (one for each coil) for a total of 15 half bridges per robot. With miscellaneous passives, the motor driver circuitry composes about 150 components on each board. The half bridges are driven by a Xilinx 100K gate Spartan 3E (XC3S100E) FPGA. The FPGA is memory mapped to a NXP ARM7 (LPC2103) which handles local feedback control with information from US Digital encoders.

The robot communicates to the server via Texas Instruments CC1101 wireless ASSC. This ASSC allows tuning from 779MHz to 928MHz including the 868MHz

ISM band. Packets from the radio modules are routed through the FPGA to the ARM to allow for future work in hardware accelerated wireless protocol research. Due to poor wireless performance in the previous year, and severe space constraints in the current design, standard monopole antennas were not used. Instead, a balanced dipole "halo" antenna will be utilized. The antenna is ideally suited for the challenge as it is very low profile and omnidirectional [12].

In a typical design, power supplies and their distribution are normally considered a trivial implementation task. In contrast, this design has five power rails; 1V2, 1V8, 2V5, 3V3, and VBATT (12V). The many power rails present a considerable routing challenge. Despite this, the board is only two layers which affords much quicker and cheaper manufacturing then other processes. The 3V3 power supply is switching for high efficiency while the other lower voltages are produced with simple linear low dropout regulators.

4.2 Kicker

The kicker circuitry charges a bank of capacitors which are then discharged into the solenoids (both the kicker and the chipper) to kick the ball. The kicker uses a Linear Technologies LT3750 flyback controller to convert the battery voltage (12V) to approximately 250 volts which charges a 5400μF capacitor bank. The capacitors are discharged into the solenoids by means of an Insulated Gate Bipolar Transistor (IGBT). Ball speeds approaching legal limit have been achieved. In comparison to other designs, this design is quite compact, requiring only about three square inches of board area. This design also boasts efficiencies exceeding 80%.

5 Conclusion

For the 2010 season, we intend to have a new fleet of robots incorporating the lessons learned through the last design revision, as well as an improved robust passing and control system. This should allow both significantly faster and more competitive gameplay, as well as more reliable robots with fewer maintenance requirements.

References

1. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. The MIT press, Cambridge, MA (2005)
2. Thrun, S., Liu, Y., Koller, D., Ng, A., Ghahramani, Z., Durrant-Whyte, H.: Simultaneous localization and mapping with sparse extended information filters. Intl. J. of Robotics Research **23**(7-8) (2004) 693–716
3. Dellaert, F.: An intuitive introduction to ML and MAP estimation, Kalman and particle filters, and Rao-Blackwellization. In preparation (2003)
4. Dellaert, F.: Square Root SAM: Simultaneous localization and mapping via square root information smoothing. Technical Report GIT-GVU-05-11, Georgia Institute of Technology (2005)

5. Triggs, B., McLauchlan, P., Hartley, R., Fitzgibbon, A.: Bundle adjustment – a modern synthesis. In Triggs, W., Zisserman, A., Szeliski, R., eds.: Vision Algorithms: Theory and Practice. LNCS, Springer Verlag (Sep 1999) 298–375
6. Bertele, U., Brioschi, F.: On the theory of the elimination process. J. Math. Anal. Appl. **35**(1) (July 1972) 48–57
7. Bertele, U., Brioschi, F.: Nonserial Dynamic Programming. Academic Press (1972)
8. Dellaert, F., Kaess, M.: Square Root SAM: Simultaneous localization and mapping via square root information smoothing. Intl. J. of Robotics Research **25**(12) (Dec 2006) 1181–1203
9. Dellaert, F.: Square Root SAM: Simultaneous location and mapping via square root information smoothing. In: RSS. (2005)
10. Fletcher, R.: Practical methods of optimization; (2nd ed.). Wiley-Interscience, New York, NY, USA (1987)
11. Team, R.R.S.: Team description paper. Technical report, Georgia Institute of Technology (2008)
12. Harrison Jr, C., King, R.: Folded dipoles and loops. Antennas and Propagation, IEEE Transactions on [legacy, pre-1988] **9**(2) (1961) 171–187