Using the Field Edge as a Localization Landmark in the Robot Soccer

Anastasia Bolotnikova
Advanced Algorithmics project by MSc student of Computer Science Curriculum, University of Tartu, Estonia, 2016

Introduction
In this project the focus is on the RoboCup SPL robot soccer software developed for the NAO robots. Current system utilizes the goal posts and the center circle as the main landmarks for the localization module. The problem occurs when those landmarks are not seen or detected by a robot. The aim of this project is to introduce the field edge as a potential new additional landmark for the localization of the robot on the field and answer the question “Can the field edge be efficiently used as a localization landmark?”

Calculating the distance to the Field Edge
First of all, field edge has to be detected in the camera frame. Secondly, detected edge has to be translated to the robot relative (RR) coordinate system, so that the distance to it can be calculated.

In this part of the project by analyzing the algorithms and conducting some experiments we answer the following questions: how fast can the edge be detected and how accurately can we calculate the distance to it.

Field Edge detection.
The camera frames are scanned column by column from the top until the point of significant patch of green is found and saved. As a result for each column we have a potential of field edge. In order to combine all the points into one line equation, the RANSAC algorithm is applied to the points. As a result, the line is fit to the data points in such a way that the number of points close to the line is maximized. This way the outliers, caused by noise and objects on the field, do not introduce big error to the line equation.

In this project an optimisation of the current field edge detection algorithm is proposed and studied. Namely, we propose to decrease the amount of sample points which are passed to the RANSAC algorithm in order to speed it up. However, this modification may have a negative effect on the field edge detection ratio (edge not detected) and accuracy (edge detected wrongly). In order to find out how much faster the optimised field edge algorithm will work and what will be the effect on detection ratio and accuracy, several experiments were conducted. The robot have been put on the centre of the filed where it was looking at the field edge. Common noise was introduced to the scene observed by robot, namely, goal, a robot and a ball. The variable controlling the number of scanned columns was changed in different experiments. Average time elapsed during the field edge detection and the number of correctly detected edges in 3 minutes were calculated. The result of the experiments can be seen at the right top figure.

Distance calculation.
In order to calculate the distance to the field edge line obtained in the previous step needs to be converted from the camera coordinates to the RR coordinate system. The transformation matrix for this conversion is put together taking into account robot’s height and the current pose (joint values), calibration values for camera offsets. As a result, we obtain the line of form $t^T x + t^T y + t_z = 0$ which represents a line on the ground plane in RR coordinate system. In order to calculate the distance from robot to the field edge line, we use the following formula $d = \sqrt{t^2 + t^2 + t^2}$. Several experiments have been conducted in order to evaluate how accurate is the value of the calculated distance. The result of experiments is shown at right bottom figure. The calculated distance from detected field edge is average value over 124 trials.

How fast can the algorithm be

<table>
<thead>
<tr>
<th># scanned columns</th>
<th>Correctly detected</th>
<th>Time elapsed (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>every column</td>
<td>20,82 edges/sec</td>
<td>6049.43</td>
</tr>
<tr>
<td>every other column</td>
<td>27,72 edges/sec</td>
<td>6444.44</td>
</tr>
<tr>
<td>every 3rd column</td>
<td>25,94 edges/sec</td>
<td>6260.94</td>
</tr>
<tr>
<td>every 4th column</td>
<td>29,70 edges/sec</td>
<td>5924.41</td>
</tr>
<tr>
<td>every 5th column</td>
<td>20,44 edges/sec</td>
<td>6010.19</td>
</tr>
<tr>
<td>every 10th column</td>
<td>15,16 edges/sec</td>
<td>5262.29</td>
</tr>
</tbody>
</table>

How accurate is the distance calculation

Position estimation

As a result of this project work following conclusion can be made:

• The number of field edge points in detection algorithm can be reduced by factor of 4, which speeds up the algorithm by ≈ 2ms.
• The distance calculation to the detected field edge is quite accurate, but starts to decrease significantly when distance is greater than 5m.
• Field Edge can be efficiently used in localization in some situations, such as when 2 field edges are detected (<5m) and there is a previous position estimate.

Conclusion

In this project work following conclusion can be made:

• The number of field edge points in detection algorithm can be reduced by factor of 4, which speeds up the algorithm by ≈ 2ms.
• The distance calculation to the detected field edge is quite accurate, but starts to decrease significantly when distance is greater than 5m.
• Field Edge can be efficiently used in localization in some situations, such as when 2 field edges are detected (<5m) and there is a previous position estimate.

Info
More information at icv.ims.ut.ee/FELL-RR
UT RoboCup Team Philosopher: icv.ims.ut.ee/philosopher
Contact: anastasia.bolotnikova@ut.ee
PROBLEM
In December 2015, there was a huge natural disaster happened in Tamil Nadu, India. The major cities like Chennai and Cuddalore were attacked by floods and hurricanes. The cities were almost immersed in water and many people needed food, shelter, clothes to survive. Some people were willing to help but they couldn’t able to find and communicate with the affected people. So, the “Communication” was the main problem here.

SOLUTION
Being a Software Engineering student in Estonia, I had developed a web application in 30 continous hours which will:
- help to connect the flood affected people with the donor who wish to help.
- help the affected people to search and track the donors nearby.
- help donors to retrieve the list of requests from affected people.

WORKING
1. I wish to contribute:
   Firstly, the donors have to register themselves with the things that they wish to donate. Here, the donors can also see the list of need requests from the affected people.

2. I need help:
   Secondly, the affected people have to search for donors. In the shown list of donors, select donor and make request. Then, donor can see the request in their list.

FUTURE GOAL
In less than a month, this app has been used by many people. The future goal is to add new features like calling and messaging in this application. This application will be extended to world-wide in the near future.

Note: The above mentioned website’s search is restricted to places in India only and result will be the donors list within 20 kms of the searched place.
Parameter Optimisation for Brain-Computer Interface

INTRODUCTION

This project laid the foundation on automatically finding good parameters for a visual evoked potential based brain-computer interface (BCI). The BCI for which the parameter optimisation was implemented is author’s previous work and its main weakness is that it has not been thoroughly tested. This project aims to make testing of the BCI easier by automating the parameter finding step and hopefully helps to reveal the true potential of the BCI.

In the course of the project several improvements were made to the BCI. Most notably, the false positives filtering was improved and more flexibility was added for choosing options for different feature extraction methods. Furthermore, now the BCI can be used to control Parrot MiniDrone Jumping Sumo.

The main idea of the BCI is that user looks at one of four blinking targets on the screen and the BCI tries to identify, which target is the user looking. Overview of the BCI pipeline can be seen below.

BCI PIPELINE STEPS

1. EEG device Emotiv EPOC measures user’s brain activity.
2. Digital signal processing techniques are applied to prepare the signal for feature extraction.
3. Power spectral density and canonical correlation analysis feature extraction methods are used.
4. The final decision is made by analysing the outputs of different feature extraction methods.

OPTIMISING PARAMETERS FOR SIGNAL PROCESSING

The signal processing step in this BCI has only discrete and non-numerical parameters. Two different optimising methods were implemented for optimising these parameters—brute force and differential evolution suitable for discrete values. These methods were chosen, because they are suitable for multivariate global optimisation. Two cost functions for these methods were implemented, one that penalises only if the best result from feature extraction is not the expected result and the other that also takes into account how much better is the best result from others.

OPTIMISING PARAMETERS FOR TARGET IDENTIFICATION

Unlike signal processing, the parameters of this step are mostly continuous and thus traditional differential evolution can be used. The continuous parameters are the weights for different feature extraction methods and the amounts how much the best result has to be better than other results for it to be considered the correct result. Optimising parameters with traditional differential evolution was implemented, because it can handle multivariate functions about which we cannot make many assumptions.

CONCLUSION

The code of the BCI was greatly improved in the course of the project and different methods for optimising parameters of signal processing and target identification were implemented. Unfortunately the most interesting part—finding how much the new method increased the performance—did not fit into the scope of this project. But this will be definitely done in the future. Furthermore, if the parameter optimisation works very well, then the next step would be to completely automate the process and make it dynamically improve the parameters during the usage of the BCI.
Procedural generation is often used to offer more variety without manually creating more content. There are many algorithms for generating terrain, however most of these are limited to creating mountains and valleys. The method described in the first chapter of “GPU Gems 3” uses the marching cubes algorithm and can create complex features such as caves or bridges, which is why it was modified and implemented for the Computer Graphics project.

**Density function**
The landscape generation is based on a single density function. This function uses several noise volumes (3D textures), that are filled randomly and interpolated.

**Dividing the terrain**
The terrain is built from two layers of blocks. For each block the marching cubes algorithm is run, resulting in a mesh that fits inside the given block.

**Marching cubes**
The marching cubes algorithm creates a polygonal mesh from a three-dimensional scalar field. Inside the block corner values are calculated for smaller cubes called voxels. The size and amount of the voxels determines the speed of generation as well as how detailed the resulting landscape can be.

**Cubes**
Each corner of a cube is given a value by the density function. A positive value means the corner is inside the ground and negative value the opposite. Each voxel is given a case number based on the sign of its corners. This is then used to find the triangles contained by the cube from a lookup table. Each voxel can contain up to 5 triangles. Vertices that form these triangles are on the edges of the cube. Their location is found by calculating the point on the edge where the density is zero. These triangles and vertices will form the surface of the terrain.
WHAT IS PARDIRALLI?

Pardiralli is an annual charity event organised by Eesti Vähihaigete Laste Vanemate Liit (Association of Parents of Children with Cancer).

The event takes place every summer in Tallinn where approximately 11,000 rubber ducks are dropped into a canal where they proceed to ‘race’. There are prizes for the participants and all proceeds go to charity.

People can participate by buying the ducks via the online platform — as it is a charity, one can pay as much for one duck as desired.

WHAT DID WE DO?

The new platform\(^1\) enables people to donate online to the EVLVL via the four most popular Estonian banks and thus participate in Pardiralli. It facilitates the visualisation and management of donation data for the administrators.

The platform is designed to be scalable and CMS-integratable. Our aim was to make the UX intuitive and fun as the event is aimed at children.

Our platform will be used starting from the 2017 Pardiralli.

\(^1\) https://github.com/pardiralli-dev/pardiralli

Tuule Sõber, Mari Liis Velner, Kaspar Papli, Pritt Paluoja
3rd year Bachelor’s students, CS, ICS, UT
Nearest Neighbour Search in 3D Visualization

Andreas Sepp  Marko Täht  Diana Algma  Raul-Martin Rebane

Try it online: https://morsakabi.com/nns/

Introduction

To people unaccustomed to spatial data, understanding nearest neighbour search (NNS) algorithms in higher dimensions can provide difficult. As such, we have developed an easy to use in-browser application to be a visual aid for understanding nearest neighbour searches in 3 dimensional space. This project was developed for the Advanced Algorithmics course in fall 2016/17 in the University of Tartu. The application requires a modern browser with WebGL support.

Features

- Generating a select size set of 3D points
- Interactive view of the 3D points
- Building KD-, random projection and octrees
- Searching for the nearest neighbour of any point in space using radius search
- Visualizing the NNS algorithm one step at a time

Octrees  KD-trees  Random Projection Trees

Conclusions

While the regular implementations of these algorithms are more straightforward, dissecting them in such a way that they can be executed on a per step basis and finding the required polygons for rendering were not. We hope the application will be of use for future algorithmics students and alike.
Introduction

Traditional virtual reality environments are based on rendering all the surroundings using 3D models of objects. This requires graphically modelling the whole environment, which is usually a long and tedious task. 360-degree cameras that have recently become commercially available might be able to change that situation, as image-based virtual reality requires no graphical modelling and therefore has the promise of being faster with the results resembling the real world more accurately. The downside of this approach is the requirement to create a separate image for each possible location in the scene that is being captured, thus resulting in a large number of images that need to be labelled, so that navigation between images would become possible. This project investigates possible methods to create such maps semi-automatically and provides an interface to display the virtual reality environments so created.

360-degree Imaging

In this project the Samsung Gear 360 camera (Figure 1) was used for capturing 360-degree content. The camera has two ultrawide lenses that both have a view angle of 190 degrees, thereby covering the whole 360-degree view together.

Fig. 1: Samsung Gear 360 camera seen from two different view points

360-degree images need to be handled slightly differently compared to images from ordinary cameras. Most conventional cameras output images that more or less correspond to the way we see the world. The view angle is at most 120 degrees in most cases, so an image looks similar to what we would see with our own eyes. 360-degree images, however, show the world differently: all the surroundings are visible on the same image at once. For a human observer that kind of situation is hard to comprehend and it would be much more meaningful to show only a subset of that image corresponding to some narrow view angle.

Stitching

The spherical images need to be stitched together before they can be used for virtual reality. This can be achieved by creating correspondences between the overlapping parts of the spherical images, and generating a new final image based on those correspondences. The result is called the equirectangular image.

Fig. 3: Equirectangular image, result of the stitching process

Rendering

After the image has been stitched together, the equirectangular image can be used for generating views from it, this process is illustrated on Figure 4. An equirectangular image is shown along with two images generated from it given two slightly different view angles. Notice, how the equirectangular image itself is not suitable for image display: some parts of the image are distorted, whereas some straight lines are shown very curved on that image. The generated images, on the other hand, are much more natural to look at, as they correspond to a view angle that resembles the view angle of human vision.

Fig. 4: Two different views generated from the same equirectangular image

Generating these views makes use of the standard computer graphics pipeline. The equirectangular image is used as a texture and mapped onto a 3-dimensional sphere. The camera is placed inside that sphere and the whole scene is rendered. This enables to generate different views by simply changing the view angle, the view angle determines, which part of the sphere becomes visible, and this creates an impression of looking around in that scene.

Map Generation

In this work, the 360-degree image data was captured using a small remote controlled car that had the Samsung Gear 360 camera mounted on top of it, shown in Figure 5. The car was used to capture image sequences of roughly parallel movement pattern as visualised in Figure 6.

Fig. 5: Remote controlled car with a 360-degree camera used for capturing video data for the virtual reality system.

Fig. 6: Equirectangular image, result of the stitching process

Extracted image sequences are mapped using a dynamic time warping based approach that places the sequences side by side. The image similarity is used to determine the cost of mapping the sequences to each other. The similarity is computed by finding the pixel-wise sum of differences between the images, while this approach might sound naive, it performed very well in determining the closest neighbours, and as a result it could be used as a reasonably good similarity metric.

The map generation algorithm outputs an implicit map that can be used for navigating the scene. At each point on that map the user can look around freely and according to the connections described by the map, the user can also move around between the different points of the map.

Conclusion

This project investigated methods for automatically creating maps for image-based virtual reality environments. A graphical user interface for displaying such environments was also created. Some existing methods were studied and tested, along with a novel method designed based on the specifics of our image capturing system.
Ultimate Disc Golf in Hyperspace

Ultimate Disc Golf in Hyperspace is a sports simulation game in virtual reality. The player can travel to surreal and colorful disc golf courses in hyperspace. The disc throwing and flight dynamics are very similar to real life, but virtual reality gives the player some additional abilities (e.g. teleportation and enhanced control over disc's flight).

The main idea is similar to real world Disc Golf - the player has to reach the end portal with the least amount of throws. Levels consist of islands and a variety of obstacles, while the disc is used as a teleportation device to move between the islands. To reach the end, there are often multiple paths with varying difficulty.

The game was designed with a focus on realistic disc flight trajectories and ease of use. Most actions are intuitive and need no instructions. It has been tested by over 50 users and user interface was improved iteratively.

Disc aerodynamics.Unity physics engine covers only translational and rotational mechanics. Aerodynamic physics script was implemented to make the disc fly like it would in a real world. Aerodynamic constants can be changed to simulate different discs (e.g. putter, mid-range, driver, etc.).

Level editor. Building good levels requires placing objects and testing the result intermittently. Level editor that allows navigation similar to pinch-zoom on mobile devices was implemented. User can also teleport to a desired location and reset the scale to instantly see the result from players perspective. Navigation script changes the users position, rotation and scale instead of moving the gameworld itself. This would allow multiple users to work on the same level without conflict in future builds.

Custom made models. All assets - materials and 3d models - in the game are custom made. That includes sound effects and background music. The only exceptions to this rule are HTC Vive controllers and logo of University of Tartu.

7 playable levels. Currently the game has seven playable levels. There is a special tutorial level that is designed to familiarize first time players with controls. Levels have different styles and difficulties. Moving platforms, stationary and moving obstacles are designed to make the game more fun to play. High scores tracking is implemented to allow users to compete against each other.

2. V. R. Morrison "The Physics of Frisbees", Mount Allison University Physics Department, 2005

UNIVERSITY OF TARTU
Institute of Computer Science
Tõnis Koppel, Computer Science 3rd year
Aadam Kaivo, Mathematics 2nd year
Supervisor: Jaanus Jaggi
Institute of Computer Science
PDF to EPUB converter

WHAT?
PDF to EPUB converter is a software solution that simplifies the process of turning PDF formatted books to EPUBs that are more suitable for various E-readers. It gives the user wide range of tools to work with. For example tools for extracting paragraphs, headings, footnotes and pictures. After that, the user can add necessary metadata and the EPUB is ready for downloading.

WHY?
Eesti Digiraamtutute Selts (Estonian Digital Books Association) was interested in an application that could significantly decrease the time that it took to convert PDF books to EPUBs. In addition, they were mostly outsourcing the conversion to external contractors who had necessary knowledge in HTML, CSS and EPUB specific formatting. Our goal was to make that process less dependent on technical skills and reduce the average amount of time it took to convert a book.

HOW?
Our application’s first step was to generate rough estimations on how difficult the conversion process was going to be. This was done mostly by counting different elements from the PDF. After that, the PDF was converted to HTML although it kept its original layout. Then the user has different tools that were done in JavaScript at their disposal. First they should select the ending point of the chapter. The next step would be to eliminate unnecessary elements like page numbers between paragraphs. After that footnotes can be selected. Then a suitable heading and all the paragraphs. Last step is the pictures. If there are more chapters left the cycle continues, otherwise, metadata can be added and after that, the EPUB file is generated.

WHAT’S NEXT?
PDF to EPUB converter is not a finished product yet. It evolves on a day-to-day bases and there are many more features that it is going to obtain in the future. For example at the moment it converts PDF to EPUB2 standard, but there is a new standard available - EPUB3. Also some other aspects like footnote linking and separate covers are potential features.

TEAM
Sebastian Värv (BSc 3y), Karl-Mattias Tepp (BSc 3y), Meelis Tapo (BSc 3y), Silver Vapper (BSc 3y) – all from Faculty of Science and Technology, Institute of computer science, Mentor: Evert Nõlv

http://pdftoepub.weebly.com/
A Framework for Analysing Topics in University Courses

Ragnar Vent (Computer Science, Master’s studies, Institute of Computer Science)
Supervisor: Slim Karus

Introduction

The idea behind this project was to build a framework that provides the opportunity to discover and understand topic overlaps between different courses. First, we gather course materials (e.g. content pages, PDF files, PPTX files) from various sources (e.g. http://courses.cs.ut.ee/). As a next step, all the textual data is extracted from these documents and appropriately cleansed and conformed. Finally, a topic modelling algorithm called Latent Dirichlet Allocation (LDA) is applied to the data to resolve topics.

General workflow overview

- **Data collection and extraction**: Web scraping, Data integration, Text extractions from files
- **Data cleaning and conforming**: Tokenization and Lemmatization, Finding frequent co-occurring words, Acronyms extraction
- **Text analysis**: Topic modelling, Topic name resolving
- **Data visualization**: Course-topic relations, Course-material topic relations

Course topic modelling

Course material topic modelling

Example of material-topic distribution for course MTAT.03.015 Computer Graphics (2015/Fall)

In addition to performing topic modelling at a course level, where the goal is to seek overlaps between different courses, we also performed topic modelling at a course material level. The aim is to investigate from which topics are individual course materials (e.g. lecture slides) composed of. This opens the opportunity to seek overlaps between two course materials that are part of completely different courses.

atiasa.cs.ut.ee
Batoru Shitī: The Tank Game

Rasmus Soome (Institute of Computer Science, Faculty of Science and Technology, Computer Engineering Ba),

Tambet Viitkar (Institute of Computer Science, Faculty of Science and Technology, Computer Engineering Ba)

Project website: https://bitbucket.org/rasmus_soome/batoru_shiti_public/

---

**The Game**

Tanks (each with 30 lives/health points) enter the battlefield with a mission to blow up the opponent. Each tank has a cannon (laser) to aim and hit enemy's hitboxes until he or she is out of lives.

Tanks will have two sensors: front and back. It's a well-known fact that tanks have a stronger armor in front. Therefore the front sensor has a maximum damage value of 5 and the sensor at the back has a maximum damage value of 10. It should make the game more fun and challenging. Damage values are not static. Instead, they are generated per hit basis. For example, the sensor detects a hit and generates a random value with normal distribution in the range from 0 to 5.

The round will continue until only one tank has health left. Once the tank has run out lives, it stops, does a 720-degree turn to signal defeat and disables itself.

---

**The Tank**

Each tank is composed of a raspberry pi, gopigo platform, an arduino, a laser cannon, a servo controlling the position of the cannon, lasers for shooting the opponents’ hitboxes and light sensors covered by red ping-pong balls acting as hitboxes.

---

**The server**

Game is coordinated by a central server, which keeps track of how much health each of the robots have and notifies the robots when they are out of health.

---

**Software**

Most of the software is written in Python (yet, some C for programming the arduino and a little bit of shell scripting to set the robot up).

The approach taken while programming the robot is to keep different functionalities in different modules so adding/swapping functionality would be as easy as possible.

---

**Figure 1. Diagram of the modules illustrating the robot software**

---

Project poster session in LOTI.05.010 Robotics
May 2018
Problem
Consider a beginner programmer trying to find a bug in a written program. A debugging session may look like this:
1. Start debugging;
2. Step through lines of code;
3. Possibly find clues about the issue;
4. Repeat steps 1-3 until enough is discovered about the issue. Once the bug is found, the programmer could have restarted debugging sessions multiple times. This can become frustrating very quickly, especially for novice developers.

To make debugging smoother, the debugging tool should allow stepping back in time. This enables the programmer to browse all states of the flawed program execution without leaving the debugging session. This was added to Thonny, a Python 3 IDE that features debugging tools for beginners [1].

Development
There are two main approaches for implementing stepping back in time: reverse execution or omniscience (saving all program states and replaying these). Omniscient debugging was chosen, as it does not necessitate large changes to the existing debugging architecture [2]. Development of omniscient debugging started by first implementing a basic proof of concept. This was taken as a foundation on which different options to integrate the solution were tested and the final solution was implemented.

Solution
The following changes were introduced to Thonny’s debugger:
- All states of the program are saved.
- Step back command added.
- Stepping both into and over in past states.
- View previous states after encountering unhandled exception.

The following operations can be done without restarting the debugging process:
- Check states prior to the current program state.
- After accidentally stepping over, step back and step into the most interesting piece of code.
- Continue debugging previous program states after encountering an unhandled exception.

There are also some expected drawbacks – memory usage and time complexity is noticeably higher, mainly affecting larger programs. Omniscient debugging will be made optional to keep the debugger feasible for larger programs.

Try it Yourself!
The beta version featuring omniscient debugging among other new features is available at Thonny’s repository – refer to the downloads page and download-install Thonny version 2.2.0b2. To try omniscient debugging:
- start debugging a program;
- issue some “Step into” or “Step over” commands;
- issue the “Step back” command by pressing the F9 key or choosing “Step back” from the run dropdown menu.

Bibliography
Particle swarm optimization in games

Particle Swarm Optimization (PSO)

- Population based stochastic optimization technique developed by Kennedy and Eberhart in 1995
- Based on social behavior of species in nature, e.g. a flock of birds or a school of fish
- Applied in many areas: function optimization, artificial neural network training, fuzzy system control.

How it works

- Swarm: a set of uniformly spread particles (S)
- Particle: a potential solution
  - Position: $x_i = (x_{i,1}, x_{i,2}, …, x_{i,n}) \in \mathbb{R}^n$
  - Velocity: $v_i = (v_{i,1}, v_{i,2}, …, v_{i,n}) \in \mathbb{R}^n$
- Each particle maintains individual best position (pBest)
- Swarm maintains its global best achieved so far (gBest)
- At each step, particle’s velocity and position are updated based on its pBest and gBest
- Algorithm stops when global optimum is reached or after $N$ iterations

Comparison with Genetic algorithm (GA)

PSO shares many common points with GA. Both algorithms start with a population of randomly generated solutions and have fitness functions for its evaluation. They produce new populations and search for the optimum.

However, PSO doesn’t have genetic operators like crossover and mutation. Particles update themselves with the internal velocity. The information sharing mechanism in PSO is significantly different. In GA, chromosomes share information with each other, thus the whole population moves like a one group towards an optimal area. In PSO, only gBest gives out the information to others. It is a one-way information sharing mechanism. The evolution only looks for the best solution. Compared with GA, all the particles tend to converge to the best solution quickly in most cases.

Simulation of PSO

The Game

In the first part of the game particles chase the character using the particle swarm optimization

In the second phase of the game particles escape from the character following the maximum distance greedy algorithm

Play it here: https://github.com/kopylash/algorithmics-project/
Shuriken Way is a physics-based puzzle game for Android devices, where the goal of the player is to complete all the levels while getting as high of a score as possible in each of them. The player is given control of a shuriken (a star-shaped throwing weapon) and the goal of every level is to collect all the coins in it. This is usually done by hitting the coins with the shuriken. Very often reaching a coin requires the player to plan out their actions and use certain mechanics of the surrounding objects. Throughout the levels there are also objects that are explicitly there to try to stop the player from reaching some coins. The score for a level is based on how quickly the player was able to complete it. Shuriken Way was developed for the purpose of writing a Bachelor's thesis but the development will be continued in the future.

A total of 8 types of game objects with different mechanics have been implemented and can be encountered while playing the 9 (currently) available levels.

**The Approach**

The game was developed without the use of game engines. Low-level techniques for computer graphics were used instead. OpenGL ES 2.0 was used for rendering the 3D graphics and Android Java was used for programming the 2D mechanics of the game.

**Play The Game**

You can play/test the game yourself by installing it onto an Android device. Go to bit.ly/2LCiVhi (scan the QR code) and go to the download page. Download the installation package (shuriken-way.apk) and run it.

Anton Tsugunov  
Computer Science  
3rd year Bachelor’s  
Supervisor: Raimond-Hendrik Tunnel, MSc  

University of Tartu  
Faculty of Science and Technology  
Institute of Computer Science