Basic information

Project name	Object movement intention annotation tool
Abbreviation	AnTool
Supervisor	*RNDr. Júlia Škovierová, Ph.D. <julia.skovierova@cvut.cz>*
Consultant	*Mgr. Radoslav Škoveira, Ph.D. <radoslav.skoveira@cvut.cz>* *RNDr. David Bednárek, Ph.D. <bednarek@ksi.mff.cuni.cz>*
Annotation	AnTool will use data captured by sensors of a self-driving car within H2020 project UP-Drive (http://up-drive.eu/). This data should suffice for the AnTool project life span. The more general and secondary goal is to make AnTool useful in other application domains, e.g. a general traffic related data. AnTool should be able to generate automatic annotations for certain specified traffic related behaviours as well as the possibility to edit the generated annotations easily (e.g. to correct possible mistakes of the automatic system) and create manual annotations.

Motivation

In recent years, the amount of statistically-based decision making applications requiring huge training data is growing rapidly. The prime example are topical convolutional neural networks (CNNs). Decision making applications are mostly learned empirically in statistical manner.

The quality of training set (annotated data) influences the error rate of the classifier significantly. The task suggested as a student software project is to design, develop and implement the semi-automatic annotation tool (AnTool). AnTool should be designed for moving objects in a data stream. AnTool will be developed to work primarily with UP-Drive data, however, it should be able to work with different data from traffic related projects (traffic monitoring, self-driving car, …).

Data annotation can be performed manually, automatically, or semi-automatically. One example of a manual annotation is LabelMe (http://labelme.csail.mit.edu/Release3.0/), which is used for labelling objects in images. However, manual labelling is often very time and money consuming, especially for large databases, therefore automatic or at least semi-automatic annotation is preferred.

Project description

The AnTool should consist of these parts:

a) Module for processing of input data

b) Annotation module

c) Interface for automatic/manual annotation

Module for processing of input data:

- The input data for the development of the software will come from UP-Drive project focused on autonomous urban driving. The data will consist of map (including road information such as junction connections, position of zebra crossings, speed limits, etc.), information about ego car (i.e. the car performing the data recording; information will include GPS position, speed, heading of the car,…) and information about all detected objects (e.g. position of other objects in the scene with respect to ego car).

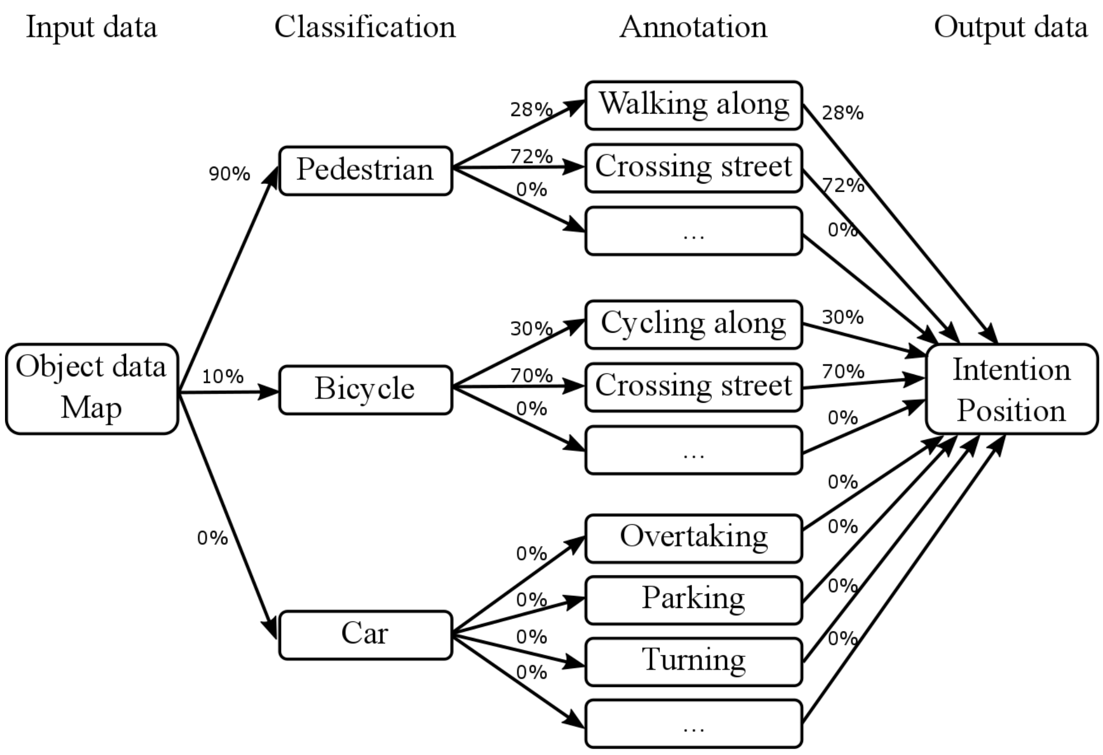
- Detected objects have attached additional (semantic) information such as object's classification, position, heading, speed, etc.
- The information about all dynamic objects is updated periodically.
- It should be possible to create custom data importer capable of reading different input data format. That means the developed software will have internal representation of the data independent of the format of the input data.
- **Functionality specification**
    - Map information
        - Load input map data from an xml file
    - Ego car data
        - Load ego data (position, speed, heading,…) for each timestamp based on specification in a header file
    - Dynamic object data
        - For each timestamp: Load information about all objects (id, classification, position, speed, …)
    - All data import functions must be programmed as separate modules to be easily replaced for different data formats

Annotation module:

- The core functionality is detection of intentions/behaviours of dynamic objects in the processed data and creating meaningful annotations of these intentions/behaviours.
- Annotations:
    - The annotations of intentions/behaviours of objects should analyse data several seconds into each object's future (it should be possible to specify the exact time interval). It is important to take into account possible imperfections in measured data, e.g. noise. Favorable solution to this is computing behavior over a certain period of time (specified by the user), i.e. computing the behavior using sliding window approach.
    - For each time step, there is information about all object`s classification, position, heading, etc.
    - The set of possible intentions/behaviours will be determined based on the object`s classification, e.g. pedestrian, bicycle, car. Examples of annotations in the case of intention estimations in traffic situations:

        - Pedestrian: crossing the road, walking along the road...
        - Car: stopping, overtaking, parking, turning...
    - Intention annotations can be derived from object`s information in time in two possible ways:
        - This information can be derived from object`s information in future, where we can see its position/speed/heading. Based on this information, we can annotate the intention in current time.
        - Looking from object`s future, its intention can be set into past.
    - The fact, that the objects are not tracked during the whole recording should be considered - e.g. add information about the reliability of intention annotation. The tool should also be able to deal with missing data (e.g. temporal disappearance of object due to occlusion or permanent object disappearance of object due to it being out of range of the sensors). Positions of missing objects should be approximated using a simplistic estimation (temporal disappearance: simple position interpolation; permanent disappearance: simple physics based position estimation set time period into the future – usually a few seconds). Approximated positions should also be scored with reliability (with reliability decreasing from 1 to 0 with increasing consecutive length of the predictions) which should be used in the final computation of the probability/reliability of the behaviour annotation.
- The annotation tool should work autonomously as much as possible, but allow for manual input as well.
- Modularity – Expanding the set of detected behaviours (i.e. creating detectors for new intentions/behaviours) should be part of the core functionality. Ideally, definitions for at least simple rule-based detectors will be separated from the program itself (e.g. in a text file).
- The possibility of the future development (the software should be implemented so that further development of general functionality will be as simple as possible).
- The possibility of online/offline data processing. Software should be capable of operating in two modes:
    - offline automatic annotation - performing annotation as fast as possible without visualization (with possibility of replaying the generated annotations with visualization),
    - online automatic annotation - visualizing the annotation process and allowing user interaction.
- It should be possible to load already annotated data for further manual editing and viewing.
- The type of annotation (manual/automatic) should be clearly marked.
- Designing suitable output format (i.e. format of the annotation data) will be part of the project.

- **Data Flow example**



- **Object intention annotation example**

  Situation:

  - Ego car is approaching zebra crossing. Pedestrian walking along the road, pedestrian will cross the road on zebra crossing

  Input:

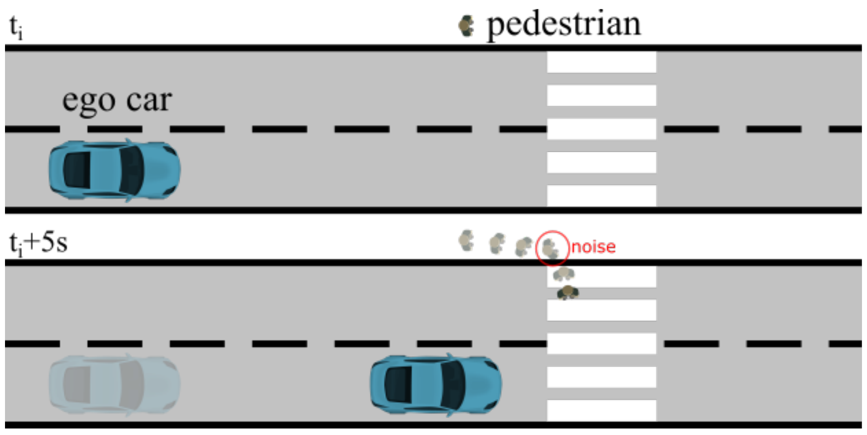
  - Object: position, heading, classification (pedestrian 90%, bicycle 10%)
  - Car: position, heading, speed, acceleration

  Annotation

  - It is important to take into account not only object information from time t_i+*n s*, but also information in time between current time t_i and time t_i+*n*. The exact time shift should be customizable (in the example below, we use *n*=*5*).
  - Pedestrian: will cross the street

  Output

  - Annotation of intention in time t_i, that pedestrian will cross the street.



- **Functionality specification - Annotations**
  - For each object this information needs to be extracted or computed from the provided data:
    - Object ID
    - Infrastructure IDs (Set of infrastructure elements that are important for the intention, e.g. zebra crossing, line)
    - Probability (Probability of this intention estimate)
    - Reliability (e.g. in the case where the object is not currently visible and its position is being approximated)
    - Specification of the intention (e.g. straight, turn, ...)
  - It is important to take into account not only the positions in times *t* and *t*+*n*, but also the positions between these times. Therefore, the input for annotations will be an array of positions of the object between time *t* and *t*+*n* and the whole map information.
  - In case the intention cannot be decided in the current time span, there should be possibility to extend this time span. E.g. if the pedestrian is on the left side of the road in time *t* and on the road in time *t*+*n*, it should be possible to look (several) second(s) into the future to see if pedestrian actually crossed the road and is on the right side of the road (otherwise it might have been a noise, pedestrian changed mind, etc.).
  - Currently, only pedestrian related intentions should be processed in the program. However, the system should be developed with the capability to implement detection rules for other traffic intentions – see section “The possible annotations for cars”.
  - The possible annotations for pedestrians:
    - Walk along the road: The position of the pedestrian in respect to the road will not change over time, e.g. the pedestrian will be on the right side of the road in all time steps between *t* and *t*+*n*.
    - Standing: The position of pedestrian will not change between the time *t* and *t*+*n*. It should be possible to set a deviation within which the pedestrian’s position will

be considered as unchanged (only in the script for the rule, i.e. not in the GUI).

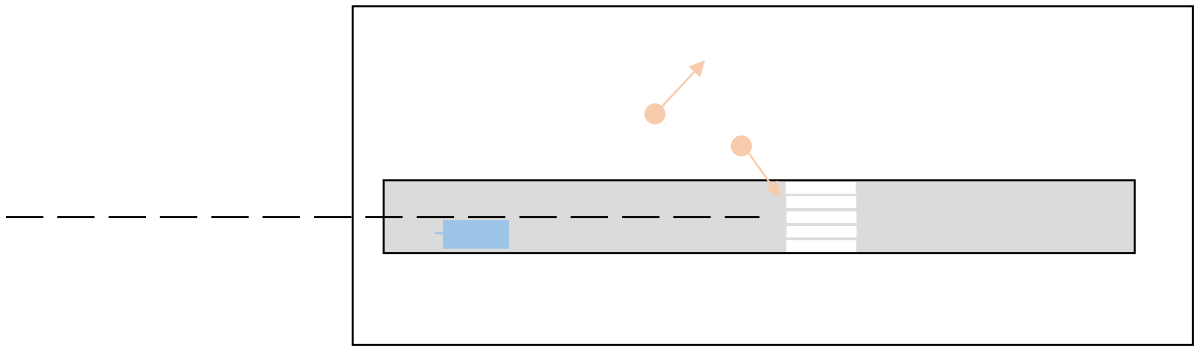
    - Cross on a zebra crossing (+direction): The pedestrian is on one side of the road in time *t* and on the other side of the road or on the road (need to check position further in time) in time *t+n*. The pedestrian is on pedestrian crossing or near it. It should be possible to set a max distance parameter, where the pedestrian is considered to be still on or near a zebra crossing (only in the script for the rule).
    - Cross outside of zebra crossing (+direction): The pedestrian is on one side of the road in time *t* and on the other side of the road or on the road (need to check further in time) in time *t+n*. The pedestrian is crossing the road away from any pedestrian crossing.
  - The possible annotations for cars:
    - Lane Intention (Follow, Change, Change Left, Change Right)
    - Multi-object intention (Follow, Overtake)
    - Cross traffic intention (Ignore, Yield, Make Way)
    - Longitudinal intention (Accelerate, Constant, Decelerate)
    - Zebra crossing intention (Stop, Pass)
    - Crossing Intention (Turn Left, Turn Right, Go Straight, U Turn)

Interface for automatic/manual annotation:

- Interface should contain (simple vector) graphical visualization of processed data (e.g. objects on the map) with visualized intentions and movement trajectories. It should also visualize temporal progression through the current recording (e.g. a progress bar). In case of replaying already annotated data, the progress bar should also serve as a mean to skip to any part of the recording.
- The human operator should be able to observe the visualized sequence and if he/she spots a wrong annotation, he/she should be able to stop the tracking and provide manual annotation. Otherwise the automatic annotations will be accepted.
- **Graphic User Interface (GUI) functionality specification**
  - Main functionality
    - Loading of the input data (map, ego car data, object data) using a dialog window.
    - Run automatic annotations with loaded data, which can be performed offline.
      - Possibility of changing of the control parameters such as prediction interval and sliding window size.
    - Real-time playback of the annotated data.

- Possibility of stopping the real-time playback and changing the annotations manually.
- Clear marking of manual/automatic annotations and annotations requiring additional correction (e.g. annotations with low reliability).
    - Save the data in general format (e.g. txt, see sec. Platform, technologies)

- Displaying the data
    - Simple vector graphic with map displaying and simple objects (e.g. see figure).



    - Display: map information, ego car, other objects (pedestrian, car, bicycle, …)
    - For each object show in separated dialog/part of the GUI:
        - All possible input informations (position, category, speed,...)
        - Annotations

Platform, technologies

The output software should contain user-friendly GUI with well-written documentation for good understanding of used approach.

AnTool should be written in C++, C# or Python to match software development tools used in UP-Drive project.

The output file should be a text file for easy portability among different programs and possible human readability. The suggested data format is JSON.

Difficulty estimation

Number of people: A team of 4-5 students.

The deadline for completion: optimally 7 months (maximum 9 months)

Plan of the work:

1st month: analysis of the input/output data, analysis of requirements for implementation, proposal of software architecture

2nd – 5th month: implementation and testing

4rd month: core functionality implemented (automatic annotation)

5th month: additional functionality implemented (user interface, manual annotation)

6th – 7(9)th month: testing, debugging, software and user documentation

Project definition

Project is focused on followed areas:

Discrete models and algorithms	
	Discrete mathematics and algorithms
	Geometry and mathematical structures in computer science
	Optimisation
Theoretical Computer Science	
	Theoretical Computer Science
Software and data engineering	
X	Data engineering
X	Software development
	Web engineering
	Database systems
X	Analysis and big data processing
Software systems	
	System programming
	Reliable systems
	Powerful systems
Mathematical linguistics	
	Computer and formal linguistics
	Statistics methods and machine learning in computer linguistics
Artificial intelligence	
	Intelligent agents
X	Machine learning
	Robotics
Computer graphics and development of computer games	
	Computer graphics
	Development of computer games

Notes

none