Proceedings of the 2013 Federated Conference on
Computer Science and Information Systems pp. 675–678

Image Recognition System for the VANET

Štefan Toth, Ján Janech, Emil Kršák
Department of Software Technologies
Faculty of Management Science and Informatics
University of Žilina
Univerzitná 1, 01026 Žilina, Slovakia
Email: {stefan.toth, jan.janech, emil.krsak}@fri.uniza.sk

Abstract—This paper describes a system for recognition of objects in traffic scene from multiple moving vehicles. The system is based on query and image processing. It allows image recognition along with a spatial relation. A user or a machine makes a query in which description of searched objects are defined. Then the query is sent to vehicles in order to process in real-time. If any vehicle recognizes object of interest given by the query, the answer is returned to query author.

I. Introduction

AT the present time VANET (Vehicular Ad hoc NETwork) is considered as one of the most important technologies in ITS (intelligent traffic systems) [1][2]. It refers to an ad-hoc network in which vehicles communicate each other or with the infrastructure. Nowadays this is a promising research area, in which many researchers and vehicle manufacturers have proposed and developed many applications expected to improve traffic safety or increase comfort of users.

In this paper we propose one of the applications focused on image processing. It allows recognition of objects in traffic scene using a camera placed in a vehicle. It proposes a novel approach based on a query processing in which a user or a machine is able to make a complex query whereby objects are detected and recognized.

II. Similar Works

VANET networks are prospective future of the automotive industry, as it provides tremendous opportunities in terms of improving safety or comfort for users. In the field of image processing and analysis, many applications have been proposed such as a vision-based active safety system for driver assistance in intersection scenarios [3], See-Through System [4] in case of passing large vehicles and vehicles with darkened rear windscreen, which are difficult to see through and make the situation before them unpredictable. Another example is a system for locating wanted vehicles [5] based on license plate recognition and also a surveillance system proposed by Badura [6].

All applications, however, are problem specific. They solve a particular problem of exactly defined area for which they were designed. Our research has not yet identified any existing solution that would be able to search for general information – the objects contained in the image on the request.

Although many methods for real-time detection and recognition of objects have been developed using MPEG-7, QBIC / CBIR systems, we have to point out that generic object detection and recognition system suitable to be used in VANET is missing. Thus we are proposing a solution for detection of traffic objects and also general objects appeared in traffic scene visible from vehicle.

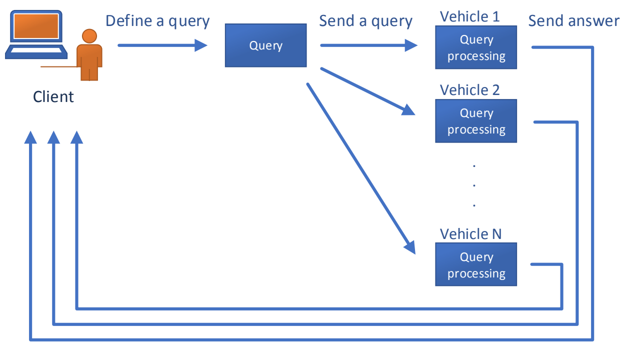

Fig. 1 Outline of the proposed query system

III. The Query System

The idea of the proposed system is very simple. It consists of following steps:

1. Query definition – a query is created manually by a person or automatically by an application. In a query an object of interest is defined, as well as required output and processing length.
2. Query sending to vehicles – a query is sent to one or more vehicles (query processing) according to defined target area in which the image is supposed to be processed (the area is defined, for example, using IP broadcast or geocast).
3. Query processing and output return – since a requested object can appear sporadically, the image captured by a camera could be processed periodically until a condition is met. Thus the length of processing can be limited either by certain time (e.g. 15 minutes), by

event (e.g. after required object was found, an engine of a vehicle was turned off) or by geographical range in space (spatial definition specified by GPS location).

The basic principle is also shown in Fig. 1.

A. Image Objects

The base of the system is image object, which constitutes either a real object or segmented part in an image. In order to recognize real objects we performed dozens of test drives in surroundings of Žilina city (Slovakia). We have identified some important objects, which could be recognized in the system:

- Sky (sun, clouds, weather condition)
- Mountains
- Roads (traffic lane, sidewalk) and intersections
- Horizontal traffic signs
- Vertical traffic signs
- Traffic lights
- Texts
- Vehicles
- People
- Animals
- Buildings and poles
- Trees, bushes
- Lakes, rivers
- Objects moving in the air (flying objects)

In order to recognize unknown objects, we define the so-called general object which is composed of a model consisting of set of templates with a description. This model will be directly incorporated into query so it will be possible to refer to it and be used in querying.

After analysis we suggest that the general object model would contain following parts:

- Unique model identifier should be included in order to refer to it in a query.
- Object characteristics – deformability, motionlessness, size etc.
- Views of model are expressed either in form of a set of source images, features or in the form of model with value parameters of a machine learning algorithm. In order to recognize an object, source images would represent the object from different viewpoints (especially if the object is not symmetric) and also capture its various visual aspects in case of deformability and especially in such positions it can frequently be seen.
- Described parts of the model which make up the model (e.g. in case of vehicle if there is a view of rear part of it, we can see a trunk, right and left light, part of right and left wheel etc.). Parts are defined for every viewpoint and will have certain position in order to refer to them in querying and e.g. to find out if they are visible or if they are in relation with other objects.

A general object can be expressed for example by using templates of still images, SIFT (Scale Invariant Feature Transform), ANN (Artificial Neural Network), or other state-of-the-art methods. Thus a general object could be represented by defining the source data such as images, descriptors, or weights as required by chosen method.

B. Querying Image Objects

We can look at querying image objects as a complex task which can be decomposed into simpler tasks and their interconnections (Fig. 2). Thanks to these interconnections the sets of image objects will pass in left-to-right direction in order to process these sets in elementary tasks.

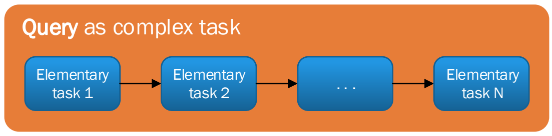

Fig. 2 Query as complex task composed of N elementary tasks

When we get to the last elementary task N, it depends on its output what happens next (Fig. 3). If the output is non-empty set, the answer will be sent in form which is defined at the end of the query. Otherwise, it depends on definition of the query whether the requirement is to be re-processed or terminated and if it terminates whether to send an answer. This behavior was chosen for a reason that it is very often necessary to search for an object in an image which is likely not to be present there. Therefore we need cyclic query processing until the answer (non-empty set) is obtained. Actually, an empty set will represent insufficient output (i.e. required object is not found).

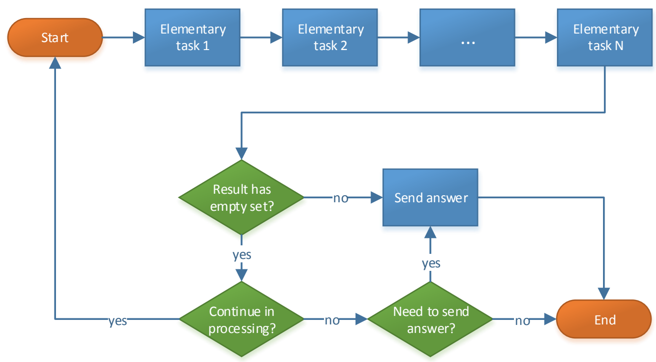

Fig. 3 The proposed system of processing a query as complex task

Processing of an image query therefore means executing of elementary tasks involved in sequentially concatenating sequence. Elementary task represents any basic operation with the image, which can be in a form of detection, localization or classification of objects, relations between them, choosing and other basic image processing algorithms built and supported in the system.

C. Inputs

The input is defined by a client. In this query, it is important to determine what to perform, when to finish the processing and what to send as an answer. Such information is possible to write in a certain defined format, e.g. XML, JSON or others. Since we will work in VANET environment, the format should be as compact and small as possible because of the transmission. Our aim is not exactly to define exact format in which data will be transmitted via the network, but only query format in which it will be processed (i.e. what and which parts are important in a query). Therefore, we define following required and optional fields:

- `INPUT` – defines a unique query name, which is also used in the output answer.
- `MODEL` – a definition of general object recognition model. Definition of the model is optional, unless we require recognition of an unknown and undefined object in the system.
- `TERMCONDITION` – a termination query condition. It is optional in case of a query with quick response (only one image processing obtained from the camera is requested) and necessary if we want to process images cyclically (video sequence). Specified condition then determines when the processing loop finishes.
- `REQANSWER` - determines whether we always require an answer, even if the query returns an empty set.
- `QUERY` – a definition of a query. It is a mandatory field, which forms the core and allows you to define what objects with characteristics are required to obtain. The last elementary task will define the output of the query.

D. Outputs

As a result of the output can be everything what is needed by original request, e.g. original image captured by the camera or only some parts of the image where are one or more objects of interest. In addition it could contain GPS coordination, common value (number, text, logical value) containing required information (the number of objects, recognized text, licence plate number etc.) or their combination.

IV. The Query Operations and Operator

In query definition we could use many elementary tasks sequentially connected each other. These tasks constitute possible operations connected by separation operator.

A. Operations

For purpose of manipulating image objects we have introduced operations as following:

- Input and output functions
  - `GETIMAGE` – captures an image from a camera or another image source and converts it to an image object. Captured image represents traffic scene as a root image object.
  - `RESULT` – returns a set of objects defined by an expression being its argument as query result. The result should be sent back to the querying vehicle.
- Object searching operation
  - `FINDOBJECTS` – detects and recognizes requested image objects in the input set of image objects. The used algorithm of object recognition depends on the system where the query is executing. It can be used any state-of-the-art recognition algorithm.
- Manipulation operations
  - `WHERE` – selection is an operation for filtering the input set of image objects by the given condition.
  - `SELECT` – collection; it is an operation for transforming each item from the input set using the expression given as the operation argument.
- Set operations
  - `UNION` – standard set union between the input set and the set given as an operation argument.
  - `INTERSECTION` – standard set intersection between the input set and the set given as an operation argument.
  - `MINUS` – standard set difference between the input set and the set given as an operation argument.
- Temporal data storage operations
  - `SAVE` – stores its input into temporal data storage.
  - `LOAD` – loads data from temporal data storage and passes it as its output.
- Spatiotemporal operations
  - `RIGHTOF` – selects all image objects on the input that are on the right side of the objects given as operation argument.
  - `LEFTOF` – selects all image objects on the input that are on the left side of the objects given as operation argument.
  - `ABOVE` – selects all image objects on the input that are above of the objects given as operation argument.
  - `BELOW` – selects all image objects on the input that are below of the objects given as operation argument.
  - `IN` – selects all image objects on the input that are inside of the objects given as operation argument.
  - `OUT` – selects all image objects on the input that are outside of the objects given as operation argument.
  - `NN` – selects nearest object to the objects from the set on the operation input from object set passed as the operation argument.
  - `DISTANCE` – selects all objects from the set on the operation input with its distance to any object from set passed as the operation argument.

B. Separation Operator

As separator between operations, operator `|` was introduced. Using that operator we can express following sequence of operations:

```
operation1 | operation2 | ... | operationN
```

where the last operation `operationN` will be a result of the query.

V. Example of Usage

To demonstrate the proposed system and language we have presented some examples of using it.

A. Image of Traffic Scene

The simplest query is returning single image from a camera:

```
GETIMAGE
```

Here we do not have to use separation operator and also RESULT operation since we do not require any additional information. If we would like to return more information and from rear camera, we can use query as following:

```
GETIMAGE('rear camera') |
RESULT(image => image, GPS.GetPosition)
```

This query means to return image and current GPS position of the car.

B. Car Detecting

If we would like to find a red car, we should use sequence of following operations:

1. Get an image from a camera (GETIMAGE).
2. Detect car objects in the image (FINDOBJECT). The result of this step is a set of image objects containing a car.
3. In order to find only a red car, we can then use WHERE operation to select only a car with red color.

Final query will be composed as follows:

```
GETIMAGE | FINDOBJECTS('car') |
WHERE(car => car.Color = 'red')
```

It depends on a person how he defines a query. There are many possibilities how to make the same search for a red car. We could use the color segmentation at first and then apply finding objects operation:

```
GETIMAGE | FINDOBJECTS('red segment') |
FINDOBJECTS('car')
```

C. More Complex Example

To demonstrate a more complex example, we define a query in which we will detect a red car with a license plate starting with ZA and ending with AB. In addition there was a specific picture (for example skull with crossbones) on right side of the license plate number. Therefore we define a general object consisting of wanted picture in query format, so we could use it in query. Then result query by that definition could be like this:

```
GETIMAGE | FINDOBJECTS('car') |
WHERE(car => car.Color = 'red' AND
            car.View = 'rear') |
FINDOBJECTS('license plate') |
WHERE(plate => plate.Text.StartWith('ZA') AND
              plate.Text.EndWith('AB'))|
SELECT(plate => plate.ParentObject.Crop(
       plate.Location.Increase(0,100,400,100))) |
FINDOBJECTS('skull') |
RESULT(skull => skull.ParentObject, GPS.Position,
                GPS.Heading)
```

As result of that code is an image of a car (ParentObject), GPS position and heading.

Other examples of proposed system are introduced in [7][8].

VI. Conclusion

In this paper we proposed a novel application for the VANET, whereby we are able to recognize objects from images captured by a camera placed in a vehicle. We described a query system and language based on object oriented manner. Using that language we can detect any objects along with relations defined by a query.

Acknowledgment

This contribution/publication is the result of the project implementation:

Centre of excellence for systems and services of intelligent transport II., ITMS 26220120050 supported by the Research & Development Operational Programme funded by the ERDF.

"Podporujeme výskumné aktivity na Slovensku/Projekt je spolufinancovaný zo zdrojov EÚ"

References

[1] J. E. Naranjo, J. G. Zato, L. Redondo, M. Oliva, F. Jimenez and N. Gomez, "Evaluation of V2V and V2I mesh prototypes based on a wireless sensor network", 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), Washington, DC, USA, 2011.

[2] J. Janech, A. Lieskovský, E. Kršák, "Comparation of Strategies for Data Replication in VANET Environment", 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Fukuoka, Japan, 2012.

[3] L. Hoehmann, A. Kummert, "Car2X-communication for vision-based object detection", International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Dubrovnik, 2010.

[4] C. Olaverri-Monreal, P. Gomes, R. Fernandes, F. Vieira and M. Ferreira, "The See-Through System: A VANET-enabled assistant for overtaking maneuvers", IEEE Intelligent Vehicles Symposium (IV), San Diego, CA, 2010.

[5] M. Ferreira, H. Conceição, R. Fernandes and R. Reis, "Locating cars through a vision enabled VANET", IEEE Intelligent Vehicles Symposium, Xi'an, 2009.

[6] Š. Badura, A. Lieskovský, "Intelligent traffic system: cooperation of MANET and Image processing", IEEE First International Conference on Integrated Intelligent Computing (ICIIC 2010), Bangalore, India, 2010.

[7] Š. Toth, Spracovanie obrazu s využitím dopytov v prostredí VANET (Query Based Image Processing in the VANET), PhD dissertation, Dept. of Software Technologies, University of Žilina, Žilina, Slovakia, 2013.

[8] Š. Toth, J. Janech, E. Kršák, "Query Based Image Processing in the VANET", 5th International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN2013), 2013.