Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 51-30 (2018) 533–537

Transportation of small objects by robotic throwing and catching: applying genetic programming for trajectory estimation

Ruslan Gayanov*, Konstantin Mironov, Ramil Mukhametshin*, Aleksandr Vokhmintsev***, Dmitriy Kurennov****

**Ufa State Aviation Technical University*
Ufa, Russia (e-mail: gayanov96@mail.ru, mramilm1996@gmail.com)
***Ural Federal University, Ufa State Aviation Technical University*
Ekaterinburg, Ufa, Russia (e-mail: mironovconst@gmail.com)
****Chelyabinsk State University, Yugra State University*
Chelyabinsk, Khanty-Mansiysk, Russia (e-mail: vav@csu.ru)

Abstract: Robotic catching of thrown objects is one of the common robotic tasks, which is explored in several works. This task includes subtask of tracking and forecasting the trajectory of the thrown object. Here we propose an algorithm for estimating future trajectory based on video signal from two cameras. Most of existing implementations use deterministic trajectory prediction and several are based on machine learning. We propose a combined forecasting algorithm where the deterministic motion model for each trajectory is generated via the genetic programming algorithm. Genetic programming is implemented on C++ with use of CUDA library and executed in parallel way on the graphical processing unit. Parallel execution allow genetic programming in real time. Numerical experiments with real trajectories of the thrown tennis ball show that the algorithm can forecast the trajectory accurately.

Keywords: robotic catching, genetic programming, forecasting, machine vision, machine learning, CUDA, parallel computing

1. INTRODUCTION

Robotic throwing and catching of small objects is a novel possible way of material transportation. According to the proposals it may replace traditional conveyor belts in the industrial environment. Robotic transportation by throwing and catching is a complex task consisting of various subtasks. This paper addresses the subtask of tracking and forecasting the trajectory of the thrown object.

Robotic system for such work was first realized in 1991 by Hove and Slotine (1991). Since then, several systems have been constructed, and several articles have been published devoted to this problem. Most authors believed that it was a common robotic task that could improve the capabilities of robotic systems. Frank et al. (2006) proposed a practical application of robotic catching. According to this work, the system of throwing and catching may replace traditional conveyor systems for some transport tasks in the production environment.

The rate of success for most robotic catchers does not exceed 85%. There are two exceptions. First, Frank et al. (2012) have achieved successful catching 100% of the throws (50 of 50). The catching system was designed for catching cylinders with high aerodynamic stability. The authors acknowledge that their approach is not applicable for less stable objects. Cigliano et al. (2015) achieved 98% success. The grip used is much larger than in most other systems; its exact size is not specified in the article, but it seems that the catch will be successful even with an error of 10 centimeters. The results of existing systems show that the improvement of robotic catching is still necessary.

Here we present a novel approach to forecasting the trajectory of the thrown body. Traditional approach to this task consist in mathematical modelling of the ballistic trajectory. It is applied in most existing robotic catchers. Learning-based approach is less popular. It was used e.g. by Mironov et al. (2015). We mix these two approaches and propose the new one, which is based on genetic programming. Theoretical prerequisites for our solution was discussed by Gayanov et al. (2017). Here we present an application for real-time trajectory prediction. The forecast is made using an equation, which is learned by the procedure of genetic programming. Genetic programming defines the structure of the equation, while the values of the coefficients are defined by least square estimation. An equation has recurrent view: position of the object on each new frame is calculated based on the reference of its position on the previous frames. Unless other learning methods like k nearest neighbours or time delay neural networks, genetic programming does not require collecting the sampling of trajectories. The beginning part of the trajectory is used to train the predictor for forecasting the final part.

This article is organized in the following way. In the second section, we describe the procedure of extracting data about object trajectory from video. In the third section we discuss proposed prediction algorithm, while its GPU-executed implementation is developed in section 4. We present the

2405-8963
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2018.11.271

results of experimental evaluation in section 5, while section 6 incorporate concluding remarks.

2. EXTRACTING COORDINATES FROM VIDEO

Used tracking system is based on stereo vision. It is described in details by Mironov (2017). Two cameras with resolution 2048 by 2048 are used for tracking. Tracking algorithm provide real-time positioning of the flying spherical object in space.

Various stages of the algorithm are shown in figure 2. Image processing include the following stages:

- **Background subtraction.** This step is added in order to avoid false recognition of background objects as a ball. Experiments has shown that such false recognitions make tracking impossible when the distance to the object is high. This is illustrated in figure 1.

- **Edge detection.** Well-known algorithm by Canny (1986) is applied on this stage.

- **Defining pixel coordinates of the ball's center on the image.** Goetzinger (2015) has compared two techniques of circle recognition: Hough transform proposed by Hough and RANSAC proposed by Fischler and Bolles (1981). RANSAC was chosen: it provides the same accuracy with less time expenses.

- **Stereo triangulation.** Spatial position of the ball is calculated based on its pixel position on the images from different cameras.

For more efficient processing of the trajectory and for simplification of further calculations it may be useful to adjust the coordinate system with the parameters of the object's motion. According to the exploration by Mironov et al. (2015) three main transformations are used:

1. **Determining gravity-related coordinates.** This is a coordinate system, where one of the coordinate axes is parallel to gravity direction. From the point of the ballistic model gravity-related coordinates have better interpretability. Gravity direction is estimated by hanging a long pendulum in the field of view of the stereo system. When the pendulum is static it is directed downwards. Two distant points are picked on the pendulum nail and the gravity direction is defined as a vector between these points in 3D space.

2. **Defining the Plane of Flight (PoF).** Gravity is always directed downwards while drag is always directed opposite to the motion direction. If these are the only significant forces the whole trajectory of the ball will lie within a plane expressed by the gravitation axis and the horizontal projection of initial object velocity. This transformation will allow accurate prediction of the flight path.

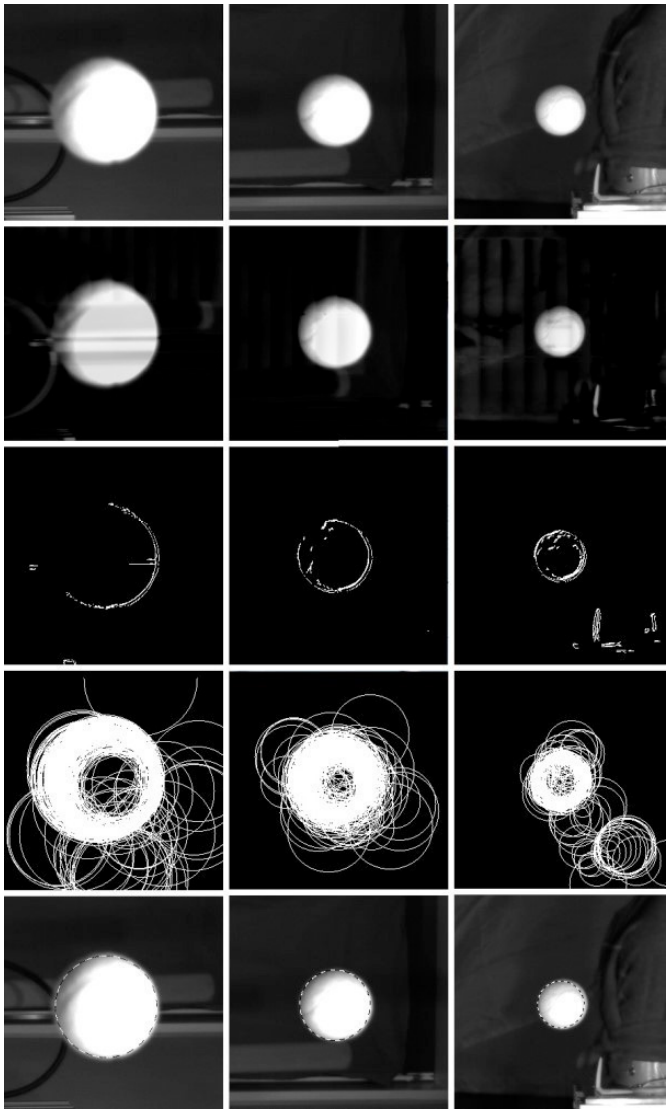

Fig. 1. Stage of circle recognition for three: original image (first row), result of background subtraction (second row), result of Canny edge detectiom (third row), random circles generated by RANSAC algorithm (fourth row), projection of the chosen circle on the original image (fifth row).

3. **Determining zero point.** One of the measured points is chosen as an origin for the final coordinate system. Due to this all trajectories will start from the zero.

3. FORECASTING ALGORITHM

Genetic programming (GP) is a specific application of common genetic algorithms. In GP chromosomes represent the structure of algorithms or equations. We propose to use GP for determining the equations, which represent the trajectories of a thrown body. Genetic Programming is an Evolutionary Algorithm. It works with a set of individuals (potential solutions), and these individuals form a generation. In every iteration, the algorithm evaluates the individuals, selects individuals for reproduction, generates new individuals by mutation, crossover and direct reproduction, and finally creates the new generation.

Unlike common optimization methods, in which potential solutions are represented as numbers (usually vector of real numbers), the symbolic optimization algorithms represent the potential solutions by structured ordering of several symbols. One of the most popular method for representing structures is the binary tree. A population member in GP is a hierarchically structured tree consisting of functions and terminals. The functions and terminals are selected from a set of functions (operators) and a set of terminals. For example, the set of operators F can contain the basic arithmetic operations: F = {+, −, ∗, /}; The set of terminals T contains the arguments for the functions. For example, T = {y, x, pi} with x and y being two independent variables, and pi represents the parameters. Now, a potential solution may be depicted as a rooted, labeled tree with ordered branches, using operations (internal nodes of the tree) from the function set and arguments (terminal nodes of the tree) from the terminal set.

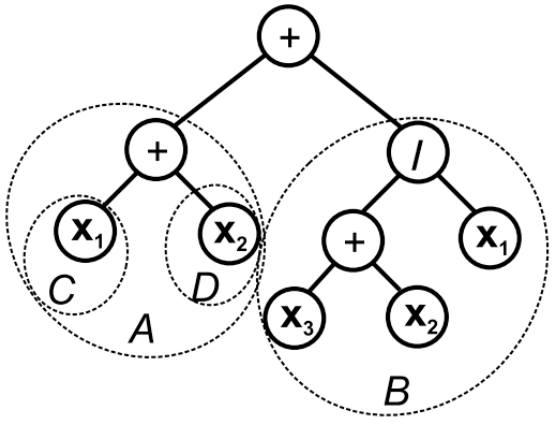

Fig. 2. Decomposition of a tree to function terms.

In the case of this example (fig. 2) the resulted parameter model is the following:

$$y = p_0 + p_1(x_3 + x_2)/x_1 + p_2 x_1 + p_3 x_3 \qquad (1)$$

GP can be used for the selection from special model classes, such as 9 polynomial models. To achieve this goal, one must restrict the set of operators and introduce some simple syntactic rules. For example, if the set of operators is defined as F = {+, ∗}, and there is a syntactic rule that exchanges the internal nodes that are below a '∗'-type internal nodes to '∗'-type nodes.

In the selection step the algorithm selects the parents of the next generation and determines which individuals survive from the current generation. The fitness function reflects the goodness of a potential solution which is proportional to the probability of the selection of the individual. Usually, the fitness function is based on the mean square error (MSE) between the calculated and the measured output values.

A good model is not only accurate but simple, transparent and interpretable. In addition, a complex over-parameterized model decreases the general estimation performance of the model. Because GP can result in too complex models, there is a need for a fitness function that ensures a tradeoff between complexity and model accuracy. Hence, suggests the incorporation of a penalty term into the fitness function.

To improve the GP algorithm, we suggest the application of Least Squares Optimization in the GP algorithm. During the operation of GP, the algorithm generates a lot of potential solutions in the form of a tree-structure. These trees may have terms (subtrees) that contribute the accuracy of the model.

The concept is the following: firstly, the trees (the individual members of the population) are decomposed to subtrees (function terms of the parameter models) in such a way it was presented in Fig. 2; then the error reduction ratios of these function terms are calculated; finally, the less significant term(s) is/are eliminated. The purpose of applying this approach in pruning a tree.

4. PARALLEL IMPLEMENTATION

To translate this into reality, we use the CUDA Toolkit. CUDA is an NVIDIA's parallel computing architecture, which will allow us to perform parallel computing not on the central, but on a graphic processing unit.

The calculations directly in the GPU perform so-called threads. The task of parallelizing a genetic algorithm using a graphics processor can be solved as follows: each member of the population is assigned its own thread (threads).

Algorithm of parallel genetic programming is the following:

Generate random population of P chromosomes
WHILE stop criterion not fulfilled DO
    DO PARALLEL FOR all individuals
        Estimate fitness function value
        Implement crossover and mutation
    Form new population
        Add the best new individuals
        Remove the worst old individuals

This approach will give a benefit in computing speed, since each thread computes the value of the fitness function for its individual, sends this value to the "main processor", and it already performs other necessary actions, such as selecting certain individuals for the next generation, calculating the fitness function for all individuals and so on.

To avoid excessive tree growth, the following tree size restrictions were introduced:

Maximum number of subtrees: 10.

Maximum tree depth: 3.

Also the following method, designed to reduce the size of the tree, is used. The subtrees that have a lesser effect on the accuracy of the model are excluded from the tree. The

concept is as follows: firstly, trees (individual members of the population) are decomposed into subtrees (functional terms of models with linear parameters), then error reduction coefficients of these functional terms are calculated; finally, less significant term (s) is excluded. This method of "trimming a tree" is implemented in each suitability assessment before computing tree fitness values. The main purpose of this approach is to convert trees to simpler trees, which are more transparent, but their accuracy is close to the original trees. This method always guarantees that elimination of one or more functional members of the model can be performed by "trimming" the corresponding subtrees, so there is no need for a structural rearrangement of the tree after this operation.

5. EXPERIMENTS

Within the experiments the implementation was able to represent the results same to results described by Gayanov et al. (2017). Initial part of each trajectory (first 50 frames) was used for learning the recurrent equation for this trajectory. This equation was used to forecast frames from 60 to 80. This procedure was applied for 20 trajectories from the data set. The results are presented in table I.

TABLE I. RESULTS OF NUMERICAL EXPERIMENTS FOR 20 TRAJECTORIES

#	***Equation***	***MSE, mm***
1	y(k-1) + (-0.038706) * (x(k-1)) + (0.033084)	6
2	y(k-1) + (-0.039329) * (x(k-1)) + (0.033553)	6
3	y(k-1) + (-0.038554) * (x(k-1)) + (0.031962)	5
4	y(k-1) + (-0.038689) * (x(k-1)) + (0.035645)	2
5	y(k-1) + (-0.038774) * (x(k-1)) + (0.033805)	7
6	y(k-1) + (-0.038546) * (x(k-1)) + (0.030245)	7
7	y(k-1) + (-0.038526) * (x(k-1)) + (0.032994)	4
8	y(k-1) + (-0.038740) * (x(k-1)) + (0.035160)	6
9	y(k-1) + (-0.038601) * (x(k-1)) + (0.033345)	3
10	y(k-1) + (-0.038454) * (x(k-1)) + (0.032473)	4
11	y(k-1) + (-0.038388) * (x(k-1)) + (0.031648)	6
12	y(k-1) + (-0.038501) * (x(k-1)) + (0.034544)	6
13	y(k-1) + (-0.038134) * (x(k-1)) + (0.033116)	9
14	y(k-1) + (-0.037313) * (x(k-1)) + (0.036494)	8
15	y(k-1) + (-0.038036) * (x(k-1)) + (0.032546)	4
16	y(k-1) + (-0.038347) * (x(k-1)) + (0.034728)	4
17	y(k-1) + (-0.038149) * (x(k-1)) + (0.034948)	3
18	y(k-1) + (-0.037972) * (x(k-1)) + (0.033884)	4
19	y(k-1) + (-0.037583) * (x(k-1)) + (0.032284)	7
20	y(k-1) + (-0.037841) * (x(k-1)) + (0.036221)	7

It may be seen that MSE do not exceed 10 mm. According to the three-sigma rule the errors of prediction do not exceed 30 mm with high probability. All equations generated by the algorithm are relatively simple and have the same type: linear dependence from both coordinates of the previous frame. At the beginning of learning genetic operations often generate more complicated equations. These equations may include coordinate values from several previous frames.

One of the questions is how many frames to use for forecasting. Theoretically, the accuracy of forecasting should improve with the increase in the number of frames for forecasting. Let us verify this statement on experimental data. We started the algorithm on 110 trajectories and calculated average deviation value and calculation time. The results are presented in table II.

TABLE II. INFLUENCE OF A NUMBER OF TAKEN FRAMES ON ACCURACY AND PERFORMANCE

Number of frames used for forecasting	*Mean deviation of the trajectory at the capture point, mm*	*Average calculation time, sec*	*Time remaining before arrival, sec*
10	102	0,223	0,74
20	52	0,227	0,66
30	45	0,231	0,58
40	15	0,235	0,5
50	9	0,246	0,42

It is important that the deviation of the trajectory does not exceed 30 mm, otherwise the object will not be caught, so the best option is to use 40 frames to predict the flight path.

6. CONCLUSION

In this paper we consider the task of forecasting the trajectory of the throwing object. We propose an approach based on use of genetic programming for trajectory forecasting. GP is used to generate the order and structures of the nonlinear model represented in the tree structure, while Least Squares is used to estimate the contribution of tree branches to the accuracy of the model. These methods are used to get a recurrence formula, where each next position of the object is calculated through the previous known ones. Successful application of the algorithm does not require preliminary training on already known trajectories. The results show high efficiency and rapidity in determining the flight trajectory of the object.

Genetic programming is executed on the graphical processing unit to decrease the time of computations. Within the numerical experiments real trajectories observed by the stereo vision system were estimated and predicted. Results showed that accuracy of trajectory forecasting is enough for successful robotic catching.

AKNOWLEDGEMENTS

Research work is supported by Russian Fund for Basic Research, grant #16-07-00243.

REFERENCES

Hough, P. (1962). A method and means for recognizing complex patterns, *U.S. Patent No. 3,069,654*.

Fischler, M. A., Bolles, R. C. (1981) Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, *Communications of the Association for Computing Machinery, Vol. 24, No. 6*, pp. 381 to 395.

Canny, J. (1986). A Computational Approach to Edge Detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6.*

Hove, B., Slotine, J.-J. (1991). Experiments in Robotic Catching, *American Control Conference, Boston, USA,* pp. 381 to 386.

Frank, H., Wellerdick-Wojtasik, N., Hagebeuker, B., Novak, G., Mahlknecht, S. (2006). Throwing Objects: a bio-inspired Approach for the Transportation of Parts, *IEEE International Conference on Robotics and Biomimetics, Kunming, China,* pp. 91 to 96.

Frank, T., Janoske, U., Mittnacht, A., Schroedter, C. (2012). Automated Throwing and Capturing of Cylinder-Shaped Objects, *IEEE International Conference on Robotic and Automation, Saint Paul, Minnesota, USA,* pp. 5264 to 5270.

Cigliano, P., Lippiello, V., Ruggiero, F., and B. Siciliano, B. (2015). Robotic Ball Catching with an Eye-in-Hand Single-Camera System, *IEEE Transactions on Control System Technology, Vol. 23, No. 5*, pp. 1657-1671.

Goetzinger, M. (2015) Object Detection and Flightpath Prediction, *Diploma Thesis, Faculty of Electrical Engineering, Vienna University of Technology*.

Mironov, K. V., Vladimirova, I. V., Pongratz, M. (2015). Processing and Forecasting the Trajectory of a Thrown Object Measured by the Stereo Vision System, *IFAC-PapersOnLine, Vol. 48, No. 11,* pp. 28 to 35.

Mironov, K. V. (2017). Transport by robotic throwing and catching: Accurate stereo tracking of the spherical object, *2017 International Conference on Industrial Engineering, Applications and Manufacturing, St.-Petersburg, Russia.*

Gayanov R. C., Mironov, K. V., Kurennov D. V. (2017). Estimating the Trajectory of a Thrown Object from Video Signal with use of Genetic Programming, *2017 IEEE Symposium on Signal Processing and Information Technology, Bilbao, Spain,* pp. 134 to 138.