Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 48-11 (2015) 028–035

Processing and Forecasting the Trajectory of a Thrown Object Measured by the Stereo Vision System

Konstantin Mironov * **Irina Vladimirova** ** **Martin Pongratz** *

* *Institute of Computer Technology, Vienna University of Technology Vienna, Austria (e-mail: {mironov,pongratz}@ict.tuwien.ac.at).*
** *Department of Technical Cybernetics, Ufa State Aviation Technical University, Ufa, Russian Federation (e-mail: ooo-flaming@mail.ru).*

Abstract: Transportation of small-sized rigid objects in industrial environment may be provided by throwing it from the source point and catching it at the destination point. This approach promises better flexibility than traditional transportation systems based on conveyor belts. Accurate real-time forecasting of the object ballistic trajectory is necessary to provide successful catching of the object by the gripper. The development of a sample-based algorithm for trajectory forecasting is a scope of this paper. The input for the forecast is a reference of object spatial coordinates measured by the stereo vision system. Such measurements allow defining the position of the object in a camera-related coordinate system with millimeter accuracy, however they sometimes include outliers. A reference of coordinate transformations is proposed, which translates object coordinates from the camera related 3D system to a 2D system with relations to gravity and motion direction. Outlier detection is made during these transformations. The forecasting is performed in 2D coordinate system with use of k nearest neighbors approach. Applying the algorithm to the measured trajectories showed that it is able to predict future position of the object with 3 centimeters precision in 92 % of cases.

Keywords: Ballistic motion, Machine learning, Motion estimation, Nearest neighbors, Prediction methods, Robot vision, Target tracking.

1. INTRODUCTION

Transport-by-throwing (TbT) is a novel approach for transportation of small-sized rigid objects, especially in industrial environment (where there is a need to transport a high number of objects between machine tools). It was introduced by Frank et al. (2006). According to this approach the transportation of an object from a source point to a destination point is provided via throwing it at the source point (towards the destination point by the specific throwing device) and catching it at the destination point (by the specific capturing device). Throwing-based transportation systems have potential advantages in comparison with traditional conveyors: better flexibility, higher productivity and lower consumption of energy and resources (Frank et al. (2006)).

While the task of the throwing device is exact - it must throw objects everytime with fixed direction and velocity - the catching challenge is more complicated: the gripper must catch airborne object in a priori unknown interception point at a priori unknown time moment with a priori unknown velocity. The task of catching an airborne object with a robotic manipulator was considered prior to the foundation of TbT as one of common robotic activities. It was introduced by Hove and Slotine (1991). To define catching time, position, and velocity the trajectory of the object is predicted based on the observation of its flight. While most existing approaches on trajectory prediction are based on physical models of the object motion, here we apply a sample-based predictor that does not require exact knowledge about physical parameters of the trajectory. Our predictor uses a modified version of the algorithm proposed by Mironov et al. (2014). The development and validation of the proposed ideas is implemented with tennis ball as a thrown object. The reason is that the tennis balls is a well-known aerodynamic object; there is a number of scientific works exploring the aerodynamic properties of the tennis ball (the review on such exploration is given e.g. by Mehta et al. (2008)). On the other hand most of existing robotic catching applications also deal with small-sized sport balls, e.g. catchers by Hove and Slotine (1991), Frese et al. (2001), Birbach et al. (2011). On the other hand the basic principle of prediction should be independent from the spherical shape of the body and may be extended to the objects of another shapes.

This article is organised in the following way. In the second section quick overview of existing techniques for ballistic trajectory prediction is given. In the third section the collected database of trajectories used for learning and validation is discussed. In the fourth section coordinate transformation reference is discussed. Coordinate transformations are made in order to increase accuracy of prediction, to decrease the complexity of further calculations (e.g. decreasing dimensionality of the data), and to make use of trajectory database more efficient. The last subsection of section 4 is concentrated on prediction

10.1016/j.ifacol.2015.09.155

of object movement in transformed coordinate system. In section 5 evaluation results for the whole algorithm based on the dataset are presented and discussed. In section 6 concluding remarks are given.

2. RELATED WORKS AND CONTRIBUTION

In the first robotic ball catcher developed by Hove and Slotine (1991) parabola fitting was used to estimate and predict the trajectory of the thrown balls. Parabolic motion models were also applied in recent robotic catchers developed by Herrejon et al. (2009) and Batz et al. (2010). In such models gravity is considered as the only significant force, which influence on the flying object, and acceleration of the object is considered as equal to gravity acceleration while the object is airborne:

$$\ddot{X} = \{-g, 0, 0\} \tag{1}$$

where $X = \{x_1, x_2, x_3\}$ is a vector of object coordinates (and axis x_1 is directed upwards), g is gravitational acceleration. The solution of this equation is a well-known formula of motion with constant acceleration:

$$X = X(0) + \dot{X}(0)t + \{-g, 0, 0\}\frac{t^2}{2} \tag{2}$$

where t is time. This model does not consider air drag which may be neglected on short distances and become a significant force on the long term (see, e.g. the simulation of ballistic motion by Tutz (2007)). Further physical models of ball motion, considering gravity and air drag as significant forces were applied e.g. by Frese et al. (2001), Barteit et al. (2008), Birbach et al. (2011). In such models the object coordinates at the certain time moment after the throw may be calculated by solving the following differential equation:

$$\ddot{X} = \{-g, 0, 0\} - k|\dot{X}|\{\dot{x_1}, \dot{x_2}, \dot{x_3}\} \tag{3}$$

where k is a coefficient defining air drag. Unlike (1) this equation has no analytical solution and is usually solved numerically. Such a model is also simplified, however allows achieving 66-80% rate of success in catching (Frese et al. (2001), Birbach et al. (2011)).

Increasing precision of the motion model lead to increasing complexity of calculations and to the need of more complicated experiments in order to define aerodynamic properties of the object. Even for simle-shaped objects like tennis balls complicated experiments in aerodynamic tube are necessary to define drag coefficients (Mehta et al. (2008)). Accurate aerodynamic model of the object is much dependent on its shape. Learning-based methods draw attention as a potential way for trajectory prediction because they do not need exact knowledge about object physical properties to work correct. Kim and Billard (2012) propose the predictor based on process model with parameters obtained by means of machine learning. This approach lies between analytical and learning-based prediction: prediction is performed by integrating tangential and angular acceleration of the object, while these accelerations are estimated based on learning. Later Kim et al. (2014) apply this concept for catching airborne objects with robotic manipulator and achieved 73% success rate. The learning based approaches were introduced by Mironov and Pongratz (2013) - neural network prediction, - and by Mironov et al. (2014) - nearest neighbors.

In this paper we extend and apply the predictor introduced Mironov et al. (2014). Observed trajectory of the object is compared with sample trajectories from the database. The input of the predictor includes the reference measurements of the current trajectory $C(1 : m) = \{X_c(1), X_c(2), ...X_c(m)\}$ where m is a number of frames captured by the observation system till the moment when prediction is made. The database include N trajectories $S_1, S_2, .., S_N$ where each trajectory $S_i(1 : n) = \{X_i(1), X_i(2), ..., X_i(n)\}$ and $n > m$ is overall number of frames. The predictor task is to calculate a forecast of the current trajectory $C(f : n) = \{X_c(f), X_c(f), ...X_c(n)\}$ where $m < f < n$. This task is solved in the following way. Two trajectories $A(1 : n) = \{X_a(1), X_a(2), , ...X_a(n)\}$ and $B(1 : n) = \{X_b(1), X_b(2), ..., X_b(n)\}$ are taken from the database such that measured points of the current trajectory $C(1 : m)$ lie higher than corresponding points of trajectory $B(1 : m)$ and lower than corresponding points of trajectory $A(1 : m)$ and that the distances from trajectories $A(1 : m)$ and $B(1 : m)$ to $C(1 : m)$ are smaller than from any other trajectories from the set. Distance is defined as a mean value of euclidean distances between corresponding points of trajectories:

$$D(A, C) = \frac{1}{m}\sum_{i=1}^{m}|A(i) - C(i)| \tag{4}$$

The forecast $C(f : n)$ is calculated as a weighted mean of $B(f : n)$ and $A(f : n)$:

$$C(f : n) = w_a A(f : n) + w_b B(f : n) \tag{5}$$

Weights w_a and w_b are defined according to the distances from A and B to C. Two extensions of the algorithm were proposed by Mironov et al. (2014). First it was proposed to sort the trajectories in the dataset with respect to launching velocity (or launching angle, or any other parameter characterizing the shape of trajectory) and take in mind only such trajectories from the database that have the same value of thus parameter as C. This allow decreasing the volume of calculations as the current trajectory is compared only with a small subset of the dataset. Secondly, it was proposed to translate the 3D trajectory coordinates into 2D "Plane-of-Flight (PoF)". Goals of this transform are discussed more precisely in subsection 3.2.

Mironov et al. (2014) explored the theoretical usefulness of the kNN approach via the numerical simulation of the object flight. This simulation was including a number of simplifying assumptions. Here we extended the prediction algorithm in order to overcome these assumptions. The main advances in comparison with the work by Mironov et al. (2014) are listed below.

- The model of the object motion based on (3) was used for simulating trajectories. Validation of the algorithm was done based on this simulation. Here we validate our algorithm by forecasting real trajectories of the thrown balls observed by the vision system.
- It was assumed that the center of the coordinate system coincide with the starting point of the object flight and coordinate axis x_1 is directed upwards.

Therefore all trajectories in the database have the coordinate value $X(0) = \{0, 0, 0\}$. Estimating the PoF coordinates in this case is simplified: it is equal to estimating the proportion a between coordinates values in x_2 and x_3. This proportion is equal to the tangent of angle α between the Plane-of-Flight and plane expressed by x_1 and x_2. In real environment we deal with object location in camera coordinate system, which has no relation to the gravity direction and to the launching point. In this article we introduce a reference of coordinate transformations which translate object trajectory into coordinate system which is, first, related to the gravity direction, secondly, related to the Plane-of-Flight, third, provide that $X(0) = \{0, 0, 0\}$ for all trajectories. The transformation sequence include detection and filtering object positions measured with high errors. Such transformation reference is invariant to the location of the cameras (if such location allow observation and stereo triangulation of the whole trajectory) and allow creating the common database even if the trajectories were acquired with different camera setup.

- It was assumed that the small subset of the trajectories with similar shape-characterizing parameters is already allocated. Here we deal with the situation when the values of these parameters must be estimated from the trajectory measurements. The subset is allocated from the dataset in real time.

3. TRAJECTORY MEASUREMENTS

Most of existing robotic catchers use stereo vision to observe the 3D position of a flying ball (e.g. Hove and Slotine (1991), Frese et al. (2001), Batz et al. (2010), Birbach et al. (2011)). As an exception Barteit et al. (2008) and Herrejon et al. (2009), use single camera setup to track the flying ball. Ball positioning in these works was based on physical model of the ballistic motion, which is not likely if we try to develop sample-based predictor without relations to physics. Stereo vision allows to determine position of the ball center in 3D space based on its pixel positions on the synchronized images from two cameras. The knowledge about these positions may be acquired using circle detection algorithm based on Hough transformation (Scaramuzza et al. (2005), Barteit et al. (2008)). Accuracy of positioning static untextured spheres using stereo vision was evaluated by Pongratz and Mironov (2015). A pair of cameras with a resolution of 2048 by 2048 pixels were examined (the same cameras are used in experiments described in this paper). The standard deviation of object positioning was found independent from distance to the cameras (when it is more than 0.5 and less than 2 meters) and lies within 2-3 mm for considered setup. This research is concentrated on the static objects and does not consider motion blur, possible visual collisions with other objects on the scene, and specific texture of the tennis ball. Exposure time of the camera sensor was set to the minimum possible value of 41 μs. If the object is flying with velocity of 10 m/s its shift after 41 μs be equal to 0.41 mm. Therefore the influence of the motion blur was considered negligible.

In our throwing experiments a specific throwing device was used to throw the balls with known force and velocity.

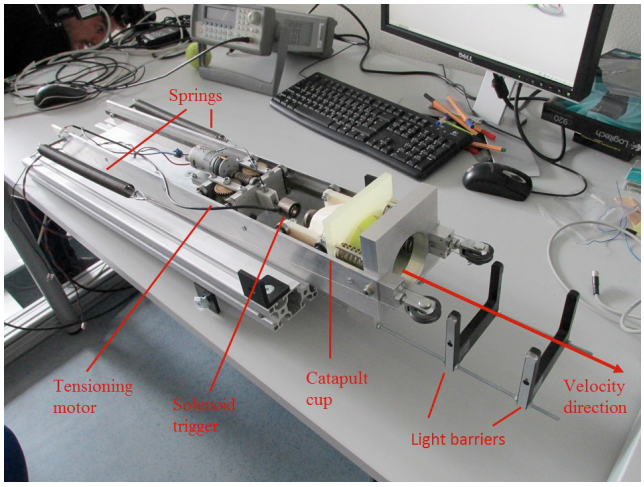

Fig. 1. Throwing device used in the experiments

The photo of this device is shown on figure 1. The device is controlled from PC via the serial port. The objects (tennis balls) are thrown by the spring catapult. An electromechanical drive is used to achieve needed tension of the springs. It is connected with the catapult cup by the solenoid magnet. When the operator switches off the magnet the connection with the cup is interrupted and the catapult throws the object. Two light barriers are mounted in front of the cup, which measure the time when the object flies through them. The angle of throw is near to 60 degrees upwards.

The cameras are mounted with 1 meter baseline and positioned opposite to the throwing device. The distance between camera baseline and the catapult is approximately 3.2 meters. Exploration on the performance of the camera system showed that acquiring and processing double 2048 by 2048 images provided by the sensors of the cameras is time-expensive and limit the feasible frame-rate to 40-50 fps. Therefore only the small area of interest (AOI) within the image with the size of 300 by 300 pixels is searched for the object by the recognition algorithm. Position of the AOI on the image is preliminary aligned to the launching area around the throwing device and then recalculated after receiving each new frame according to the new position of the object. This approach reduces computational expenses and increase available frame-rate up to 110 fps.

Using the described setup 183 trajectories of the tennis balls were acquired. The tension of the spring was mostly adjusted to provide the launching velocity of 4.5 m/s. However the measured velocity deviated from this nominal value varying from 4.3 to 5.1 m/s. In addition a few throws with lower or higher tension were made, therefore the overall range of launching velocities is 3.7-5.3 m/s. Each trajectory includes 80-95 frames until the object leaves the field of view. Each frame is processed by the Canny edge detection algorithm first, then Hough transformation is applied to determine the center position of the ball in the images. Then the stereo triangulation procedure is applied to ball center positions in order to define 3D coordinates of the object center.

In most of the cases the reference of the center positions has an appearance of the smooth curve (figure 2.a). However in some cases strong deviations in trajectory was

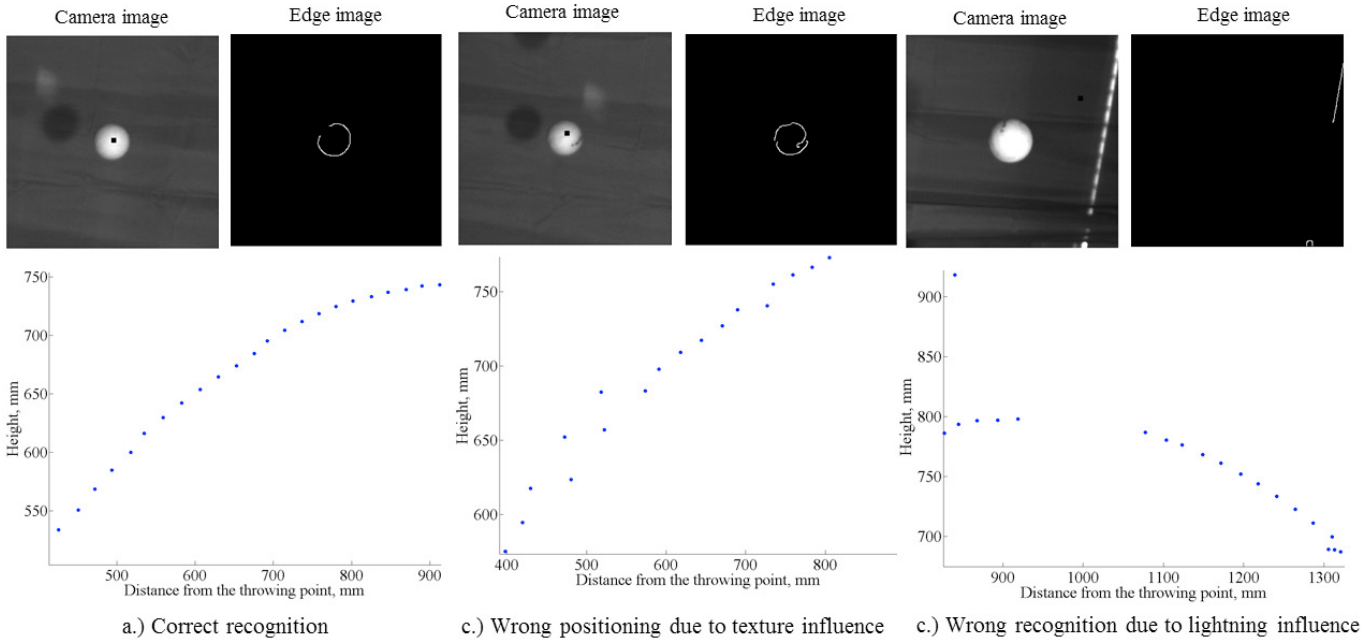

Fig. 2. Illustration of the effects that distort object positioning.

detected. The appearance and nature of such deviations are shown on figure 2.b and 2.c. The texture (printed text and curve lines) on the surface of the ball may be recognized as a part of ball edge. This leads to the errors in center positioning which achieve more than 10 mm (figure 2.b). Much bigger errors in positioning are connected with the influence of scene objects. The edges of the objects on the scene (e.g. light sources, figure 2.c) may also be recognized as edges of the ball. Three trajectories in the set have artificially added background noise (the light-sources were switched on) which leads to wrong positioning.

4. PREDICTION IN MODIFIED COORDINATES

The stereo vision tracker allows to measure position of the object $X_c = \{x_{c1}, x_{c2}, x_{c3}\}$ in the coordinate system connected with optical center of the camera. However for a more efficient processing of the trajectory and for simplifying further calculations it may be useful to adjust the coordinate system with the parameters of the object motion. The reference of coordinate transformations aiming such simplification is discussed in this section. Three main transformations are applied: first of all gravity-related coordinates $X_g = \{x_{g1}, x_{g2}, x_{g3}\}$ are defined (the coordinate axis x_{g1} is collinear with gravity direction), then coordinates are transformed from 3D space into 2D Plane-of-Flight (PoF) $X_p = \{x_{p1}, x_{p2}\}$ and after that one of the measured points of the trajectory is picked as a zero point for the coordinates $X_s = \{x_{s1}, x_{s2}\}$. Relations between the coordinate systems are shown on figure 3. Forecasting operation is made in the zero-point related coordinates.

4.1 Gravity-related coordinates

Mironov et al. (2014) consider that trajectories are represented in such coordinate system that one of the coordinate axes (x_1) is collinear with gravity direction. The first transformation aims to make this assumption true. The

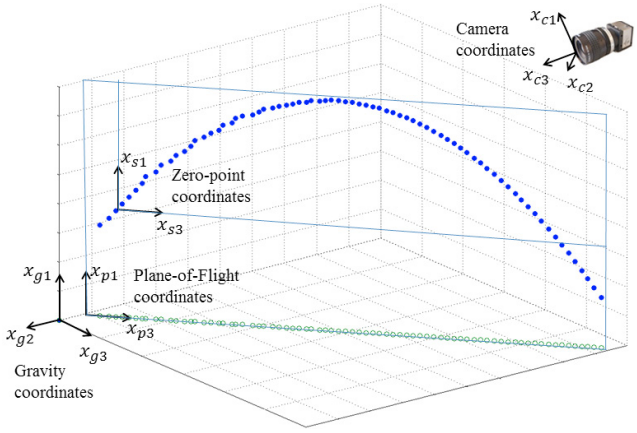

Fig. 3. Relation between the coordinate systems

main motivation of this step is to simplify the estimation of the Plane-of-Flight (PoF). As it is shown in the next subsection, the definition of the PoF in a gravity-related coordinate system is a task of line fitting in the plane, while the definition of the PoF directly from the camera coordinate system is a more complicated task of plane fitting in 3D space. From the point of the ballistic model gravity-related coordinates have better interpretability as gravity acceleration g is localized in dimension x_{g1}. The dependencies of coordinates x_{g2} and x_{g3} from time are monotonous functions and the dependency of coordinate x_{g1} from time is a convex function.

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. The direction of horizontal axes is not an important factor on this stage, however for better interpretability x_3 was aligned with the nominal direction of throw.

4.2 Defining the Plane of Flight

Gravity is always directed downwards while drag is always directed opposite to the motion direction. If these are the only significant forces (i.e. there are no forces directed to the side from the motion direction) the whole trajectory of the ball will lie within a plane expressed by the gravitation axis x_{g1} and the horizontal projection of initial object velocity. In case of rotating object the Magnus force exist, which may curve trajectory to the side; however no significant influence of any side-directed force was detected during the throwing experiments. Therefore 3D object coordinates may be transformed to a 2D coordinate system defined by these two vectors. Mironov et al. (2014) proposed to forecast object coordinates within this 2D "Plane of Flight" (PoF) and then translate the results back to the 3D coordinate system. PoF representation has the following advantages in comparison with a 3D representation:

- Decreasing volume of the data. Less items are needed to represent trajectories and less volume of calculations is needed to process it.
- More efficient use of the database. The nearest neighbors prediction is based on searching the database for the trajectories that are similar to the current one. When trajectories are compared in 3D space two trajectories with various horizontal directions can not be considered as neighbors because the distances between corresponding points are high. In PoF representation trajectories with various horizontal directions could be also considered as neighbors. Simulation by Mironov et al. (2014) showed that accuracy of prediction in PoF is higher than in 3D.
- Outlier detection. Good fitting of the points to the PoF shows that the measurement of these points is accurate. If the fit of certain point to the plane is bad this point may be marked as an outlier and rejected from the further calculations.

The question is how to define PoF from the available data. In common case a 2D plane in 3D space may be expressed as a dependence of one coordinate from two others. Such dependence has three independent parameters marked as a, b, and c:

$$x_2 = ax_3 + bx_1 + c \tag{6}$$

The task of plane fitting in 3D space mean the determination of these parameters from the available data. As the PoF is vertical in the gravity-related coordinate system, one of its axes is equal to x_{g1}. Therefore the parameter b is equal to zero and the task simplifies to line fitting in 2D space: the position of the second axis with respect to x_{g2} and x_{g3} must be estimated:

$$x_2 = ax_3 + c \tag{7}$$

Estimation of a and c may be done using various methods. Such method should have the following properties:

- Accuracy. It is hard to determine the accuracy of the fit as no ground-truth data about object coordinates is available. If the points are translated to the estimated PoF and then back to 3D, their coordinates will vary from the original value. It is hard to distinguish whether such a variation is a result of algorithm inaccuracy or the correction of measurement errors. According to Pongratz and Mironov (2015) there is no significant systematic error in positioning spheres on a long distances and the deviations have random appearance. The wrong estimation of a and c leads to appearance of a systematic component in the coordinate differences. Hence, if there is a significant systematic errors in the PoF transformation the estimator can be marked as inaccurate.
- Robustness. The algorithm must work correct even if there are a few points with wrong coordinates (like shown on figure 2.a and 2.b). The algorithm should allow detecting such outliers and rejecting them from the further calculations.
- Stability. The estimated values of a and c should not vary strongly depending on what points of the same trajectory were used for estimation. Especially the estimated values of the parameters should be the same when the whole trajectory $\{X_g(1), X_g(2), ..., X_g(n)\}$ is used for estimation and when only the input part $\{X_g(1), X_g(2), ..., X_g(m)\}$ is known.
- Performance. As the application is real-time, all calculations must be done within the fixed period.

Several estimation methods were applied to determining a and c: least squares fitting (LS), robust least squares (RLS), random sample consensus (RANSAC), mean calculation, median calculation. LS showed high speed and sufficient stability. It works accurate enough on the trajectories that do not contain outliers (errors seem to be randomly distributed around zero value, they are usually less then 2 mm). However the robustness of LS is very low: systematic errors achieve several tens centimeters for the trajectories with false recognitions. Application of RLS (we used the implementation from MATLAB (2012) curve fitting toolbox) improve the tolerance to outliers. However on the trajectories where the number of outliers is more than 5 it also does not work correctly. The stability and accuracy on the outlier-free dataset has the same quality as for LS, while time expenses increase drastically (in MATLAB environment RLS estimation of the trajectory takes about 20 ms, while LS takes 1-2 ms).

RANSAC PoF estimation is based on picking two random points on the trajectory and checking whether the vertical plane including these two points coincide with the PoF. For this purpose the distance from all the points to the candidate PoF is calculated. The points for which the distance does not exceed certain threshold (three times the standard deviations in static positioning defined by Pongratz and Mironov (2015) was taken as such threshold; this value is equal to 9 mm) are marked as inliers, other points are marked as outliers. If the percentage of outliers does not exceed certain limit (variations of the limit from 10% to 50% were checked) these outliers are removed from the set and the values of a and c are estimated based on remaining points using LS method. If the number of outliers exceed the limit, the whole operation is repeated. RANSAC implementation showed better robustness than LS, however the accuracy and stability of the estimation is much worse. Estimated values of a and c have oscillations about 20% in various runs of the algorithm. This lead to differences up to 1 centimeter in the estimated coordinates of the object.

RANSAC algorithm picks the points randomly, which allows processing even large ammounts of data fast. On the other hand the number of points in one trajectory is relatively small (less than 100). Therefore a deterministic way of picking the pairs of points is proposed, as it is not very calculation-expensive in this case. If the available part of the trajectory is $\{X(1), X(2), ..., X(m)\}$ where m is even number, then $m/2$ pairs of points are picked: $\{X(1), X(m/2+1)\}$, $\{X(2), X(m/2+2)\}$, ..., $\{X(m/2), X(m)\}$ and $m/2$ hypotheses about the values of a and c are made. The most simple way of estimating PoF from this hypotheses is calculating mean values of a and c. Unexpectedly accuracy and stability of such estimation is not worse then for LS. To make the estimation more robust it is done twice. After the first iteration outliers (defined in the same way as in the RANSAC solution) are removed from the set and mean calculation procedure is repeated. The robustness of this algorithm is not worse than for RLS, however it also can not deal with the trajectories where the rate of false recognitions is high. In this case the value of systematic error achieve several centimeters.

If the median values of a and c are taken instead of mean values the robustness of the estimation increases. The median is much less distorted by the outliers then the mean: in the sorted set outliers lie in the beginning or in the end of the list, while median value lies most probably between inliers. Therefore median calculation is chosen as a method for estimating PoF. It is accurate (errors seem to be randomly distributed around zero value, they are usually less then 1 mm, only for 1-2% of the cases difference exclude 2 mm), robust (accuracy stays the same even when the number of outliers is about 20% of the whole dataset), stable (values of a and c do not vary more than in second digit depending on is $X(1:n)$ or only $X(1:m)$) and faster than any other method (execution in MATLAB take about 1 ms for processing the trajectory consisting of 80 points).

Good accuracy and stability of median PoF estimation also proves that PoF model is valid and there is no significant side-directed force influencing on the thrown object. If the horizontal projection of the trajectory is not a straight line, errors would be dependent from the time and the parameters of the PoF would vary depending on m. As median fit does not include such effect the influence of sideforce is considered as negligible. These results show that PoF transformation is valid for spherical objects thrown by linear launching device. Applicability of this concept for other objects depends on the shape of these objects. As the TbT on the current stage is considered as an approach for transporting the compact objects (with properties near to the point mass) thrown without a spin, the influence of sideforce on the distance of several meters should be small for such objects. More careful evaluation of sideforce influence for various objects is a part of future work.

4.3 Determining zero-point

One of the assumption made by Mironov et al. (2014) that allows using nearest neighbors solution is that there is certain zero time moment when coordinates of the objects from all trajectories are equal to zero. This point in theory corresponds to the launching point. In practice it seems not feasible to position the launching point accurately as it lies outside of the cameras' field of view and there is no confidence that it is every time the same. In order to have common zero coordinates for all the trajectories we pick one of the points $X(i)$ (zero-point) in the beginning of the trajectory and subtract its coordinates from next points of this trajectory:

$$C_s = \{0, X_p(i+1) - X_p(i), ..., X_p(n) - X_p(i)\} \quad (8)$$

The following heuristic is proposed for choosing zeropoint $X(i)$. As it should be the point measured with relatively good accuracy monotony and convexity of the trajectory around this point is examined. The validity of the following statements is checked:

- The coordinate values for $X(i)$ in both dimensions are higher than for $X(i-1)$ and lower than for $X(i+1)$. This means that the dependence of x_1 on time is monotonous (growing) around $X(i)$.
- The object velocity in vertical dimension after $X(i)$ is less than before: $x_1(i+1) - x_1(i) < x_1(i) - x_1(i-1)$. This means that the dependence of x_1 on time is concave around $X(i)$.

First point on the trajectory that satisfies these statements is taken as a zero point. This heuristic does not guarantee that chosen point is measured very accurate however it allow rejecting bad points.

4.4 Sorting and prediction

A light-weight robot (KUKA LWR 4+) was proposed by Pongratz et al. (2013) as a capturing device. In preliminary experiments it was defined that it need about 30 ms (which correspond to 4 frames of the vision system) to accelerate to the maximum speed. This means that catching movement will take approximately 50-70 ms (about 9 frames). It was assumed that data frames from 10 to 40 may be used to predict object coordinates on frames from 60 to 75, which correspond to 180 ms time limit for the catching movement.

Mironov et al. (2014) propose to sort the trajectories in the database with respect to a certain parameter p and compare the current trajectory C only with those trajectories from the set, which have a similar values of p as C. Three parameters of the trajectory were suggested to be taken as p: angle of throw α, launching velocity v, horizontal projection of the velocity v_h. Here we sort the trajectories with respect to v_h. This parameter is preferable as it can be estimated directly from zero-point coordinates. It corresponds to the change of object coordinate x_{s3}. Another advantage is that v_h does not change its value much in comparison with v and α. The gravity does not effect directly on the horizontal velocity and the influence of air drag is relatively small on a scale of several tens milliseconds (see e.g.Tutz (2007)). Therefore the estimate of v_h could be calculated as a mean of its measurements from several neighboring frames.

The mean horizontal velocity after i frames can be calculated using the following equation:

$$v_h(1:i) = \frac{x_{s3}(i)}{i-1} \quad (9)$$

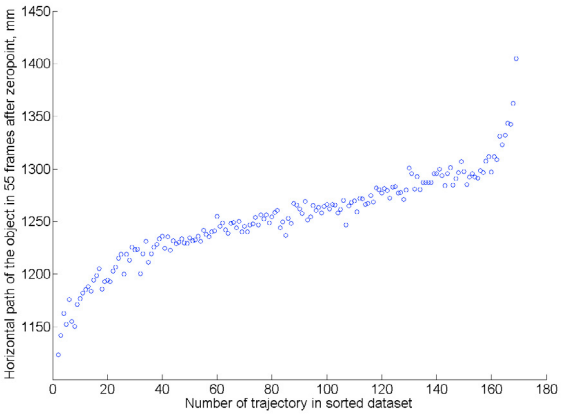

Fig. 4. Dependence between the sequential number of trajectory in the sorted dataset and the path made by the object in 55 frames after the zero-point

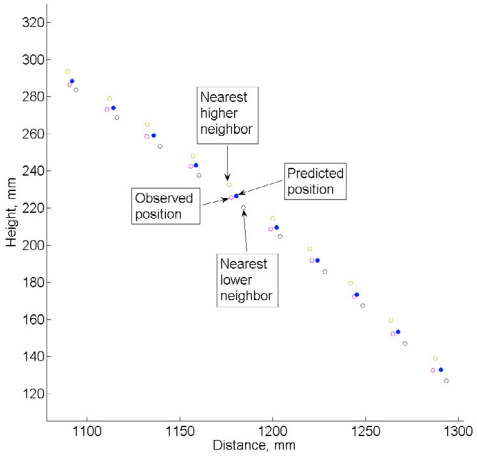

Fig. 5. Reference positions of the ball from the current trajectory, its lower neighbor, and its higher neighbor; filled circles show the predicted positions of the object

If $v_h(1:i)$ is used to characterize trajectory, it is better if the value of i is big, as the influence of measurement errors on the estimate is low in this case. Therefore the last frames available before prediction are used to determine horizontal velocity. The estimation of p is done in the following way:

$$p = \hat{v}_h(1:37) = \frac{1}{5}\sum_{i=35}^{39}\frac{x_{s3}(i)}{i-1} \tag{10}$$

Each element of the sum represent the mean object velocity on the interval from the first frame to frame number i. Therefore p roughly corresponds to the mean value of v_h on the interval from the 1st to the 37th frame. Due to the influence of the air drag the value of v_h is not constant, however it can be assumed that if the object from trajectory A has value of v_h similar to object from trajectory B in the beginning of flight they will have similar horizontal coordinates in the end of flight. The illustration of this assumption is given on figure 4. Here the horizontal positions of the objects in zero-point coordinate system in 55 frames after the zero-point are plotted with respect to sequential number of the trajectory in the dataset, where trajectories are sorted with respect to $\hat{v}_h(1:37)$. It can be seen that the difference in horizontal positions for the neighboring trajectories with similar sequential numbers (i.e. with similar values of $\hat{v}_h(1:37)$) does not exceed several centimeters (at least for trajectories with sequential numbers from 20 to 160). Trajectories with sequential numbers less than 20 and more then 160 was acquired, when the throwing velocity was set lower or higher then 4.5 m/s. There were not much such cases hence small difference in sequence number correspond to high difference in $\hat{v}_h(1:37)$ than for the trajectories in the middle of plot.

Prediction itself is implemented in the following way. Candidate trajectories A and B are picked among the trajectories that have nearest values of p to the current trajectory. The number of such trajectories (the size of the cluster according to Mironov et al. (2014) terminology) can vary from 3 to the size of complete database. If there are no higher or no lower values within the cluster they are searched outside the cluster and first one, which is lower or higher, is considered as a neighbour. The forecast is then done according to (5). Two versions of the algorithm were evaluated: simple kNN (both w_a and w_b are set to 0.5) and weighted kNN (w_a and w_b are defined as the mean proportion of the distances from A and B to C; this definition was proposed by Mironov et al. (2014)).

5. EVALUATION

For evaluating the usefulness of the proposed solutions the work of the predictor is applied to the trajectories acquired during the throwing experiments. Prediction for each trajectory is done based on other trajectories from the set. Trajectories that have significant outliers were not used in prediction of other trajectories (i.e. the learning algorithm removes trajectories with outliers from the set). Prediction of trajectories, which were acquired with unusual value of spring tension were not considered (it is assumed that such values are set on the throwing device only on learning stage to provide that all trajectories generated with nominal throwing setup will have higher and lower neighbors).

Totally the prediction of 150 trajectories was checked using this setup, while the size of the set used for prediction was 168 trajectories. The exploration about the cluster size showed that it seems useless to make it more than 20 trajectories: the numbers of A and B do not change if the cluster size is more. On the other hand the time needed to perform prediction increases drastically with increase of cluster size. In MATLAB environment it takes 3 ms to perform the forecasting with 20 trajectories in cluster, 9 ms with 60 trajectories in cluster, and 24 ms with 140 trajectories in cluster. Therefore the cluster size of 19 trajectories was found to be optimal in the current conditions. An example plot of successful trajectory prediction is shown on figure 5. Here the references of measured positions for trajectories A (yellow circles), B (dark circles) and C (magenta circles) and respective predicted positions of C (blue filled circles) are plotted. The difference between respective blue filled dots and magenta dots represent the error of prediction. It can be seen that the distance between observed and predicted position is small absolutely (does not exceed several mm) and in comparison with distances to respective points of A and B.

The results of prediction accuracy evaluation on the dataset are shown in table 1. The error here is defined as the euclidean distance from the predicted position of the object at the certain time moment to its measured position at the same moment. The term "error $<$ 20 mm" means that there is no cases along the trajectory, when the difference between observed and predicted coordinate is more that 20 mm.

Table 1. Accuracy of the prediction

Parameter	simple kNN	weighted kNN
% of throws with error $< 30mm$	92	85
% of throws with error $< 20mm$	85	73
Median error in mm	10	13

The use of 20 and 30 mm thresholds is based on the exploration of Birbach et al. (2011) where a precision from 20 to 30 mm is found to be sufficient for successful grasping of a ball by the robotic hand. If the object is caught passively, e.g. into a basket or cup, allowed error could be higher and depends on the basket size; such setup was applied e.g. by Frese et al. (2001). The results show that the proposed algorithm has relatively good precision of the forecast. Unexpectedly simple kNN works more accurate than weighted kNN. The reason could be that the distances between trajectories are sensitive to the small positioning errors. The reason for the most of erroneous cases is either fringe effect (higher errors when the velocity of the object is near to maximum or minimum values) or the situation of the wrong choose of the neighbor trajectories. Within the future work the accuracy may be improved by collecting larger dataset consisting of trajectories with various velocities and direction and improving the strategy of nearest neighbor choice.

6. CONCLUSION

An algorithm of estimating and forecasting the ballistic trajectory of a flying ball is developed and evaluated based on stereo vision measurements. The reference of coordinate transformations allows reducing size of the data, detecting outliers, increasing efficiency of processing the database. The algorithm of nearest neighbors trajectory prediction in transformed coordinates is also evaluated. Nearest neighbors choice based on similarity of the horizontal velocity and on picking one higher and one lower trajectory shows good accuracy of prediction and lower time expenses in comparison with taking in mind only the distance between trajectories. The evaluation of proposed forecasting algorithm shows that it is able to achieve 2 cm prediction accuracy in 85% of cases and 3 cm prediction accuracy in 92% of cases. These results are an intermediate step in the development of a robotic ball catcher. Next steps include integration of the algorithm into the robot control in order to provide the information about ball future position to the robot motion generator and exploration of the algorithm applicability to the objects with various shapes.

ACKNOWLEDGEMENTS

The research work of Konstantin Mironov is supported by European Erasmus Mundus Consortium.

REFERENCES

Barteit, D., Frank, H., and Kupzog, F. (2008). Accurate prediction of interception positions for catching thrown objects in production systems. *IEEE International Conference on Industrial Informatics*, 893–898.

Batz, G., Jaqub, A., Wu, H., Kuehlenz, K., Wollherr, D., and Buss, M. (2010). Dynamic manipulation: Nonprehensile ball catching. *Mediterranean Conference on Control and Automation*, 365–370.

Birbach, O., Frese, U., and Baeuml, B. (2011). Realtime perception for catching a flying ball with a mobile humanoid. *IEEE International Conference on Robotics and Automation*, 5955–5962.

Frank, H., Wellerdick-Wojtasik, N., Hagebeuker, B., Novak, G., and Mahlknecht, S. (2006). Throwing objects: a bio-inspired approach for the transportation of parts. *IEEE International Conference on Robotics and Biomimetics*, 91–96.

Frese, U., Baeuml, B., Haidacher, S., Schreiber, G., Schaefer, I., Haehnle, M., and Hirzinger, G. (2001). Off-the-shelf vision for a robotic ball catcher. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 66, 1623–1629.

Herrejon, R., Kagami, S., and Hashimoto, K. (2009). Position based visual servoing for catching a 3-d flying object using rls trajectory estimation from a monocular image sequence. *IEEE/RSJ International Conference on Robotics and Biomimetics*, 665–670.

Hove, B. and Slotine, J.J. (1991). Experiments in robotic catching. *American Control Conference*, 381–386.

Kim, S. and Billard, A. (2012). Estimating the non-linear dynamics of free-flying objects. *Robotics and Autonomous Systems*, 60, 1008–1122.

Kim, S., Shukla, A., and Billard, A. (2014). Catching objects in flight. *IEEE Transactions on Robotics*, 30, 1049–1065.

MATLAB (2012). Curve fitting toolbox, http://www.mathworks.com/products/curvefitting/, visited on january, 18th, 2015.

Mehta, R., Alam, F., and Subic, A. (2008). Review of tennis ball aerodynamics. *Sports Technology*, 1, 7–16.

Mironov, K. and Pongratz, M. (2013). Applying neural networks for prediction of flying objects trajectory. *Vestnik UGATU*, 17, 33–37.

Mironov, K., Pongratz, M., and Dietrich, D. (2014). Predicting the trajectory of a flying body based on weighted nearest neighbors. *International Work-Conference on Time Series*, 699–710.

Pongratz, M. and Mironov, K. (2015). Accuracy of positioning spherical objects with stereo camera set. *IEEE International Conference on Industrial Technologies*.

Pongratz, M., Mironov, K., and Bauer, F. (2013). A soft catching strategy for transport by throwing and catching. *Vestnik UGATU*, 28–32.

Scaramuzza, D., Pagnoletti, S., and Valigi, P. (2005). Ball detection and predictive ball following based on a stereoscopic vision system. *IEEE International Conference on Robotics and Automation*, 1573–1578.

Tutz, M. (2007). Simulation - schiefer wurf ohne und mit luftwiderstand/dynamischem auftrieb/magnus-effekt, http://www.tutz.ws/js/simulation-schraeger-wurf-$f_l - f_a - f_m$.html, visited on january, 16th, 2015.