Realistic Ball Motion Model for a Tennis Videogame

Lopukhov Andrey
National University of Science and Technology MISiS
Moscow, Ul. Dmitiya Ulyanova 13\1, 52
+79261204135
andrew_lopukhov@mail.ru

Serikov Alexander
National University of Science and Technology MISiS
Moscow, Ul. Profsoyuznaya 83\2
+79258554854
ivanmitrafanych@gmail.com

ABSTRACT

In this paper we speak about the modern simulation games, tennis in particular. We give reasons for using complex mathematical models for providing more realistic physics. The majority of forces influencing the ball flight are described, thus forming a system of motion equations, which possible solution with numeric methods we illustrate. A respective optimization problem is stated and solved with gradient projection. In conclusion areas of further study are suggested.

General Terms

Modeling

Keywords

Videogame, mathematical model, physical model

1. INTRODUCTION

Videogame industry is one of the leading consumers of modern computer technologies. In fact, virtually every game uses complex mathematical apparatus. In contemporary simulator games an average player demands high degree of realism. To achieve this, one has to take into account real world laws of physics when developing the mathematical model. In tennis, for instance, initial ball spinning effects its motion severely so respective laws of motion should be considered. Applying angular velocity to the ball allows the player perform a variety of different hits, e. g. using topspin with maximum force one can send the ball to the court, whereas just a flat hit would have caused an out. To implement such opportunities one have to consider Magnus force and some others, which will be described below.

In our model we try consider all forces that have a great influence on the ball movement so that the best user experience can be provided. In this paper we also introduce methods for calculating initial ball hit conditions, which correspond with the motion model.

2. TENNIS BALL MOTION

Equations of motion can be obtained from Newton's laws of motion. For tennis ball we have three main forces [1][2]: gravity, air resistance and Magnus force.

While ball moves in the air, it has air resistance. This drag force can be as large as gravitation force and cannot be neglected.

In tennis and other ball sports angular velocity of ball is very important. If symmetrical object (such as a tennis ball) translates in air or liquid it experiences only drag force. But when it has angular velocity it receives force perpendicular to its travelling direction. This fact is called Magnus effect.

From articles [1][2], we take following information about forces.

For flat hits (with zero angular velocity) the force of air resistance is proportional to the square of tennis ball speed:

$$F_d = C_d A d \frac{\upsilon^2}{2}, \qquad (1)$$

where $A = \pi r^2$ is the cross-sectional area of the tennis ball, r is ball radius, $d = 1.21 \frac{kg}{m^3}$ is air density, υ is ball speed and $C_d \approx 0.55$ is drag coefficient.

For Magnus force Rod Cross gives following formula:

$$F_m = C\ A d \frac{\upsilon^2}{2}, \qquad (2)$$

where

$$C_l = \frac{1}{2 + \frac{\upsilon}{\omega r}}, \qquad (3)$$

where ω is angular velocity of the ball.

Ball spin change the drag coefficient. A good fit to the experimental data is given by

$$C_d = 0.55 \left[\frac{1}{22.5 + 4.2\left(\frac{\upsilon}{|\omega| r} \right)^{2.5}} \right]^{0.4}, \qquad (4)$$

which indicates that C_d can rise by about 50%.

In this work, we consider two-dimensional ball movement that is projected to three-dimensional space. In such case, we can describe angular velocity as scalar ($\omega > 0$ when Magnus force is directed down and $\omega < 0$ in other case). Using Newton's lows of motion and equations (1)-(4), we have the following system of ordinary differential equations:

$$\begin{cases} \dfrac{dx}{dt} = \upsilon_x \\ \dfrac{dy}{dt} = \upsilon_y \\ \dfrac{d\upsilon_x}{dt} = -k_d \upsilon_x \sqrt{\upsilon_x^2 + \upsilon_y^2} + \\ \qquad + \operatorname{sign}(\omega) k_m \upsilon_y \dfrac{|\omega| r \sqrt{\upsilon_x^2 + \upsilon_y^2}}{2|\omega| r + \sqrt{\upsilon_x^2 + \upsilon_y^2}}, \\ \dfrac{d\upsilon_y}{dt} = -g - k_d \upsilon_y \sqrt{\upsilon_x^2 + \upsilon_y^2} - \\ \qquad - \operatorname{sign}(\omega) k_m \upsilon_x \dfrac{|\omega| r \sqrt{\upsilon_x^2 + \upsilon_y^2}}{2|\omega| r + \sqrt{\upsilon_x^2 + \upsilon_y^2}} \end{cases} \qquad (5)$$

where $k_m = \dfrac{d\pi r^2}{2m}$, $k_m = C_d \dfrac{d\pi r^2}{2m}$, m is mass of ball and C_d is given by (4) if $\omega \neq 0$ or $C_d = 0.55$ otherwise.

Assuming that initial conditions for this system are given, we can treat it as a Cauchy problem. For solving this problem we can use any numeric method.

Modern game engines provide us with opportunity to perform computations between frames with great ease. That is the main reason we considered using one-step methods with fixed time step. Another important criteria of a proper numeric method is its simplicity, conventionality and computational ease. Thus the best fitting methods are Euler's method[4] and the family of second-order Runge-Kutta methods with two stages[5]. These methods obviously calculate only the approximation of a real process. We decided to state that the absolute value of the error must not exceed 0.1 meter per each coordinate during the motion process.

Euler's method fails to meet the precision criteria (for comparisonwe used Runge-Kutta methods with 4 stages and 1/15 or 1/30 time step). From Runge-Kutta family we tested midpoint method[5] and Heun's method[3]. Both succeeded to meet the criteria, varying in precision depending on the initial conditions. Heun's method appeared to fit out purposes the best.

For videogames, 30 frames per second is the sufficient frequency. Our Heun's method implementation in Unity3D computes the solution much faster than 1/30 of a second, thus making it affordable. Further evaluations that are described in par. 3 take time of 2-3 frames, taking in consideration other important computations running simultaneously.

3. INITIAL CONDITION CALCULATION

In tennis player chooses target at court and tries to perform a hit that will accelerate the ball towards the target. An ideal method should evaluate initial conditions (velocity vector $\vec{\upsilon}_0 = (\upsilon_{x0}; \upsilon_{y0})$ and angular velocity value ω) that satisfy the following requirements:

- trajectory of ball goes through target point $(X_{target}; 0)$;
- ball doesn't hit net, which is segment $(X_{net}; 0) - (X_{net}; H_{net})$;
- time of flight is as minimum as possible;
- $\upsilon_{\min} \le \upsilon \le \upsilon_{max}, \alpha_{\min} \le \alpha \le \alpha_{max}, \omega_{\min} \le \omega \le \omega_{max}$, where $\upsilon = |\vec{\upsilon}_0|, \operatorname{tg}(\alpha) = \upsilon_{y0} / \upsilon_{x0}$. The minimum and maximum values represent the player's skill, equipment, game situation, etc.

As far as there are restrictions for angle of the starting velocity, we search initial conditions as three values: absolute value of ball velocity υ, its angle relative to the ground α and angular velocity ω.

We introduce implicit function $\bar{x}(\upsilon, \alpha, \omega)$ that evaluates the x coordinate of ball bounce point. This function can be find as a numeric solution of the system (5).

This system cannot be solved analytically thus an explicit equation for trajectory $y = f(x)$ or $x = g(y)$ cannot be found. Despite this fact we can assume that function $\bar{x}$ has following properties:

- is monotone function;
- is strictly increasing function of υ for any fixed α and ω;
- is concave function of α for $\alpha_{\min} \le \alpha \le \alpha_{max}$ with other arguments fixed;
- is concave function of ω for $\omega_{\min} \le \omega \le \omega_{max}$ with other arguments fixed.

3.1 Nested searches

In this method, we assume that for minimizing fly time maximizing initial velocity is enough. This is not strictly right, though provides us with a decent approximation.

Working with this idea, we can search for a solution by performing nested searches of each argument. The outer one is a binary search of velocity, while the nested ones are ternary searches of angular velocity and angle.

This solution can find initial conditions for trajectory that goes through target point, however this does not guaranty that the ball will not hit the net. This problem can be solved by iterating through a set of such solutions. Since the real range of angles is relatively small, we can find solution for different fixed angles and then choose the best one.

3.2 Optimization problem

The method suggested above can give us a solution that is very performance costly and does not fulfill conditions stated before.

So we consider the following optimization problem:

$$\begin{cases} F(\upsilon, \alpha, \omega) = \bar{\tau}(\upsilon, \alpha, \omega) + p_{target}(\upsilon, \alpha, \omega) + p_{net}(\upsilon, \alpha, \omega) \to \min \\ \upsilon_{min} \le \upsilon \le \upsilon_{max} \\ \alpha_{min} \le \alpha \le \alpha_{max} \\ \omega_{min} \le \omega \le \omega_{max} \end{cases} \qquad (6)$$

where $\bar{\tau}(\upsilon,\alpha,\omega)$ is implicit function evaluating flight time, $p_{target}(\upsilon,\alpha,\omega)$ and $p_{net}(\upsilon,\alpha,\omega)$ are penalties for missing the target and hitting the net.

We defined $p_{target}(\upsilon,\alpha,\omega)$ and $p_{net}(\upsilon,\alpha,\omega)$ with the following formulas:

$$p_{target}(\upsilon,\alpha,\omega)=\begin{cases}0, & \left|\bar{x}(\upsilon,\alpha,\omega)-X_{target}\right|<\varepsilon\\ R_{target}\left(\bar{x}(\upsilon,\alpha,\omega)-X_{target}\right)^2, & otherwise\end{cases}$$

$$p_{net}(\upsilon,\alpha,\omega)=\begin{cases}0, & \bar{y}_{net}(\upsilon,\alpha,\omega)\geq H_{net}\\ 2R_{net}, & \bar{x}(\upsilon,\alpha,\omega)<X_{net}\\ R_{net}(1+H_{net}-\bar{y}_{net}(\upsilon,\alpha,\omega)), & otherwise\end{cases}$$

where $\bar{y}_{net}(\upsilon,\alpha,\omega)$ is implicit function that evaluates the y coordinate of ball trajectory at X_{net} (see Figure 1), R_{target} and R_{net} are penalty coefficients.

We use gradient projection method to solve the optimization problem (6). This solution is very sensitive for the starting point. For choosing it we can use brute force or method described in subsection 3.1.

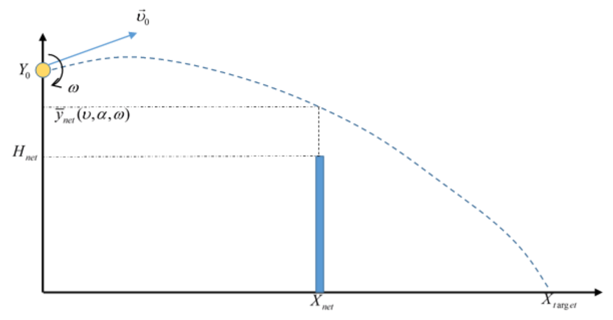

Figure 1

4. RESULTS

We implemented model and methods described above in C# for Unity3D engine. The ball flight of this implementation is more realistic than built-in Unity physics engine solution as it considers Magnus effect and its influence on air resistance.

The initial conditions calculating algorithm is accurate in terms of hitting target.

5. CONCLUSION

We have presented model of realistic tennis ball motion. This model can be used not only for tennis, but also for another ball game with only backspin and topspin rotation type.

Our algorithm for calculating initial conditions of this model is nearly optimal in terms of precision and evaluation performance.

In future work, we plan to optimize initial conditions calculation method to make possible calculation of different hits during one frame.

6. REFERENCES

[1] R. Cross and C. Lindsey, Technical Tennis, Racquet Tech Publishing, Vista, CA, USA (2005).

[2] R. Cross, Aerodynamics in the classroom and at the ball park, Am. J. Phys. 80, 289-297 (2012)

[3] E. Süli, D. Mayers, An Introduction to Numerical Analysis, Cambridge University Press, ISBN 0-521-00794-1 (2003)

[4] Lakoba, Taras I., Simple Euler method and its modifications (2012)

[5] Cellier. F, Kofman. E., Continuous System Simulation, Springer Verlag ISBN 0-387-26102-8 (2006)