An Implementation of Visibility Aware Toric Camera

He, Hao 1600012742

November 2, 2018

1 Introduction

Toric space provides an intuitive and efficient way for controlling camera in virtual environment[1]. Camera trajectories can be easily manipulated and interpolated in Toric space representation. However, the trajectory computation doesn't take any obstacle into account. Obstacles, both static and dynamic, are common in real shooting scenes.

Therefore, this project attempts to explore ways that enable camera to automatically avoid incoming obstacles inside the Toric space, with two objectives:

1. The obstacle should be avoided at all time.
2. The image should change minimally during the avoidance.

A Toric manifold can be defined by the desired shooting target positions on the final image. After that, the camera position and orientation in the Toric manifold are specified by (α, θ, ϕ) defined in [1]. Therefore, our target is to optimize these parameters to solve the obstacle avoidance problem. Methods based on simulated annealing and numerical gradient descent are implemented and evaluated. The basic evaluation we use is the scene in Figure 1.

2 The Visibility Problem

The first problem is to determine whether the actor is occluded by some obstacles. Although algorithms based on actor meshes and obstacle meshes can be done and is extremely accurate, the computation costs are too high for interactive applications. Thus simple bounding geometry are frequently used as substitutes, like bounding spheres, capsules and boxes.

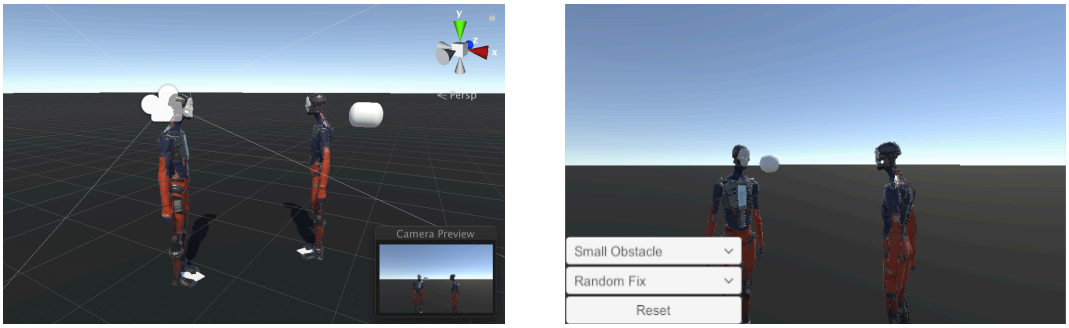

Figure 1: The basic evaluation scene in this project

The Unity physics engine (Based on Nvidia PhysX) provides excellent functionalities for implementing this. The `Collider` components are attached to both actors and obstacles, like `SphereCollider` and `CapsuleCollider`. Then ray casts can be done from camera to determine efficiently whether the actor is blocked by some obstacles.

At first, the visibility problem is solved by using `Physics.CapsuleCastAll` (or equivalent geometry casting functions) from camera to the obstacle, with the casted size equal to the obstacle size. This is extremely efficient (one cast) and works relatively well when the obstacle is small, close to actor and far from camera, but has two fundamental weakness: 1. It breaks when the obstacle is large and far from camera. 2. It cannot measure to what extent the actor is blocked by obstacles.

Therefore, later it is replaced by generating sample points on the actor bounding sphere, and casting ray from camera to the sample points to find the number of sample points blocked by obstacles. If not too much sample points are used, interactive frame rates can be achieved.

3 Part I: Stochastic Local Search Methods

In order to avoid large changes in the target image, α is assumed to be constant and only changes in (θ, ϕ) are necessary to achieve a successful avoidance. Then the problem can be formulated as the following: At each time step t, given current (θ_t, ϕ_t) and desired (θ_0, ϕ_0), choose an appropriate new $(\theta_{t+\Delta t}, \phi_{t+\Delta t})$ value for time step $t+\Delta t$ that satisfies the following constraints:

1. Camera velocity $|\vec{v}| \leq |v_{max}|$. Here $\vec{v} = (\theta_{t+\Delta t} - \theta_t, \phi_{t+\Delta t} - \phi_t)/\Delta t$, which is velocity in the (θ, ϕ) space. This is to make camera motion as smooth as possible.

2. No occlusion, or choose the best effort value that minimizes occlusion.

3. The distance D to desired value should be minimized. Here we define D as the Euclidean distance $\sqrt{(\theta_{t+\Delta t} - \theta_0)^2 + (\phi_{t+\Delta t} - \phi_0)^2}$

For the rest of this section two stochastic local search methods implemented in this project will be introduced: naive random fix and simulated annealing.

3.1 Naive Random Fix

The simplest way to do obstacle avoidance is to choose a random (θ, ϕ) where there is no occlusion. This violates Constraint 1 and often has large D values, but will almost always produce a valid result that satisfies Constraint 2. See video for the result of a naive random fix.

3.2 Simulated Annealing

Simulated Annealing is a common heuristic for optimization problems, and can be used in a continuous space[2]. The algorithm first find a random (θ, ϕ) value as initial value (run the random fix algorithm), and then use simulate annealing to heuristically find an optimal point satisfying both 3 constraints.

The simulated annealing algorithm takes a long time to converge and generates very unstable camera trajectories. Also, for real-time applications it's too computationally expensive. Thus it cannot solve the problem efficiently and naturally. This failure is due to the stochastic nature of this algorithm. See video for implementation results.

4 Part II: Gradient Descent

Gradient descent is a common optimization algorithm for finding the minimum of a function[3]. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point. For this specific problem, It means for each time step t, find the minimal of given function $f(\theta, \phi)$, which is defined by Constraint 2 and Constraint 3, and generate a new value $(\theta_{t+\Delta t}, \phi_{t+\Delta t})$ for any (θ_t, ϕ_t) under Constraint 1. The result is that $(\theta_{t+\Delta t}, \phi_{t+\Delta t})$ either is the computed minimal point, or moves toward the computed minimal point. $f(\theta, \phi)$ is defined as follows:

$$f(\theta, \phi) = \begin{cases} n_{occluded}, & \text{if } n_{occluded} \neq 0 \\ D, & \text{if } n_{occluded} = 0 \end{cases}$$

where $n_{occluded}$ is the number of ray cast samples occluded by obstacles, D is the distance defined in Constraint 3.

Although $f(\theta, \phi)$ is a continuous space function, in implementation (θ, ϕ) is modified in a discrete manner, and $\nabla f(\theta, \phi)$ on the border between $n_{occluded} \neq 0$ and $n_{occluded} = 0$ is defined as always negative from $n_{occluded} \neq 0$ to $n_{occluded} = 0$, in order to heuristically fit this problem. In other words, avoiding obstacle has higher priority than minimizing distance to desired (θ_0, ϕ_0).

Gradient descent generates much more smooth camera motion and after drifting away from the desired (θ_0, ϕ_0) (to avoid obstacles), the camera can automatically return to it. What's more, it works for both near actor, small sized obstacles and near camera, large sized obstacles. See video for the implementation results.

References

[1] Christophe Lino and Marc Christie. Intuitive and efficient camera control with the toric space. *ACM Transactions on Graphics (TOG)*, 34(4):82, 2015.

[2] Marco Locatelli. Simulated annealing algorithms for continuous global optimization. In *Handbook of global optimization*, pages 179–229. Springer, 2002.

[3] Wikipedia. Gradient descent.