Department of Creative Informatics
Graduate School of Information Science and Technology
THE UNIVERSITY OF TOKYO

Master's Thesis

Predicting 3D Bird Flock Motion From 2D Projection

2D 投影からの鳥の群れの 3D モーションを推定

Che-An Lin
林哲安

Supervisor: Professor Takeo Igarashi

January 2019

Abstract

Bird flock animation plays an important role in making realistic outdoor scenes in games and movies. However, since making bird flock animation involves a large quantity of moving entities, it takes effort to create and control an animation. To solve this problem, we introduce a new approach to make this task easier. Our main idea is using a single-view video as input. From the input video, by using animal tracking technique, the two-dimensional position of each bird is retrieved in each frame. We then predict the depth information of each bird by minimizing error function based on trajectory smoothness and flock behavior. With the predicted depth, the three-dimensional bird flock animation is then generated. In this research, we focus on synthesizing three-dimensional flock motion from two-dimensional projection. Generating visually plausible result is our goal. We show our system is capable of generating bird flock motion from various input videos.

概要

鳥の群れの動きは、ゲームや映画の外景表現において重要な要素である。しかし、群れの全ての動きを制作するのは、各鳥の動きの自然さと群れらしさを同時に考慮する必要があるため非常に手間がかかる。また、生成結果をユーザがコントロールすることも難しい。本研究では、鳥の群れの動画を入力とすることで、3 次元の鳥の群れのアニメーションを生成する手法を提案する。具体的な手順としては、トラッキング技術を基に得られた二次元的な動作情報を基に、「各個体の軌跡のスムージング項」と「既存の群れ動作アルゴリズム」に基に、深度情報を推定する。本論文では、複数の動画データを用いた 3D アニメーションの生成結果から有用性を評価する。

Contents

Chapter 1

Introduction

The art of flock simulation has received increasing interests in the multi-media and entertainment industry. Bird flock scene is widely used in games and movies to make the scenes richer or more natural. However, since the number of objects involved is large, creating bird flock takes considerable time and efforts.

The work of Raynolds [1, 2] is our starting point about the study of a distributed behavior model. Reynolds proposed boid model, which described the behavior of large groups of birds, herds, and fish with perceptual skills existing in the real world. However, as stated by Reynold, the original boid model can only model flock wandering behavior. Controlling flock remains a necessary task for synthesizing bird flock.

In this paper, we propose an alternative method for modeling bird flock. Our method uses a single RGB bird flock video as input to synthesize flock motion. The goal of our method is not to reproduce the exactly the same flock motion as it in the video. Instead, our method uses video as a reference to synthesize flocks that looks similar and natural to those in the video. We think perfect reconstruction of bird flock is nearly impossible without depth information, as we do not have precise three-dimension motion data as ground truth for evaluating our result.

The biggest challenge of this approach is that RGB video does not contain depth information. This makes flock motion synthesizing a difficult task. Human eyes can easily track moving objects, but tracking moving objects, especially bird flocks, is a challenging task in computer vision. The reason why we use RGB video as input is, obtaining depth information of bird flock with depth camera is difficult in outdoor scene. In work from Ju et al.[3], or recording locomotion of a single bird as training data, they recorded the motion of dove using marker-based optical motion capture and high-speed video cameras for their work. They used 28 cameras to capture a single dove motion in region of 10m × 10m × 7m. Apparently, more space and devices are needed if we do the capture on a bird flock. In fact, we did capture bird flock motion by taking bird video using a 360-degree camera in outdoor scenes, but we failed to get useful video due to bad condition of outdoor environment. And it is also difficult to set up an indoor scene as they did for capturing bird flocks, since the space needed is too large. Thus, we consider predicting depth information of bird flock is helpful for making bird flock motion using real video as reference.

Two-dimensional projection data is the basis for predicting three-dimensional position. The main contribution of this research is to provide an approach for predicting three-dimensional bird flock motion from two-dimensional track data retrieved from video. That is, to predict depth from each bird to the camera in each frame. With predefined camera parameters and projection on screen p_n^f, if the distance between the bird and the camera can be predicted, three-dimensional position of the bird b_n^f in frame f of bird n can be obtained by:

$$b_n^f = Proj(p_n^f, d_n^f) \tag{1.1}$$

To predict d, track data must be obtained from input video first. We use an animal tracking technique for this task. This part is done by using an existing interactive feature tracking technique proposed by Buchanan et al.[4]. Interactive feature tracking is the process of extracting long and accurate tracks of three-dimensional features observed in two-dimensional video. Although the system is designed to track feature points, such as human face or eyes, it is also suitable for tracking bird flock, which contains multiple small trace targets that can also be treated as feature points.

After the track data is obtained, predicting depth of each bird is the next step. Based on our observations and boid flock models, we define two properties that a should have to retain the quality of flock motion. First, considering one bird flying trajectory, we do not expect a bird in frame f to appear too far from its position in frame $f-1$ from its previous speed. That is, changes in position tend to be gradual. This is the first property: trajectory smoothness, as shown in Figure 3.3. After trajectory smoothness is defined, now we consider a group of birds. Reynolds' research about flock behavior [1, 2] is our starting point about flock behavior.Reynolds proposed three steering behavior rules: separation, cohesion and alignment. In case of flock simulation, three rules are considered as three forces applied to each bird in each simulation steps. We also apply these rules in our system as the second property, flock behavior similarity. Illustration of the three rules are shown in Figure 3.4. Based on the two properties, we design an error function to represent the quality of a flock motion. Thus, the task of predicting depth can be restated as on optimization problem: choose depth set of N bird in F frames to minimize the error function. However, optimizing over $N \times F$ depths takes time, making the system far from interactive system. To overcome this issue, we propose a frame-by-frame optimization framework. In our implementation, we just optimize the position of N birds in one frame based on the result on previous frames, then optimize the positions in next frame until all frame are optimized. By this frame-by-frame framework, the computation time is greatly decreased. In our experiment, a 10-second 30-fps video with 200 frames and 5 birds can be optimized in less than one second. We will further discuss the details of the optimization method in Chapter 3.

The system presented in this paper can be separated into four stages. The first stage is trace retrieving stage. In this stage, trace data is retrieved by the system with user indications. Trace data contains projected two-dimensional positions in each frame for each bird. The system only allows user to retrieve trace for only one bird at a time, so trace data must be generated and saved for each bird before going to the next stage.The second stage is optimization stage. In this stage, optimization is performed based on defined energy function to predict flock motion in three-dimensional space. Since the processing can be done in real time, user can adjust parameters to the systems and see the result directly for better results.The last stage is refinement stage. Since the system only predict the position in last stage, the orientation of each bird is calculated based on its position in each frame to complete the flock motion as output.

For evaluating our method, we test our system by various input video. We further implemented a flock simulation system to synthesize bird videos as input, since it is difficult to find good video with real flock motion that fit the requirements of the system. Our result shows that the system is capable of generating various bird flock motion.

The structure of the paper is as follows: In chapter 2, we discuss related work. In chapter 3, we present the overall design of our system discussing the requirements needed for predicting flock motion, and give implementation detail of our system. In chapter 4, we present the results generated by our system. In chapter 5, we discuss about the

limitations of our system and future work. Finally, we summarize this research in chapter 6.

Chapter 2

Related Work

2.1 Animal simulation

Computer graphic researchers have been fascinated by creating life-like computer-generated creatures. Several models are proposed for the nature motion of animals. Most of this research focus on humans or terrestrial animals. Social force model proposed by Helbing and Molnar [5] has been widely used in pedestrian behavior simulation. Miller [6] proposed a physically-based simulation method for snakes and worms. Satoi et al. proposed a unified motion planner [7] that models fish motion with different swimming styles. Bayazit et al. [8, 9] studied four kinds of group behaviors, including homing, exploring, passing through narrow areas, and shepherding. In their method, they used global information in the form of a roadmap to model these flocking behaviors. Most of these simulation models use rule-based algorithm based on local or global behaviors.

2.2 Bird flight simulation

Simulation of bird flight has also gained attention for creating life-like creatures. The work of Wu and Popović [10] describe a physics-based method for synthesis of bird flight animations by given user-specified three-dimensional path. Ju et al. [3] proposed a data-driven approach for controlling flapping flight. These works focus on simulating bird behavior. However, they simulate single bird locomotion based on its structure and aerodynamics, which does not consider the interaction between birds in a flock. In our research, rather than the realism of bird locomotion, we focus on synthesizing life-like flock motion based on trajectory smoothness and flock behavior. For simulating on group behavior, Anderson et al [11], used an iteractive sampling method to generate user-constrained group animation by specifying path in three-dimensional space. Xu et al [12] proposed a shape-constrained flock system for interactively controlling flock navigation. Their systems aim to produce plausible animation that satisfy user-specified constraint while retaining the realistic properties of the underlying behavior model. Nevertheless, synthesizing realistic behavior of bird is our common goal.

2.3 Interactive control of simulation

Crowd simulation is the most commonly used interactive simulation system in games and movies to make crowded scenes. Crowdbrush, proposed by Ulicny et al. [13] is a tool for interactive authoring of real-time crowd scenes. As a pioneer of controlling simulated crowds, it provides interface for controlling crowds with brush tools. But the control operations are still limited to small group of individuals by specifying the property or rule, which still needs lots of work for controlling large crowds. Golaem[14] and Massive[15]

are mature commercial crowd simulation software widely used to build crowded scenes in movies and video games. Both software tools allow user to manage and control crowds based on path planning and steering behaviors. However, modeling aerial motion is more challenging than humans or terrestrial animals, since birds do not only stay on the ground as they do. Golaem also includes tool for controlling flocks, but it has much less function than tools for crowd simulation, only providing target-based control and collision avoidance.

2.4 2D-based modeling

In computer graphics, Image-based modeling is a method which relies on a set of two-dimension images of a scene to generate a three-dimensional model. Iwasaki et al. [16] proposed a modeling method of clouds from a single photograph. Okabe et al. [17] models volumetric fluid such as smoke or fire, from sparse multi-view images. For most works about 3D reconstruction like [18], systems are based on shapes from silhouette in multiple views. Generating three-dimensional curves through sketching is also a rich research domain. Pentland and Kuo [19] presented an approach for reconstructing three-dimension object from two-dimensional sketch. In work of Ijiri et al. [20], they presented a system for modeling flowers which synthesizes three-dimensional botanical structure by constant curvature with two-dimensional sketch.

2.5 Motion sketching

Despite the success of computer-generated animations, traditional hand-drawn approaches for creating animation stays popular with expressive and stylized looks. However, creating hand-drawn animations is a tedious process that requires years of training by an artist. Thus, motion sketching has become an popular approach for creating realistic animation. Jovan P et al. proposed a sketching interface for specifying rigid body animation [**?**]. The method estimates simulation parameters from the hand-drawn sketch. In the research of Marek D. et al., they proposed a approach to create stylized 2D rigid body animations by hand-drawn sketch[**?**]. They also proposed ToonSynth [**?**], an example-based synthesis of cartoon animations, which produces hand-colored character motion by only drawing skeletal motion. Those approaches are designed to reduce the authoring effort of artists. Although our system receives video as input, it is also capable of receiving hand-drawn sketch as track data of bird flock. We will discuss about this possibility of our system in Chapter 5.

Chapter 3

Method

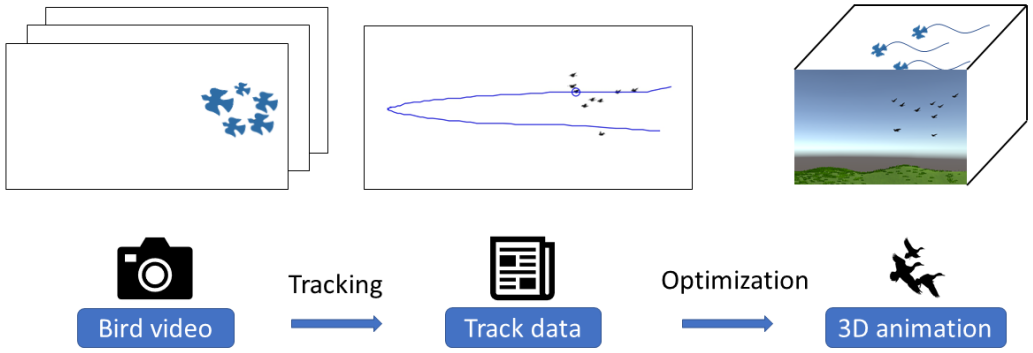


Fig. 3.1: Method overview

3.1 Method overview

Figure 3.1 shows an overview of our method. As mentioned in Chapter 1, our method synthesizes three-dimensional bird flock motion from input video by predicting depths for each bird in each frame. First, trace data of each bird in the video is retrieved from input video. With the interactive feature tracking technique, track data, which contains two-dimensional projection position trajectory of each bird, can be retrieved with the help of user's indications. Next, depth for each bird in every frame is predicted in optimization step. The goal of predicting depth is to maintain the quality of flight trajectory, which is separated in two terms: trajectory smoothness and flock behavior similarity. These two terms will be discussed in later sections. Here, we consider an image sequence of length F frames as input, with N birds in each frame. Track data contains two-dimensional projection of each bird in the input video. The set of track data is denoted $P = \{p_n^f\}_{n=1...N, f=1...F}$, where $p_n^f = (x_n^f, y_n^f)$, representing two-dimensional projection on screen, as shown in Figure 3.2. We want to predict depth d_n^f of bird n in

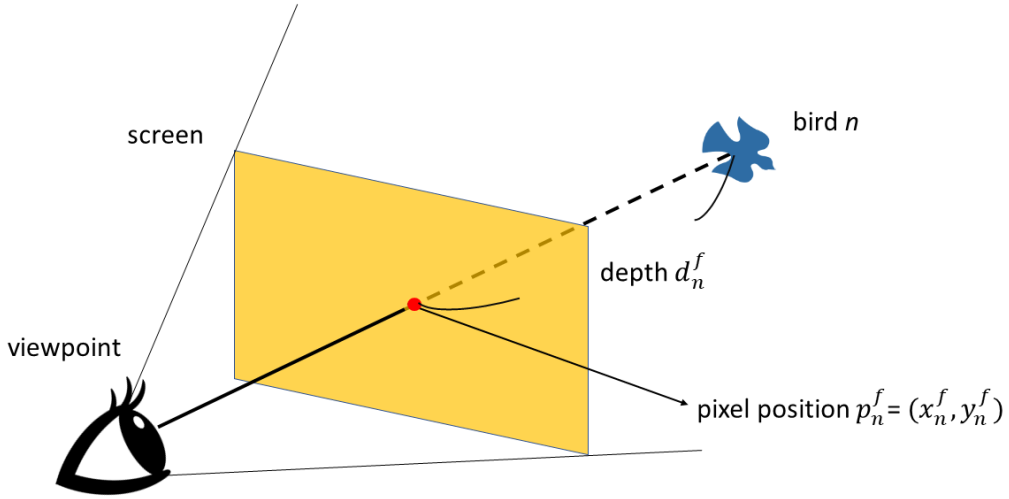


Fig. 3.2: View scenario in frame f.

frame f, while maintain the two properties above. The quality of a flight with a candidate depth set $X = \{d_n^f\}_{n=1...N, f=1...F}$ can be defined as:

$$E(X) = \sum_{n=1}^{N} \sum_{f=1}^{F} e(n, f) \tag{3.1}$$

$$e(n, f) = \lambda_t t(p_n^f, p_n^{f-1}, d_n^f) + \lambda_f f(p_n^f, p_n^{f-1}, d_n^f) \tag{3.2}$$

$e(n, f)$ denotes error function for bird n in frame f, which includes a trajectory smoothness term $t(.)$, and a flock behavior term $f(.)$. λ_s and λ_t are tuning parameters corresponding to trajectory smoothness and flock behavior. Thus, the predicting problem can be restated as an optimization problem: choose X to minimize $E(X)$. After the optimal X is found, with the predefined camera parameter and optimal depth d, we can get world position of the bird from its screen position.

In most simulation system, velocity of an object is calculated upon mass and force in simulation steps. However, since our optimization method is over frames, we use position-based dynamics system [21], in which velocity in frame f is denoted as

$$v(\vec{f}) = b(f) - b(f-1) \tag{3.3}$$

where $v(f)$ denotes velocity, and $b(f)$ denotes bird position in world space. This makes our optimization system simpler and faster: we can calculate velocity of each bird in current frame and do not need to keep track of it. Although position-based dynamics system is not physically accurate, since our goal is to make visually plausible flock motion, we use it in our system. For s and t in equation 3.2, we use the simple Euclidean distance function on the two positions. We have also experimented with an acceleration term, of the form $s(p_n^f, p_n^{f-1}, p_n^{f-2})$, but it increases the computation cost, and do not produce better result. Details of each term will be discussed in the following sections.

3.2 Bird tracking

To obtain trajectory of each bird in the input video, tracking technique is used in our method. We use an existing tracking system ZooTracer[22], developed by Microsoft research, to do the task. ZooTracer is based on interactive feature tracking technique, proposed by Buchanan and Fitzgibbon [4]. With the tracking system, after preprocessing is done, the user can make tracking data interactively from input video in real time. The system predicts the trajectory of single bird in the video based on user's initial indication. If the tracking is wrong in some frames, user can provide keyframe features by few clicks and improve the track. The main challenge of using interactive feature tracking on tracking birds is distinguishing between tracked birds. The system records fixed-size image patches for tracking feature points, such as eye on a face. However, in case of tracking bird flock, all birds look similar in small patches. It leads to many errors found in the result, making wrong tracking data. Therefore, more manual manipulations need to be done to get an optimal track. On average, user can complete the track data of one bird in a 200-frame video in less than 1 minute.

3.3 Trajectory smoothness

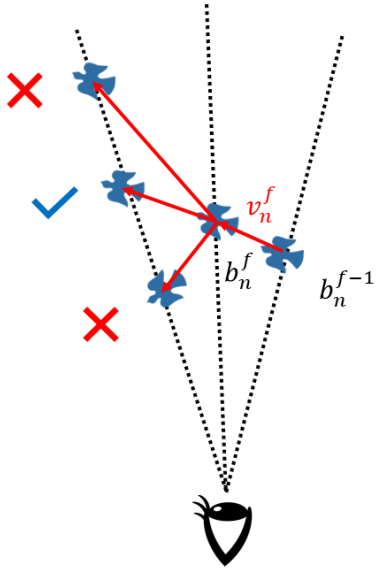


Fig. 3.3: Illustration of trajectory smoothness. The trajectory in the middle is considered a better trajectory, since the speed difference is smaller.

Trajectory smoothness is the first term in our error function in equation 3.1, denoted as $t(.)$. In this term, only the trajectory of one bird is considered. We assume that a bird tends to minimize its velocity change during its flight. The importance of retaining trajectory smoothness is especially obvious when bird is on turning point of the track. In our experiment, we found that if trajectory smooth is not considered, birds will show a sudden turn motion, which is unnatural. Figure 3.3 shows the illustration of trajectory smoothness. Trajectory smoothness term is then denoted as:

$$t(p_n^f, p_n^{f-1}, d_n^f) = |||v_n^f|| - ||v_n^{f-1}||| = |||b_n^f - b_n^{f-1}|| - ||b_n^{f-1} - b_n^{f-2}||| \quad (3.4)$$

as first term in error function to be minimized in optimization stage to retain trajectory smoothness.

3.4 Flock behavior similarity

We use Reynolds' boid model as basic flock behaviors. In boid model, each bird in flock, which is called a boid, observes three steering rules: separation, cohesion, and alignment. The definitions of these steering rules rely on the neighbors of the boid. The neighbors of one boid include its surrounding neighbors who are sufficiently close, and each rule has its perceptual neighborhood. Figure 3.4 illustrates the definition of three boid rules. As the basic rules of flock behaviors, we believe that more the result matches the boid rules, the better result is. This property is considered as the second term: flock behavior similarity, meaning how close the flock motion shows to flock behavior rules. To calculate flock behavior similarity, target position of boid n in frame f must be calculated first. The target position is then compared with the predicted boid position. Target position is the sum of boid rules. Here we further describe the calculation of the three steering rules.

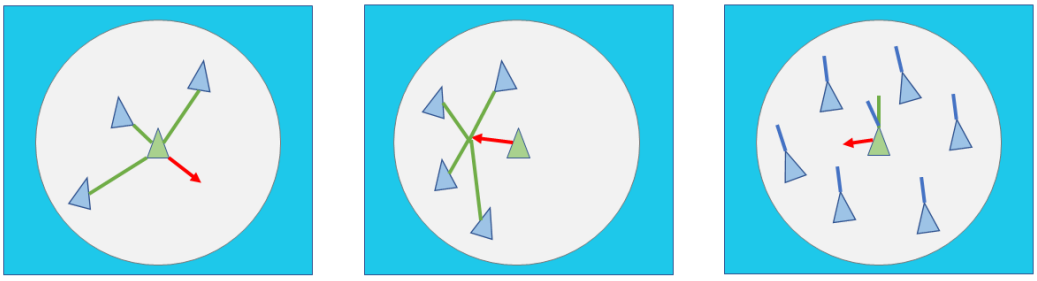


Fig. 3.4: Illustration of three boid rules. (a) separation behavior. (b) cohesion behavior. (c) alignment behavior.

Separation steering behavior, as shown in Figure 3.4 (a), gives a boid the ability to maintain a certain separation distance from others nearby and to prevent boids from crowding together. We think that separation rule is the most important rule for creating nature flock motion, since collision with other birds is obviously unnatural. To compute separation force, first a search is made to find other boids within the specified neighborhoods. For each nearby boid, the force is computed by subtracting the positions of the boid and the nearby boids, normalizing, and then applying a $1/r$ weighting, making force of nearer nearby boid stronger. All forces for each nearby boid are summed together to produce the overall separation force.

Cohesion steering behavior, as shown in Figure 3.4 (b), gives a boid the ability to approach and form a group with other nearby characters. That is, boids steer towards the average position of the nearby boids. This behavior helps flock to form a group and not be too separated. The neighborhood of cohesion is set much bigger than neighborhood of separation does to keep the flock form in group while preventing collision.

Alignment steering behavior, as shown in Figure 3.4 (c), gives a boid the ability to align itself, heading in the same direction with other nearby boids. However, since in the optimization stage, we only consider optimizing position of each boid, we do not use alignment rule here. Instead, we refine the alignment after optimization is done.

```
// Calculate separation force s⃗ of bird n
s⃗ ← 0⃗ ;
neighbor ← 0 ;
for i ← 1 to N do
    if i=n then continue;
    distance ← ||b_n − b_i|| ;
    if distance ≤ d_sep then
        neighbor ← neighbor + 1 ;
        s⃗ ← s⃗ + (b_n − b_i)/distance ;
    end
end
s⃗ ← s⃗/neighbor ;
```

Algorithm 1: Calculation of separation vector $\vec{s}$

```
// Calculate cohesion force c⃗ of bird n
c⃗ ← 0⃗ ;
for i ← 1 to N do
    if i=n then continue;
    c⃗ ← c⃗ + (b_i − b_n) ;
end
c⃗ ← s⃗/(n − 1) ;
```

Algorithm 2: Calculation of cohesion vector $\vec{c}$

Combining rules above, now we can calculate flock behavior similarity term $f(.)$ in equation 3.2 by calculating Euclidean distance between the target position and predicted position. Algorithms of calculating separation and cohesion are shown in Algorithm 1 and 2. In calculating separation vector $\vec{s}$, a neighborhood distance d_{sep} is set to decide the minimum distance to keep in flock motion between each birds. Tuning this parameter, which can be done by user in optimization stage, is considered as an interactive element in our system. In boid algorithm, a neighborhood of cohesion is also set. However, since we assume all birds in input video belong to one flock that are aware of each other, we just consider every birds in the flock for calculating cohesion force. This can also reduce number of parameters while keeping other parameters more focused.

In most flock simulation approaches, collision is detected in advance by adjusting its direction and speed when other agents are about to collide. In our flock model, since we have separation rule that separate, birds tend to move away to each other when they are too close. Therefore, we do not handle collision avoidance.

3.5 Frame-by-frame optimization

Data: Trace set $X = \{d_n^f\}$
`// Algorithm start`
Scatter bird positions in frame 1 ;
`// Frame-by-frame optimization`
for $i \leftarrow 2$ **to** F **do**
  $sum \leftarrow 0$;
  for $j \leftarrow 1$ **to** N **do**
    $\vec{v} \leftarrow b_j^i - b_j^{i-1}$;
    if $i = 2$ **then**
      `// Initial speed`
      $sum \leftarrow sum + ||||\vec{v}|| - v_{target}||$;
    end
    else
      `// Trajectory smoothness`
      $\vec{v_{last}} \leftarrow b_j^{i-1} - b_j^{i-2}$;
      $sum \leftarrow sum + \lambda_t||||\vec{v}|| - ||\vec{v_{last}}||||$;
      `// Flock behavior similarity`
      Calculate separation force $\vec{s}$;
      Calculate cohesion force $\vec{c}$;
      $target \leftarrow b_j^{i-1} + (w_{sep}\vec{s} + (1 - w_{sep})\vec{c})$;
      $sum \leftarrow sum + \lambda_f||b_j^i - target||$;
    end
  end
  Minimize sum ;
end

Algorithm 3: Optimization algorithm.

The task in our optimization stage is to minimize error function equation 3.1 with target depth set X. Totally, since we have track data of N birds in F frames, there are $N \times F$ target depths. If the optimization is done globally, the computation time will be costly. Since our goal is to make system that user can adjust the result interactively in real time, we consider doing the optimization frame-by-frame. Frame-by-frame optimizations means we do not optimize Equation 3.1 directly, instead, we predict N depths in frame f from the first frame to next frame, based on the results from last frames. That is, to optimize Equation 3.2 for $n = 1...N$ This approach greatly decreases computation time, making it possible for user to do the optimization and see the result interactively while adjusting parameters.

To prevent optimization result make bird flock fly too far or too near to the camera, as inequality constraints in optimization algorithm, a near and a far constraint of depth are set as:

$$d_{near} < d < d_{far} \tag{3.5}$$

However, since constraints d_{near} and d_{far} do not affect the result much. It only avoid flock fly too near or too far to the camera, which seldom happens. Therefore, we do not

consider these two parameters as interactive elements for user. Since target error function can be calculated in each frame, with the inequality constraint 3.5, after optimization is done in every frame, optimized X can be found. The definition of our optimization is denoted as:

$$\{d_n^f\} = \arg\min \sum_{i=1}^{N} e(i, f) \tag{3.6}$$

subject to

$$d_{min} < d < d_{max} \tag{3.7}$$

and do the frame-by-frame optimization:

$$d_n^0 -> d_n^1 -> d_n^2 -> ... \tag{3.8}$$

Algorithm 3 shows our optimization algorithm. As mentioned above, our frame-by-frame optimization optimizes each frame based on the result from last frames. However, we do not have any previous frame for the first frame. Therefore, we have to decide the initial positions of N birds in the first frame. This part can also be discussed in many aspects. However, we decide to make this part simple: just scatter it in the range within d_{near} and d_{far} randomly, with at least a distance between them.

After the positions of birds in the first frame are decided, next is the second frame. Since the distance of bird positions between frame 1 and frame 2 greatly affects optimization in later frames, the position in frame 2 is important. Here, we use a user-defined speed factor v_{target} as a reference to decide the positions in frame 2. We simply spawn birds in frame 2 in position that approximate the defined distance v_{target} to the position in frame 1.

The reason why we do not dig in the part of deciding initial positions is that it is less important than how trajectory shows in flock motion, since our goal is not to reproduce the original flock motion. That is, how initial positions are decided does not affect the evaluation of the result. Thus, we focus on the transition of bird flock rather than initial position.

After the positions in frame 1 and 2 are decided, now we can apply the rule of trajectory smoothness and flock behavior similarity based on positions in previous frames. The first term, trajectory smoothness, is calculated as velocity difference between this frame and last frame. The second term, flock behavior similarity, is calculated as sum of separation force $\vec{s}$ and cohesion force $\vec{c}$, which are shown in 1 and 2. We balance these two force using parameter w_{sep}, which is considered as an interactive element in our system. As mentioned in section 3.4, separation force prevents birds from colliding, while cohesion force keeps flock stay together. Final force is decided as $(w_{sep}\vec{s} + (1 - w_{sep})\vec{c})$. By tuning w_{sep} from 0 to 1, the user can decide the flock in result to scatter or gather. After two terms are decided, we also balance the two terms are by two parameters: λ_t and λ_f, balancing the trajectory smoothness and flock simulation similarity. After the sum of two terms are calculated, it is optimized by changing d_n^f. After the optimal depth d_n^f is solved by the optimization algorithm, the system moves to next frame until all frames are done.

We implement the optimization part using NLopt library [23] with optimization algorithm constrained optimization by linear approximation (COBYLA)[24].

3.6 Modeling flock motion

After we get the optimized depth data, next step is to visualize the optimized flock motion. We implement the visualization with Unity[25]. And optimization system is loaded as

plugin to make it can be used in an interactive way.

In optimization stage, the position of each bird in every frame is predicted. However, since the system only finds optimal position of each bird, orientation needs to be assigned to complete flock motion. Here we just simply decide orientation of each bird frame-by-frame based on its current position and position in last frame. First, the orientation vector of bird n in frame 2 is simply decided by position difference $\vec{o}_n^f = d_n^2 - d_n^1$. After that, we just interpolate position difference vector and previous orientation $\vec{o}$ with parameter 0.1 in every frame, meaning that the orientation is slightly changed each frame while keeping its previous orientation. We use interpolation here because sometimes the result seem weird because of errors in original track data, causing birds changing orientation frequently.

3.7 Interactive user interface

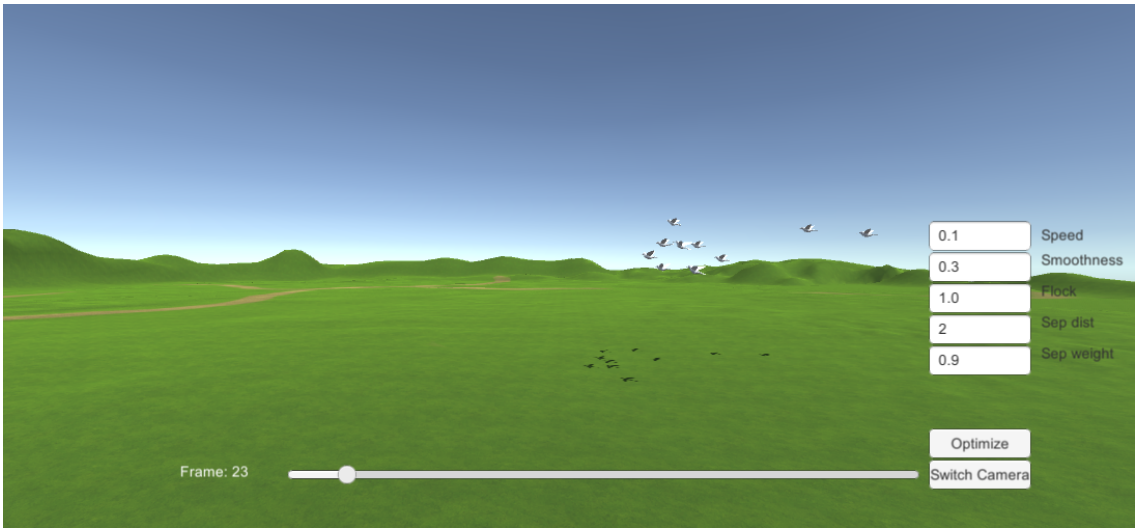


Fig. 3.5: User interface of our system in optimization stage.

By now, we have introduced 5 parameters for optimization stage: v_{target} for target speed of each bird, λ_t for weight of trajectory smoothness, λ_f for weight of flock behavior similarity, d_{sep} for neighborhood distance for separation rule, andw_{sep} for weight of separation rule. In our system, these parameters are treated as interactive elements of our system. Since the optimization can be done in seconds, user can see the result and adjust parameter can do the optimization again to get better results. Figure 3.5 illustrates our user interface. With our interactive user interface, user can move camera to check the result in each frame. 5 parameters can be edited by the text fields on the right part of the screen, and the slider on the bottom part of the screen can adjust the frame shown. User can also switch between top-view camera and side-view camera by pressing switch button.

Chapter 4

Result

Fig. 4.1: A view of our flock simulation system. The ball on the left is the target for navigating flocks.

4.1 Flock simulation for generating input video

For testing our method, a video with bird flock is needed. We planned to use real bird video at first. However, after searching for real bird videos, we found that videos we can find are not good enough as input video, since our approach has several assumptions: all birds stay in the view of camera, and camera is not moving. We also captured various bird videos, but it is still difficult to obtain enough videos that satisfy the assumptions.

Due to difficulties mentioned above, other than using real bird video, we further implemented a bird flock simulation system and capture the flock motion as input video to test our method. The simulation system is based on boid model, which assumes a flock is simply the result of the interaction between the behaviors of individual birds. With these behaviors, the system can produce fine flock motion with numbers of birds. However, boid model can only model flock wandering behavior. That is, after assigning initial parameters, all birds become uncontrollable during the simulation, and the simulation always produces same results. To keep the diversity of the generated simulation result, we further include the homing behavior introduced in [12, 8]. With this behavior, we can generate different flock motions efficiently with various different results. After the simulation result is generated, we capture videos from a view. The captured video is then used as an input

video to the system. Figure 4.1 shows a view of our flock simulation system.

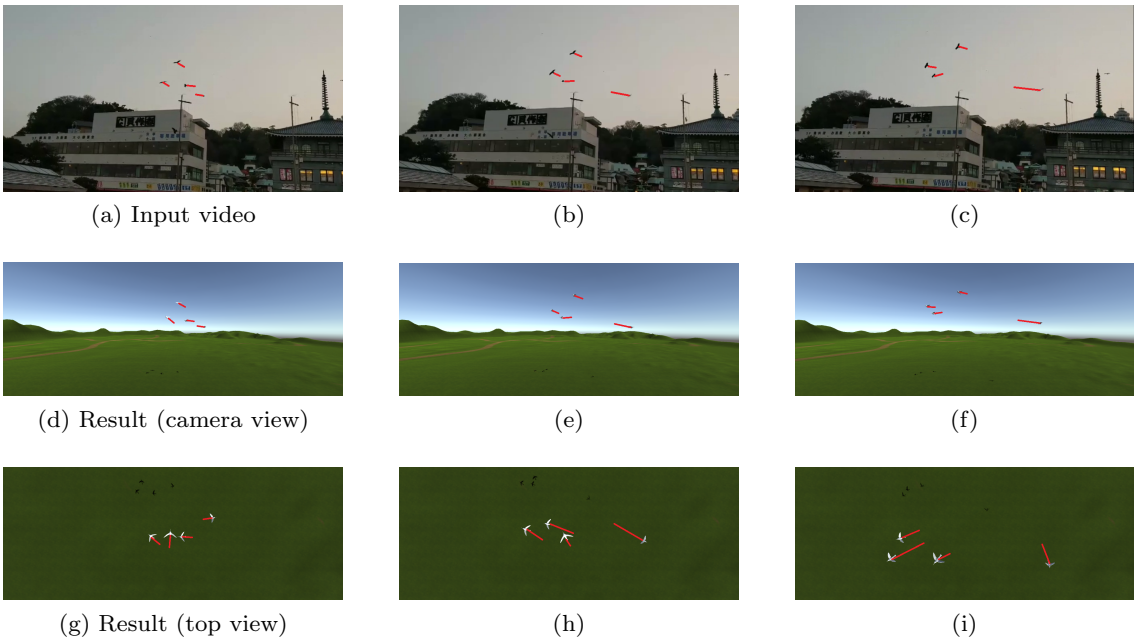
4.2 Generated flock motion

We have tested our with various bird videos. Here we show three representative results. The first one is a real bird video taken with RGB camera by ourselves, while the other two are generated input videos from our flock simulation system. Information and parameters of three input videos are shown in Table 4.1.

	resolution	bird num	frame num	v_{target}	λ_t	λ_f	w_{sep}	d_{sep}
Result 1	854x480	4	120	0.1	0.3	1.0	2	0.9
Result 2	1280x720	10	340	0.1	0.3	1.0	2	0.9
Result 3	1280x720	5	230	0.5	0.3	0.1	2	0.8

Table 4.1: Information and optimization parameters of 3 input videos.

Figure 4.2 shows our first result generated with real bird video as input. The input video is relatively short and has 4 birds. Camera also slightly moves during recording. Although the quality of the input video is not good, we test our system with this real video before using generated videos as input to claim that our system can be used on real bird videos. Results of 3 frames in time sequence are shown in Figure 4.2. The first row is the input video, and the second, third rows are generated flock motions from our system, in camera view and top view.



(a) Input video (b) (c)

(d) Result (camera view) (e) (f)

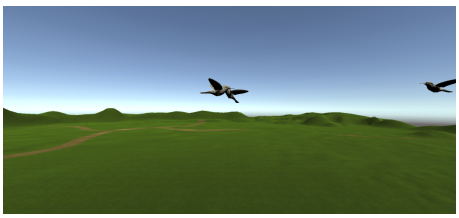
(g) Result (top view) (h) (i)

Fig. 4.2: Comparison of Result 1 in frame 40, 60, 80. Red lines are bird trajectories in previous frames.

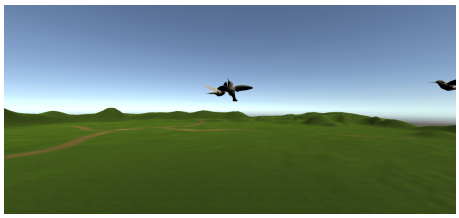
By comparing between the input video (a) (b) (c) and generated result in camera view (d)(e)(f), it can be observed that the trajectories of all birds are exactly same, since two-dimensional projection is used as a constraint in the optimization stage. This shows that our optimization algorithm retain the track information from the original video. To

confirm whether the depth is properly predicted, we check the result motion from another view from top, shown in (g)(h)(i). Since this result contains only four birds, it is more difficult to observe flock behavior. However, by in (g) and (h), we can still observe that two birds avoid each other when they are two close, showing that flock behavior has its functionality.

An experiment has also be done with this video input to further examine the functionality of the flock behavior similarity term in our error function. We generated two different result, with one applying flock behavior similarity term, and one disabling the term by setting λ_f to 0. Figure 4.3 shows the result. Figure 4.3 shows the particular frame when colliding happens. In (a) and (b), representing results in camera view, we can observe that two birds overlap with each other in camera view. In another view (c) and (d), we can see that, with the flock similarity term being active, the two birds does not collide in (c), while birds collide in (d). This shows that the functionality of flock behavior similarity term works well, since it is defined to keep birds in flock from colliding with each others while keeping the flock together.



(a) Result with $\lambda_f = 1.0$ (camera view)



(b) Result with $\lambda_f = 0$ (camera view)

(c) Result with $\lambda_f = 1.0$ (top view)

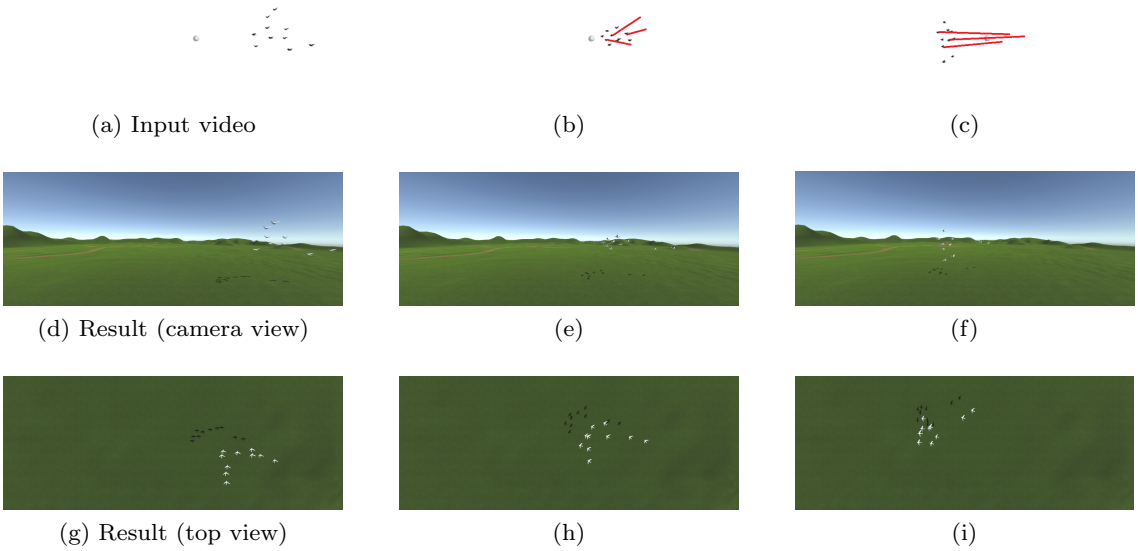
(d) Result with $\lambda_f = 0$ (top view)

Fig. 4.3: Comparison between results with flock behavior similarity term on and off.

However, since result 1 is too short and lack of numbers to observe flock behavior, we use generated video instead for testing with various conditions. Based on our observation on bird flocks, we think that bird flock can be categorized into two types: united flock and separate flock, depending on the relativity between birds in a flock. Thus, we decide to make two kinds of bird flocks to test if the system is capable of having these two kinds of flock motion.

Figure 4.4 and Figure 4.5 show an result with a video generated by our flock simulation system. The video is relatively long and has 10 birds flying around an object in a united form. Birds fly in similar direction during the whole video. This video is used to evaluate the ability of video with standard united flocks. In the generated flock motion, flock behavior can be observed: birds stay united and collision avoidance is well done. Turning of the flock happens in frame 80 and 100, and the generated motion also turns smoothly, based on trajectory smoothness term in optimization algorithm. This result shows that turning scenario is also handled well even the depth is unknown in the original video.

Figure 4.6 shows another result with a video generated by simulation. We test our system with this video to observe if birds in the original video fly towards camera. In

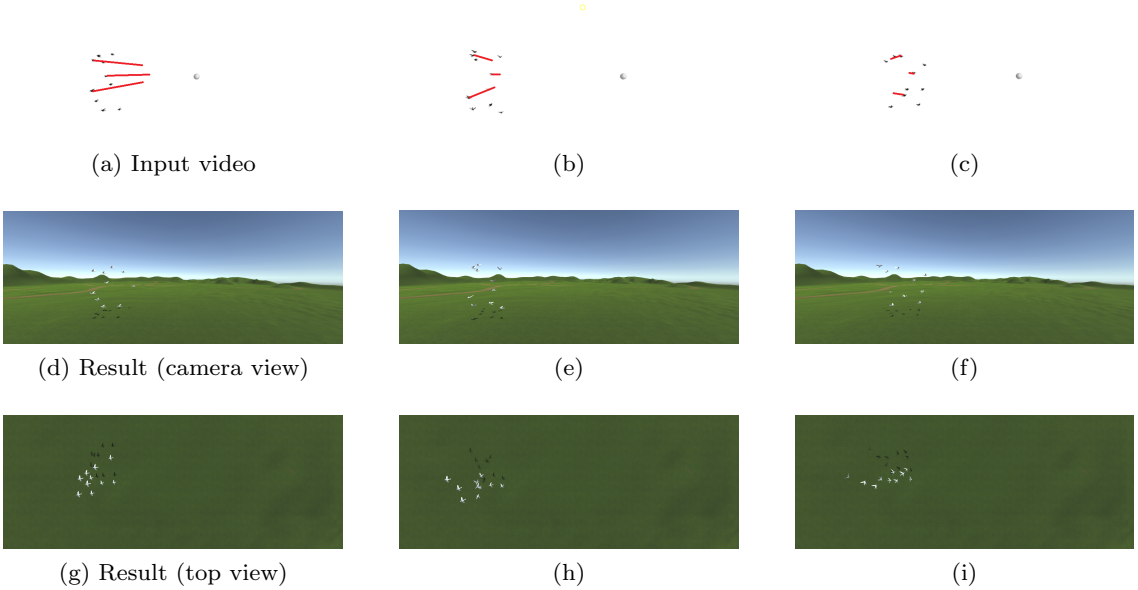


(a) Input video (b) (c)

(d) Result (camera view) (e) (f)

(g) Result (top view) (h) (i)

Fig. 4.4: Comparison of Result 2 in frame 0, 20, 40.



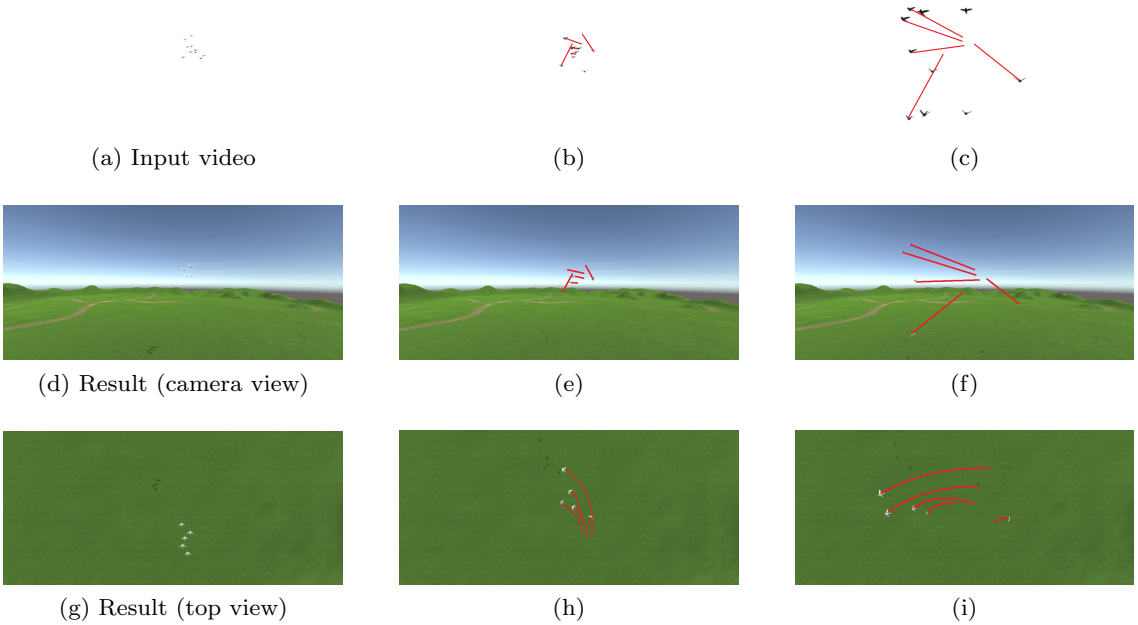
(a) Input video (b) (c)

(d) Result (camera view) (e) (f)

(g) Result (top view) (h) (i)

Fig. 4.5: Comparison of Result 2 in frame 60, 80, 100.

such scenario, the change in projected position is relative small, meaning that the change in depth is considered to be large. As a result, the generated motion is different with expected: birds leave the camera instead of approaching camera like those in original video. However, this is also and acceptable result, since our goal is not reconstructing the original flock motion. The result still shows a visually plausible in flight trajectory and keeps the birds united. This result shows that our system is also capable of generating flock motion from various kinds of bird flock in the input video.



(a) Input video (b) (c)

(d) Result (camera view) (e) (f)

(g) Result (top view) (h) (i)

Fig. 4.6: Comparison of Result 3 in frame 0, 30, 60.

Chapter 5

Discussion

Here, we discuss about limitations and future work of this research. Our system has four limitations:

First, our method uses frame-by-frame optimization as our core method for optimization. An error function is also designed depending on this method. This method makes the optimization can be done in real time for interactive use. However, this approach also makes the generated bird flock having a lack of globally planned trajectory. When optimization is done in every frame, since it only has the information on previous frames, considering data in future frames in advance becomes impossible. This may result in some unnatural results, such as an sudden displacement for avoiding collision. Lacking data in initial frames also makes it difficult to decide positions in those frames. As future work, we can improve the algorithm to optimize globally by reducing computation costs.

Second, bird tracking is also an issue to be solved in this research. Our optimization method can predict three-dimensional position of bird flock in real time. However, the time spend in bird tracking part is the bottleneck of the system. In our experiment, it takes nearly 30 minutes to complete the preprocessing on a 300-frame video by the tracking application we use. Furthermore, manually obtaining track data also takes about 1 minute for 1 bird in a 10-second video. The time needed for manual tracking also limits the number of birds to deal with in a single video. It is fine to track several birds by human eyes. However, when the number of bird increase to like 100 or more, it becomes impossible to track every bird in the flock. Our current system also only allows user to track one bird at a time. As future work, another approach for tracking and synthesizing these dense birds flocks should be proposed. Volume modeling technique in [17] might be useful for modeling dense bird flock.

The third limitation is the method of modeling bird flock. We believe that the method we use to model flocks can be further studied to synthesize better flock motion. In this research, we only consider trajectory smoothness and flock behavior, while environmental effect such as aerodynamics or obstacles is not considered. These factors can be crucial for generating flock motion. In addition, bird locomotion is also not considered while it is a critical part for creating natural flock motion. In addition, although the system allows user to change parameters to modify result, the ability of modification is still quite limited. Although we provide interface for adjusting parameters, the outcome of adjusting these parameters is not instinctive for user. As future work, user interface and parameter tuning can be further improved.

The last limitation in our system is that the assumptions about input video limits the usability of our system. We assume all birds stay in the view field of camera, and the camera stays unmoved during recording. Although it is possible to generate videos that satisfy the assumptions from flock simulation system, it is difficult to have such videos from capturing real bird flock with camera. It is common that birds leave and enter the view

field, which is not considered in our system, and bird videos taken with moving camera is also normal. These two assumptions make using real bird video as input difficult, and this is why we make most of the tests with videos generated by flock simulation. More processing needs to be done to eliminate these assumptions.

As future work, we think that the system can be further developed to receive hand-drawn sketch as input. Since human sketch is similar to two-dimension track data obtained from video, as shown in Figure 5.1, we think it is possible to develop an similar system that receives hand-drawn sketch and generate results with same method. As we discussed in Chapter 2, motion sketching has become an popular approach to create animations. In our method, we use the track data retrieved from input video. If the track data is replaced by hand-drawn sketch, the time spend in the tracking stage can also be reduced. However, since drawing bird trajectory by hand is not as instinctive as it seems, more works need to be done to achieve the goal. We consider the idea of sketch input as our future work.

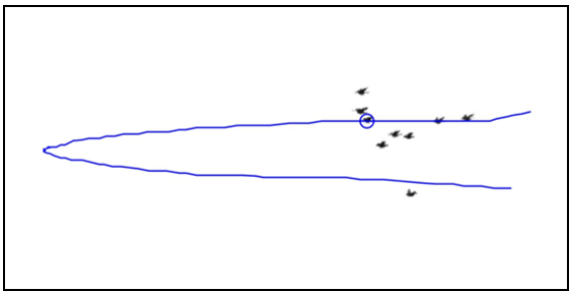


Fig. 5.1: Track trajectory of a bird.

Chapter 6

Conclusion

In this research, a novel approach for creating bird flock motion from video based on two-dimensional projection is proposed. The proposed system aims to help user to create and model bird flock motion by presenting a bird video as a reference. The system also provides interactive user interface to adjust the generated result in real time by tuning parameters. The benefits of our approach are twofold: providing a new way to create bird flocks by using video input as reference; and the ability to predict three-dimensional flock motion from two-dimensional track of bird flock. An interactive interface is also proposed for user to adjust parameters and see the result in real time to make better flock motion according to user's wish. We also show our generated results to validate that our system is capable of generating various kinds of bird flock from video. We hope that this research can improve the efficiency of creating rich scenes with bird flocks in game or movie industry.

References

[1] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, 21(4):25–34, 1987.

[2] Craig W. Reynolds. Steering behaviors for autonomous characters. *Proceedings of Game Developers Conference*, pages 763–782, 1999.

[3] Eunjung Ju, Jungdam Won, Jehee Lee, Byungkuk Choi, Junyong Noh, and Min Gyu Choi. Data-driven control of flapping flight. *ACM Transactions on Graphics*, 32(5), 2013.

[4] A. Fitzgibbon A. Buchanan. Interactive feature tracking using k-d trees and dynamic programming. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 34(93), 2006.

[5] Peter Molnar Dirk Helbing. Social force model for pedestrian dynamics. *Physical review. E*, 51:4282–4286, 1995.

[6] Gavin S. P. Miller. The motion dynamics of snakes and worms. *ACM SIGGRAPH Computer Graphics*, 22(4):169–173, 1988.

[7] Daiki Satoi, Mikihiro Hagiwara, Akira Uemoto, Hisanao Nakadai, and Junichi Hoshino. Unified motion planner for fishes with various swimming styles. *ACM Transactions on Graphics*, 35(80):169–173, 2016.

[8] Amato NM Bayazit OB, Lien JM. Roadmap-based flocking for complex environments. *10th Pacific Conference on Computer Graphics and Applications*, pages 104–115, 2002.

[9] Amato NM Bayazit OB, Lien JM. Swarming behavior using probabilistic roadmap techniques. *Lecture Notes in ComputerScience*, 3342:112–125, 2005.

[10] Zoran Popović Jia-chi Wu. Realistic modeling of bird flight animations. *ACM Transactions on Graphics*, 22(3), 2003.

[11] Chenney S. Anderson M, McDaniel E. Constrained animation of flocks. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 286–297, 2003.

[12] Jiayi Xu, Xiaogang Jin, Yizhou Yu, Tian Shen, and Mingdong Zhou. Shape - constrained flock animation. *Computer Animation and Virtual Worlds*, 19:319–330, 2008.

[13] Daniel Thalmann Branislav Ulicny, Pablo de Heras Ciechomski. Crowdbrush: interactive authoring of real-time crowd scenes. *SIGGRAPH '05 ACM SIGGRAPH 2005 Courses*, (3), 2004.

[14] Golaem. `http://golaem.com/`.

[15] Massive. `http://www.massivesoftware.com/`.

[16] Makoto Okabe Kei Iwasaki, Yoshinori Dobashi. Example-based synthesis of three-dimensional clouds from photographs. *CGI '17 Proceedings of the Computer Graphics International Conference*, (28), 2017.

[17] Makoto Okabe, Yoshinori Dobashi, Ken Anjyo, and Rikio Onai. Fluid volume modeling from sparse multi-view images by appearance transfer. *ACM Transactions on Graphics*, 34(93), 2015.

[18] Andrew Zisserman Andrew W. Fitzgibbon, Geoff Cross. Automatic 3d model con-

struction for turn-table sequences. *Lecture Notes in Computer Science*, 1506:155–170, 1998.

[19] Jeff Kuo Alexander P. Pentland. Three-dimensional line interpretation via local processing. *Electronic Imaging*, 1249, 1990.

[20] Takashi Ijiriand Shigeru Owada, Makoto Okabe, and Takeo Igarashi. Floral diagrams and inflorescences: interactive flower modeling using botanical structural constraints. *ACM Transactions on Graphics (TOG)*, 24(3):720–726, 2005.

[21] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.

[22] Zootracer. `https://www.microsoft.com/en-us/research/project/zootracer/`.

[23] Nlopt. `https://nlopt.readthedocs.io/`.

[24] M. J. D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. *Advances in Optimization and Numerical Analysis*, 275:51–67, 1994.

[25] Unity. `https://unity3d.com/`.

Acknowledgements

I would like to thank my supervisor, Professor Takeo Igarashi, for all the guidance and insight he provided me during the course of my master degree. Without his help, this thesis would hardly have been completed. I would also like to thank Tsukasa Fukusato for his advice and assistance in keeping my progress on schedule. I also want to thank I-Chao Shen for his useful suggestions.

Finally, I would like to express my greatest gratitude to my family for their continuous help and support. They encouraged me to explore new directions in life, making this amazing adventure of my life happen.