T-BIRD: Tracking Trajectories of Migrating Birds Around Tall Structures

Final Paper

Study by:
Arik W. Brooks
Nicholas J. Patrick

Project Advisors:
Dr. Brian D. Huggins
Dr. Donald R. Schertz
Dr. Thomas L. Stewart

Bradley University
College of Engineering and Technology
Department of Electrical and Computer Engineering

May 10, 2004

Abstract
This project involves the design an implementation of a system to track the trajectories of birds flying near tall structures to allow wildlife biologists to study bird kills near these sites. The positions of the birds are determined using stereoscopic vision by placing the two cameras a known distance apart in parallel with each other, and all processing is accomplished by a set of PCs connected via the Internet. The system output is a display depicting a three-dimensional fit of the trajectories, and data relating to the trajectories.
Introduction

Every year, migratory birds make a round trip between a northern area where they breed and an area far to the south where they then spend the winter. When making the trip, birds often pass near tall structures including skyscrapers and communication towers. Often, these birds are attracted to the structures resulting in large bird kills. One theory explaining this disturbing phenomenon is that the birds are attracted to the lights produced by these structures including lights left on in skyscrapers at night.

This project is a continuation of a study that was performed during the 2002-2003 school year at Bradley University by the senior electrical engineering team of Brian Crombie and Matt Zivney.

Objective

The objective of the project is to develop a system to assist biologists in their investigation of bird kills near tall structures by tracking the birds’ flight paths using a stereoscopic imaging approach. By using this method, the system can perform reasonably accurate measurements without the cost and setup requirements of a more sophisticated approach such as using a RADAR-based system. All measurements are made in real-time so that researchers can adjust environmental factors while reviewing the system output. Also, processing is split between two computers by transmitting information across the Internet.

The system calculates the position of each bird in three-dimensional space for every set of frames captured by two cameras set a known distance apart, one above the other. The trajectories are displayed in a representation of their three-dimensional positions over time along with other important statistical data that may be useful.

Significance of Research

Currently, there is no practical method for researchers to collect accurate data regarding bird kills during migration. In recent studies, researchers have relied on data from publicly available resources including Doppler RADAR readings intended for weather monitoring to collect data on bird migration patterns. This project is significant because it will provide a practical means for wildlife biologists to study the flight paths of birds flying near tall structures and their reactions to changes in parameters controlled by the researcher (i.e. lighting, sounds, etc.). This research could lead to a better understanding of bird behavior and the elimination of bird kills near tall structures. The software methodology used in this project is also of significance because it can easily be expanded to other tracking applications.
System Block Diagram

A system level block diagram of the system is shown in Figure 1. The system includes several hardware subsystems. Figure 2 shows their configuration and signal direction.

Figure 1: Overall Block Diagram

Figure 2: Hardware Block Diagram
Subsystems

- Cameras
- Frame Grabber
- PC’s/Network

Cameras
The camera subsystem includes two CCD monochrome video cameras that convert photons into data. The cameras are mounted in parallel a known distance apart on a boom. This allows objects to be located in three-dimensional space.

Inputs
- Photons -- Images from the environment within the field of view of the cameras
- Synchronization -- Images are captured synchronously by coordinating the capture based on the phase of the power signal.

Outputs
- Data -- Image data (NTSC format) transmitted to the frame grabber

Operation in System
- The cameras capture images at a rate dictated by the speed of the preprocessing algorithm

Frame Grabber
The frame grabber simultaneously captures images from both cameras and supplies the data to the preprocessing PC.

Inputs
- Data -- Image data from the cameras
- Setup -- Information from the PC

Outputs
- Image Data to PC

Operation in System
- The frame grabber operates at a rate dictated by the speed of the preprocessing algorithm

PC’s/Network
Two PC’s are networked together to divide computation between the preprocessing and trajectory calculation computers and allow remote analysis of the system outputs. The preprocessing computer extracts moving objects from the images provided by the frame grabber. The trajectory calculation computer calculates and displays trajectories and statistical data based on the information from the preprocessing computer.

Inputs
- Image Data -- Arrays of intensity information
- Calibration Input -- Calibration data for the cameras being used

Outputs
• Display – GUI showing trajectories plotted in a three dimensional representation
• Statistics -- Pertinent data calculated from bird trajectories
• Raw Data -- Data file containing all preprocessed data

**Operation in System**
• The PC’s and network operate continuously

**Methodology**

The project consists primarily of software algorithms and interfacing to implement a fast image processing system. The primary steps required to implement the system are as following:

1. Grab a set of images from the cameras at the same time
2. Identify and locate moving objects in each image
3. Find corresponding objects in the two camera images
4. Add objects to a trajectory based on the history of object locations
5. Display information and trajectories

The software is split into two sections: the image preprocessing written in C++ and the tracking algorithm implemented in MATLAB. The two sections are implemented on separate computers, which are connected via the Internet. The functionality of each section is described below:

**Preprocessing software**
• Grab images from both cameras simultaneously
• Threshold and convert each image to binary data
• Apply noise filter (Gaussian)
• Find centroids (x and y pixel locations) of all moving objects
• Transmit data to the computer hosting the tracking algorithm

**Tracking software**
• Receive centroids from preprocessing computer
• Save data for later use
• Find every possible 3D positions for the objects present
• Search for closest position to predicted position, within a threshold, for each trajectory based on its previous two object locations
• Search for objects that were first detected in the previous frame based on closest position and area within a threshold
• Search for new trajectories
• Display trajectories and statistics

**Preprocessing**
The preprocessor obtains images from the cameras through a dual input frame grabber that is interfaced to the computer. A rolling background image is created by averaging the images over time. This background is subtracted from the image to create contrast between moving objects and “static” scenery. Each image is thresholded to create a binary representation of the scene where values of 1 (white) represent movement. A routine is then executed that searches through
the image to obtain the pixel location of the center of each object. This routine calculates the center position by tracing only the perimeter of each object, using an algorithm to determine the center based on this data. A TCP/IP socket is opened when the code is executed, and it is used to transmit the center positions of each object to the tracking computer. The interface for the preprocessing is a graphical user interface (Shown in Figure 3), which contains controls to operate the system.

The main requirement for the preprocessing software is fast operation. A high frame rate is necessary for accurate tracking of fast-moving objects so that an adequate number of location points can be found to identify the trajectory correctly. High-resolution images are also required to allow objects to be detected accurately and at a reasonable maximum distance. This translates into a large amount of data needing to be processed for each set of frames, and the software must be optimized to allow for this processing.

The background subtraction routine was optimized by minimizing the number of multiplies and divides required per pixel. This is accomplished by using only multiplication and division by powers of two, which can be replaced by shift operations. The background image is contained in bits 8 through 15 of a matrix of integers. Each time a frame is added to the background, the current pixel value is added to the 16-bit background and the background is shifted down 8 bits and subtracted from itself. The effective operation performed by this action is to multiply the background by 255, add the current image, and divide the sum by 256 to create the new background.

\[
\text{Background} = \text{Background} - (\text{Background} >> 8) + \text{Current_image} \quad \text{(Eq. 1)}
\]

This method stores the background in 8 bits with 8 extra bits for accumulating round-off error. The variable “Background” in equation 1 above includes both the 8 bits for the background image plus the 8 bits of round-off error. Using shift operations instead of multiplication and division allows for faster operation on individual pixels, which speeds up the program significantly, especially since the number of pixels processed per iteration is very large (640×480×2 = 614,400 pixels per set of frames).

After the background subtraction, the images are thresholded to clearly identify which pixels represent an object that is moving. After thresholding, the image is represented in binary with white pixels corresponding to moving objects and black pixels corresponding to the background.

The preprocessing software then searches the image for white objects and determines their center position. The function begins by searching the image for a white pixel. When a white pixel is found, it follows the perimeter of the object until it reaches the beginning point. With this method, the center of an object can be found quickly. This method is an improvement upon the previous method that averaged the positions of every connected white pixel in an object.

A software flow chart showing the operation of the preprocessing software in a graphical manner is included in Figure 4.
Figure 3: Preprocessing program user interface

Figure 4: Preprocessing software flow chart
Trajectory Calculation
The tracking algorithm in MATLAB receives the center pixel locations of each object via a TCP/IP socket connection. This data consists of all object center locations from both the upper camera image and the lower camera image. This data is immediately saved to a file for later analysis. The algorithm then takes all valid pairs of upper and lower camera objects and finds the corresponding location in space. To be considered a valid pair, the horizontal variation of an object’s center location must be less than 30 pixels between the upper and lower camera images. Figure 5 shows the camera image orientations and the equations used to transform pixel locations in two images to X, Y, and Z coordinates. The equations were found by the previous team of Brian Crombie and Matt Zivney on the Cooper University website, www.ee.cooper.edu, in the week 5 lecture notes for EE 458.

\[ X = \frac{X_d \times d}{X_d - X_u} \]
\[ Y = \frac{Y_d \times d}{X_d - X_u} \]
\[ Z = \frac{d \times f}{(X_d - X_u) \times \text{conversion factor}} \]

\( f \) = focal length

Figure 5: Diagram and equations to determine an object’s spatial location (X, Y, Z) from its pixel location in two camera images (X_u, Y_u, X_d, Y_d)

The tracking algorithm also examines the locations of objects over time and identifies objects belonging to the same trajectory based on the assumption that bird movement can be approximated as linear motion over small time increments. This is obviously impacted by the maximum and average flight speed of migrating birds. The algorithm finds new points in a stored trajectory by making a linear prediction of the next bird location and searching a region around that predicted position for a new point. New trajectories are recorded when three points in a row meet a set of criteria defining how much variation in distance and non-linearity is allowable.

Also, there are many contingencies built into the software that make the tracking algorithm very resistant to lost or erroneous data points, which will be numerous in this type of image processing application. For example, when no object is found near the location predicted by the previous points in a trajectory, an object is added at the predicted position (but not recorded or displayed). This added position is used to search for the next point in the trajectory, thus allowing for recovery from the object not showing up in a single frame.
The equation that is used to determine the predicted position of an object is shown below (Equation 2). The predictive search will be based on a velocity vector from the previous two locations of each object. If an object is found within a particular distance of the predicted position, it is considered to be the same object.

\[
\text{Predicted position}(t) = \text{position}(t-Td1)+\left(\frac{\text{position}(t-Td1) – \text{position}(t-Td2)}{(Td2-Td1)}\right)*(Td1) \\
\text{(Eq. 2)}
\]

\( Td1 = \text{time difference between current frame and previously grabbed frame} \)
\( Td2 = \text{time difference between current frame and 2nd previously grabbed frame} \)

The radius of search around the predicted position is determined based on average bird velocity, time between frames, current velocity, and distance from the cameras.

The final tasks performed by the tracking computer are saving the trajectory data, printing the statistical data as it is calculated (max. bird speed, closest bird location, etc...), and plotting the trajectories. This data is displayed in a graphical user interface window, which also contains user controls for the program (shown in Figure 6). Much like the preprocessing software, the plotting section of the tracking software had to be optimized for fast operation to allow for real-time results. To accomplish this, the software generates all the plots upon start-up and simply writes white over the old plots before drawing the new data on the plots. By taking this approach, slow redraw commands are eliminated from the code, allowing for real-time plotting.

Figure 7 shows a software flow chart of the steps in the trajectory calculation software.
Results

The preprocessing software effectively isolates moving objects from the background and locates their center positions at a rate that is acceptable for real-time operation. An example of the final output from the preprocessing code is shown in Figure 8.

Figure 7: Trajectory calculation software flow chart

Figure 8: Example of final output from preprocessing code (actual output is the x and y pixel locations of the crosses displayed on the plot)
Following is a speed-comparison of the preprocessing software from last year versus the new preprocessing software.

**Speed improvements (640x480 images, no find object routine)**

| Old: | 10.6 frames per second |
| New: | 15.9 frames per second |

**Updating average every 60 frames**

| New: | 24 frames per second |

**With find object routine added**

| New: | 18 frames per second |

The final system with full functionality operates at approximately 17 frames per second on average with a few moving objects within the few of view of the cameras.

The trajectory calculation code is able to identify individual trajectories from images containing multiple objects traveling in various direction most of the time. A comparison of the new tracking algorithm finding trajectories of two swinging tennis balls versus the system from last year calculating trajectories from the same data is shown below in Figures 9 and 10.

**Figure 9: Output of old system tracking algorithm for two swinging tennis balls**
Two accuracy tests were conducted on the system. The first test measured the system’s accuracy in predicting the position of a single “slow-moving” object. To perform this test, a wire was hung from the ceiling in the laboratory and marked at five heights relative to the lower camera location: 1 meter above and below, 0.5 meters above and below, and at the level of the lower camera. Then, this wire was placed at a known distance from the cameras, and an object was slowly extended and removed from the field of view of the cameras at each of these known heights and distances. The results of this test are shown in Figure 11 for distances of three, six, and nine meters.
The second accuracy test involved dropping a ball from the ceiling and comparing the motion recorded by the system to that predicted by the laws of motion. Difficulties arose in conducting this test because the time of release of the ball could not accurately be determined. Predicting a falling object's location based on the laws of motion requires knowing this time (time=0), so an assumption was required to evaluate the system. This assumption is that the first point measured by the tracking system is absolutely correct. By making this assumption, the drop time can be reverse calculated and further points in the drop can be predicted based on the time between frames. The results of this test are shown in Figure 12 for distances of three, six, and nine meters.

Figure 11: Results of accuracy test #1

\[ X = 0 \text{ is at a height equal to the height of the lower camera, which is at } 5 \text{ feet from the floor.} \]

Figure 12: Results of accuracy test #2
A simple test was also performed to evaluate the system’s performance in tracking multiple objects at one time by flying four paper airplanes across the field of view of the cameras. The system was able to recognize each of the planes and plot their trajectories as they arced towards the floor. A plot of the four planes’ trajectories is shown in Figure 13.

![Figure 13: Trajectories of four simultaneous paper airplane flights](image.png)

### Datasheet

- **Average Migratory Bird Diameter** .................. 0.152 m
- **Average Migratory Bird Speed** ..................... 8.9409 m/s
- **Max # of Objects Tracked Simultaneously** .......... 30 (based on software set limit)
- **Max Distance from Cameras** .......................... 20 m
- **Min Distance from Cameras** .......................... 3 m
- **Max Location Error** ................................. 0.375 m (theoretical)
- **Light Level Sensitivity:**
  - **Lab Cameras** ......................................... 0.22 Lux
  - **Low Light Cameras** .................................. 0.0002 Lux
- **Max Framerate** ....................................... ~15 FPS
- **Total Volume of Space Observed** .................. 606 m³

*Separation of Cameras assumed for calculations is 0.5 m*
Suggestions for Further Work

- Implement rotatable boom to adjust field of view from the computer application (mechanical system and controls)
- Obtain and integrate high end cameras
- Optimize code (analyze algorithms, streamline processes)
- Port MATLAB to C++
- Investigate feature detection methods for improved target recognition
- Add software for offline processing of stored files
- Setup system and perform testing in the actual environment

Conclusions

This study proves that real-time tracking of multiple objects can be accomplished using a stereoscopic imaging approach. The system has reasonable accuracy for a small number of objects (2 or 3) moving at a slow velocity (less than 4 meters/sec) in a laboratory setting. Further investigation of the system’s accuracy for more bird-like movements was not possible in the time allotted, but the system’s accuracy may be sufficient for collecting usable data about bird flight paths. It was not possible to determine the performance of the system in the required environment for the proposed application of tracking of birds near tall structures on overcast nights.
References

http://www.intel.com/research/mrl/research/openCV/
Pinhole camera model, image processing reference.

http://www.digibird.com/primerdir/eqn.gif
Equations relating focal length to zoom

http://www.ipsimaging.com/support/camerasensitivity.htm
Light levels for various time of day and weather conditions.

http://sportscience.org/adi2001/adi/services/support/faq/software_genlock.asp
Estimating position when synchronized cameras are not available.

http://www.fmsystems-inc.com/vtmtips_article.htm
Using line lock cameras.

Equation relating focal length to target object size, distance, and CCD width.

http://www.machinevisiononline.org/public/articles/cohu.PDF
Measurements for various CCD sizes.

Project proposal from previous group

Chen, Tieh-Yuh; Bovik, Alan Conrad; Cormack, Lawrence K. “Stereoscopic Ranging by
Matching Image Modulations,” IEEE Transactions on Image Processing. Vol 8, # 6,
June 1999, pg 785-797.

Standards

There are no overarching standards that apply to bird tracking, but several standards are used to
interface cameras to the PC.

NTSC
The cameras selected produce NTSC compatible signals, which is the standard in North
America. The Frame Grabber converts NTSC inputs to digital images.

DirectX
DirectX is a defacto standard for Microsoft Windows, which includes a programming interface
to video capture devices such as frame grabbers. DirectX was chosen over proprietary APIs to
maintain a maximum amount of hardware independence.
Patents

<table>
<thead>
<tr>
<th>Patent #</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>6,366,691</td>
<td>Stereoscopic image processing apparatus and method</td>
</tr>
<tr>
<td>6,028,954</td>
<td>Method and apparatus for three-dimensional position measurement</td>
</tr>
<tr>
<td>6,035,067</td>
<td>Apparatus for tracking objects in video sequences and methods therefor</td>
</tr>
<tr>
<td>5,812,269</td>
<td>Triangulation-based 3-D imaging and processing method and system</td>
</tr>
</tbody>
</table>