Abstract—With the advent of streaming platforms, the companies that develop miniatures games and those who play them have made numerous attempts to stream their events. While these streams are interesting, lack of access to critical information prevents any announcers from being able to give in-depth game analysis to the viewer. The application described here attempts to give announcers access to this critical information, allowing for a much improved experience for the viewer.

Index Terms—Object Tracking; Image Analysis; Warmachine; Miniatures Games

I. INTRODUCTION

In recent years the companies that develop miniatures wargames and their players have made many attempts to share their games online. In some cases this involves a live stream of the game with experts announcing and giving their opinions on the state of the game, similar to how sportscasters discuss events as they are played. In other situations players will record a game they played then come back to it and discuss the game in review. Some players will also record their own games as a method to look back and try to improve their own play.

All of these situations suffer from some similar issues. In all cases the person observing or reviewing the game does not have access to critical data that would allow them to give a more informed review of the game in question. This project is an attempt to get that information to the observer, allowing them to be able to more accurately analyze the game. The software does this through analysis of images and live video.

II. BACKGROUND

A. What Are Miniatures Games

A miniatures game is a tabletop game in which each player is controlling one or more models and is attempting to defeat their opponent. Games of this type have existed for ages with the first published game of this type, Little Wars by H.G. Wells, releasing in 1913 [1]. In the modern era the market has numerous options for this style of game. The one generally considered the most popular, Warhammer 40,000, is published by Games Workshop and is a sci-fi game. However there are games to fill most fictional niches such as steam punk with Privateer Press’s Warmachine, high fantasy with Games Workshop’s Warhammer Fantasy, sports with Steam Forged Games Guildball, or spaceship combat with Fantasy Flights X-Wing, just to name a few.

While rules vary greatly between games in this genre they generally have some basic similarities. First there is generally a way to determine the strength of a player’s models based on a point value. If each player brings the same total number of points then they should each have an army of the same strength. The board the game is played on varies in size based on which game is being played. However in most miniatures games there are terrain elements on the board. For instance certain areas will be forests, hills, buildings, etc. which then effect gameplay in various ways; hiding in a forest makes models harder to hit, standing on a hill gives models some form of bonus to hit enemies on lower ground, or hiding behind a building may mean enemy models cannot target you at all. The goal of the game can also vary between systems. In some cases the goal is to eliminate all of the opposing models while, in other cases models need to capture certain areas on the board designated by scenario elements. Another important aspect to note for these games is that they rarely use any form of grid for positioning. Generally all measurements are done in inches with absolute positioning. For instance a certain model can move 6” every turn and its gun can shoot targets 12” away.

There are a few key pieces of information that cannot be easily inferred from a video or photo of a game. First there are times when an observer cannot quickly locate a model on the playing field, second they need the ability to accurately judge distances from models, and third in many of these games they need the ability to determine line of sight; A model cannot shoot something it cannot see. Giving the observer easy access to this information for improved live or post-game analysis is the goal of this project.

For this project I am focusing on only one game, Warmachine & Hordes by Privateer Press, to help create a specific set of requirements. Warmachine & Hordes was first released in 2003 [2] and continues to be a very popular game today. Most importantly it is one of the most commonly streamed games of this type.

III. SETUP

One of the greatest strengths of miniatures games is the variety of boards you can play on and places you can play.
In order for this project to see adoption it needs to work in as many situations as possible. However there are some requirements: first I am restricting myself to a single 1080p webcam suspended directly above the play area shown in Figures 1 & 2. While I could gain more detailed images from a higher resolution camera, the increased cost of the camera as well as the increased processing power necessary to keep up with live video would make the setup prohibitively expensive for many users. Aside from the camera suspended above the play area, the user also needs a 5” circle template. These are standard to Warmachine and all players should have access to one. Next the user needs 4 tokens of some form to denote the corners of their play area. I used 1” red square tokens, however, the size, shape, and color are irrelevant.

The software guides the user through the following setup steps:

1) Record an image of the blank game board with nothing on it (Figure 3)
2) Place the 5” template on the board and record the image (Figure 4)
3) Place one token in each corner of the board and record the image (Figure 5)
4) Place any scenario elements that are static and do not affect line of sight and record the image (Figure 6)
5) Place all terrain elements and record the image (Figure 7)
6) Play may now begin (Figure 8)

Each step is described in more detail in the Implementation section.

IV. IMPLEMENTATION

The implementation of this project has 3 main sections. The first is live analysis, the second is a simple game recording, and the third is post game analysis. All three depend on the steps listed in the Setup section. The software was developed for Windows with python making heavy use of OpenCV for the image analysis.

1) By recording an image of the empty board, the software
2) After the user has placed the 5" template, the software finds the differences between the two images. Since the only difference is the addition of a template with known dimensions, the software uses this change to determine the number of inches per pixel. This allows the software to achieve one of our first goals, access to distance measurements.

3) By identifying tokens at the corners of the game board, the software is able to focus the algorithms down to only the area of the image that matters. The software does this in a way similar to how it finds the 5" template by simply finding the changes in the image. While it is theoretically possible to have the camera only contain the game board, that would require very precise positioning of both the camera and the board. In many cases even with perfect positioning, the camera’s resolution would not match the dimensions of the board.

4) Placing scenario elements separate from the rest of the terrain has two purposes. First these elements rarely effect gameplay outside of scoring so the software does not need to monitor them for cases like line of sight. Second there is often terrain placed inside scenario elements so the software needs to log this image to find those changes in the next step.

5) Once terrain is placed the software compares this image to the one containing the scenario elements to determine what was just placed on the board. Once that is done the user has the ability to define what each piece of terrain is, which allows the program to take those into account when determining line of sight. See Section V for a description of how the line of sight rules work.

6) Finally the game can begin. As the user adds models to the game they can label each of them, which gives the software a template to use to attempt to track the models throughout the game.

### A. Live Analysis

In live analysis I assume there are two people playing and a third person making use of the software. This person could be streaming what is happening to an audience or recording it to place on a streaming service at a later time. Setup happens as indicated in Section III and the user has the option to record an image at any point in the game for later use in the review system.

### B. Recording a Game

The software at this point is not trying to perform any work on the images it is receiving instead it simply logs an image any time one of the players wants it. These images will then be
recorded to the directory the user chose and used for the review system in the same way the live analysis records images.

C. Reviewing a Game

When reviewing a game the software is pulling in a set of images recorded by either the live analysis or the recording feature. It goes through the 5 setup images and then allows the user to move forward or backward through the recorded images.

V. Features

Live analysis and reviewing a game have the same feature set: the ability to detect models, the ability to track models, the ability to check distances from a model, and the ability to check line of sight from any point.

A. Model Detection

In order to solve this problem I am taking advantage of one of the traits of Warmachine; all models are on circular bases that are either 30mm, 40mm, 50mm, or 120mm in diameter. To easily find these models the software performs what is called a Hough Transform [3]. In simple terms a Hough Transform takes an image, and using edge analysis, locates circles of a certain size. By running this over the image the software can locate the models being played. Since the majority of the model should be contained within this circle we can create very exact templates to use for the tracking features. This in turn greatly reduces the overhead for tracking the models when compared to trying to find features across the entire image. Due to the nature of the models used in the game, I run the Hough Transform with very loose parameters because quite often models in this game have parts overhanging their base which causes them to appear as incomplete circles from above. This means that the software also gets a large number of false positives in each frame.

B. Tracking models

Once models have been placed on the board the user can begin labeling them. They fill in a name for the model, choose the model size, and click on the model in the image. This cannot be automated because one of the features of this game is the ability for players to paint and pose their models in any way they would like. Players even have some ability to replace pieces of the model, making any form of automated identification far outside the scope of this project.

Once the user has labeled a model the software needs to match it to the same model in the next frame. In the case of post-game review that next frame will be the next image the user saved. For the case of a live analysis it will be the next frame from the camera. To match the circles to each other I am using ORB Feature Tracking [4]. ORB was specifically designed as a feature tracker that should run fast enough to work on live video, allowing the software to match the features between frames. In simple terms ORB feature matching works by taking an image and analyzing it for key points. It then can compare one image’s set of key points with another image’s and returns the best matches and the “distance” between them. Distance in this case means how similar they are to each other. The shorter the distance the more similar the key point.

Once a model has been labeled the software saves that image as a template to be used in all future comparisons. With each new frame to analyze the software compares the model’s template to all templates of the same size in the new frame to determine the model’s new location. The software uses several criteria to perform the match. First and foremost is ORB Distance. Specifically the software totals the distances for each matched key point, then divides it by the total number of matches to get an average difference between the two images. Next the software checks the match percentage. If less than 35% of the key points of either the template or the circle the software is matching to were not matched, the software throws the comparison away. Next the software checks to see if the circle it is comparing to existed before models were put on the board (the image recorded in step 5 of setup). If so the software increases the ORB Distance to reflect the fact that the circle may be an already known false positive.

I considered and tested several other criteria but none of the following improved the matching rate. Using circle location as a factor did not improve performance because of the likelihood of the cameras view to models being blocked during the course of normal play. Anything related to model location based on legal moves was not considered due to the increased amount of information that would need to be added to the game, since there are several hundred unique models that could be used. Color information such as average value per channel did not improve things beyond what the ORB matching was doing.

Once the software has done the comparisons for each model labeled to each potential model found in the current frame it goes through the returned ORB Distances to find the shortest. That location is labeled as the model, and then the software goes through the list of returned distances to find the next best model match and label that location as a model. This repeats until all models have been identified or there are no longer any good matches in which case I assume the model has been removed from the board and remove it from the displayed list of models.

C. Distance Measurements

In step 2 of setup I place a 5” template on the game board (Figure 4). This template is one of the basic implements used to play the game, so every player should have easy access to one. When the template is placed on the board the software compares the two images to find the difference. From there it figures out how many pixels across the template is and divides 5 by that number to calculate the number of inches per pixel.

With this the software knows how many inches each pixel represents, so determining distances from models becomes trivial. When the user has selected a model you can simply put a number in the text field and it will display a ring around the model that many inches away. In tests this ring was generally very accurate and allows any observer to have easy access to
what is arguably the most important bit of information when playing the game.

D. Line of Sight

In step 5 of setup (Figure 7) the user places any terrain elements to be used on the board. In terms of line of sight, terrain can have 3 possible effects. Elements like water, hills, and rubble have no effect on line of sight. Elements like clouds and forests can partially block line of sight. The rules state that a model can see up to 3” into a forest or cloud but not completely through unless it is within. Finally obstructions completely and immediately block line of sight. These are generally objects like giant rocks or houses. For an example see Figure 9. In this example model A can see model B in the forest since there is less than 3 inches of forest in the way. Model A cannot see model C because there is more than 3 inches of forest between them. Model A can see model D because there is nothing in the way. Model A cannot see model E because the house is an obstruction which completely blocks vision.

When terrain is placed the software compares the image with the image from step 4 and notes the differences. The user can then label the terrain. This cannot be an automated process due to the huge variety of terrain people use a lot of which is often homemade.

Once terrain has been labeled by the user the software can determine the line of sight from any point on the table. The user simply clicks on the board and the software determines line of sight from that point. To do this I use a modification to Bresenham’s Line Drawing algorithm [5]. I start at the point clicked then check each adjacent point forming a square around the clicked point. For each of these points I check to see if they are contained within any piece of terrain. Next I use the modified Bresenham to find out what point is on the line back to the origin and check what that point’s line of sight value was. Depending on these two values I calculate the line of sight value for this point then record it. Once each point in this square has been analyzed I then move out to the next ring and repeat until I have determined line of sight from the click point to everywhere on the board.

Once line of sight for the whole board has been calculated I black out anything that cannot be seen from the origin point and display it to the user. This allows the user to easily determine what a model can see further allowing them to understand the game state.

VI. Results

This project had 4 pieces of information it was attempting to extract from gameplay images; model locations, model identifications, distance measurements, and line of sight calculations.

Locating potential models works well. In the vast majority of frames all models that exist in the image are detected. However as discussed earlier, due to the inconsistency of the bases appearance from above, the Hough Transform is run with such loose parameters that there are a large number of false positives.

The biggest challenge in this project by far is the matching of models between frames. The 30mm size models are so small when viewed on a 4’ by 4’ table that it is very difficult to get enough detail from the model to truly track it. In live video the tracking failed in more frames than it succeeds for the 30mm and 40mm size models. However the software has good accuracy with the 120mm models which indicates that it is less a problem of the algorithm and more a problem of image quality and detail.

Because of this, in order for the software to be useful the ability to relabel models was added to the review mode. This serves 2 purposes. First when a user relabels a model, it gives the software another image to compare and thus greater accuracy in future frames. Second this manual identification allows a user to in the worst case scenario manually label each model in each image.

In order to improve the usability of the live mode the ability to freely select from the located circles was added. This allows the announcer to pick out models they see and perform distance judgments from them without the software actually knowing what model it is.

The distance measurements themselves have been extremely useful and nearly as accurate as the measurements that occur directly on the table. This section is a complete success.

Line of sight calculations work reasonably well. The issues stem not from the algorithms but rather from the changes in image. It is quite often that the changes introduced by placing terrain on the board are not drastic enough to create perfect terrain definitions, which causes the line of sight display to look fragmented but they are generally accurate.

A. User Reviews

I had the opportunity to use this software several times in the local area, and some post game reviews I recorded were posted online to gather feedback from the community. These were some of the responses:

“This project could quickly become a staple of the wargaming/bat-rep community at large. Please finish this killer application!”

Rochester Institute of Technology
“The information provided definitely improves the watchability of games”
“It seems really cool”
In general all feedback from the community was very positive.

VII. Future Work
I plan on continuing to develop this software and hope to reach a point where I think it is good enough for a commercial release by the end of the year. While there are numerous usability improvements to be made, the core functionality is there. I intend to continue working to refine the model tracking by testing with a better quality camera and adding additional criteria to match on. Model detection may be improved to decrease the number of false positives. I also would like to improve the line of sight displays so that they look smoother and more refined. The way the image turns out currently could be much cleaner.

VIII. Conclusion
While the model tracking was not able to become accurate enough to be an asset to the user, of the 3 desired pieces of information, it was the least important because model identification is the part that humans are best at. Conversely the line of sight calculations and the distance measurement functionality both work very well and do a lot to improve a watcher’s experience. With access to this information announcers will be able to give their audiences informed opinions on the different options available to players. Post-game battle reviews become far more informative for the user because they no longer have to guess at what other options existed at certain points in the game.

Acknowledgment
I would like to thank Privateer Press for their continued work on a truly excellent game, Just Games Rochester who allowed me to set this up in their shop several times for real world testing, and my local play group for putting up with me while I was working on this.

References
[1] H. G. Wells, Little wars, by h.g. wells. F palmer, 1913.