A Curling Shot Tracking Program

Reggie Wilcox, Stanford University, rgwilcox@stanford.edu

Abstract—We present an algorithm for tracking the position of a curling stone from broadcast footage of curling events. Due to the constant panning of the camera in addition to perspective changes in the image, this introduces a need for image rectification and tracking of the camera view on the ice. We break the algorithm into four distinct components: rectification, frame tracking, stone tracking, and localization. The resulting algorithm has great performance in tracking the stone as it travels down the ice. This has promising applications for advanced analysis of curling shots, including the prediction of stone paths and precise quantification of the effect sweeping has on the stone.

Index Terms—Curling, Object Tracking, Rectification

I. INTRODUCTION

While viewing live feeds of curling events, it can be desirable to track the progress of the stone on the sheet. The cameras are not set up to provide this kind of information since they are designed to view a small section of the ice to provide a detailed view of the stone and sweepers. To do this, they rotate to follow the stone as it moves down the ice. This creates two major problems for tracking a curling stone: the perspective of the image is constantly changing and the visible part of the ice is constantly changing.

This means that an algorithm which can track the location of the stone in the current field of view is not sufficient for tracking the stone’s position on the ice. We need additional information about which portion of the ice is in view to pinpoint the exact location of the stone. The changes in perspective add a further challenge since this warps the stone and the ice features in addition to changing the scale of image. To overcome these challenges, we utilize a four part algorithm which can accommodate for the perspective and view change.

We provide details of the mathematics that we utilize during image rectification in Section II. We present the details of each part of the algorithm in Section III. Section IV discusses the results of applying this algorithm on real curling videos. Finally, we present some interesting directions for future work in this area and some of their applications in Section V.

II. PERSPECTIVE BASICS

Here, we discuss the mathematics behind the rectification algorithm. We provide a simple model for the ice and camera setup and give a derivation of the equations used to scale the image.

A. Camera Angle

We first derive how to calculate the angle the camera is pointed at based on the angle that the bumpers are angled at. Let \( d \) be the height of the camera above the ice, \( \theta \) the angle of the camera from vertical, \( w \) the width of the ice sheet, and \( a \) the distance from the camera to the point on the ice it is aimed at. For convenience, we consider the camera pointed in the \( +\hat{z} \) direction located at the origin and rotate the sheet. We can describe an un-rotated sheet by the points \( p \) which satisfy

\[
\mathbf{n}_0 \cdot (p - c) = 0,
\]

where \( \mathbf{n}_0 = (0, 0, 1) \) is the normal to the plane and \( e = (0, 0, a) \) is the distance from the camera. Note that

\[
a = \frac{d}{\cos \theta}.
\]

Define:

\[
R = \begin{pmatrix}
1 & 0 & 0 \\
0 & \cos \theta & \sin \theta \\
0 & -\sin \theta & \cos \theta
\end{pmatrix}
\]

Then the normal to the rotated plane is given by \( \mathbf{n}_\theta = R\mathbf{n}_0 \) and the rotated plane is given by \( \mathbf{n}_\theta \cdot (p - c) = 0 \).

We will look at where the points on the bumpers map due to the projection. First consider the location of the points on the un-rotated sheet. These are described by points with \( x \) coordinate \( \frac{w}{2} \) and \( z \) coordinate \( a \). We need only two points to figure out how the bumper will look, so we choose:

\[
p_1 = \left( \frac{w}{2}, h, a \right), \quad p_2 = \left( \frac{w}{2}, -h, a \right)
\]

Choice of \( h \) is arbitrary as it cancels out of the equations later. Now, when we perform the rotation of the plane by \( \theta \), these points map to \( p_{1,\theta} = R(p_1 - c) + c \). Thus, the points become:

\[
p_{1,\theta} = \left( \frac{w}{2}, h \cos \theta, a + h \sin \theta \right)
\]

\[
p_{2,\theta} = \left( \frac{w}{2}, -h \cos \theta, a - h \sin \theta \right)
\]

We are interested in finding how these points appear to the camera. To do this, we form rays originating at the camera which pass through these points, then find out where they intersect a plane normal to the camera. For convenience, we choose this plane to be unit distance from the camera, so it is described by points \( p \) where \( \mathbf{n}_0 \cdot (p - e_3) = 0 \), with \( e_3 = (0, 0, 1) \). A ray passing through \( p_{1,\theta} \) is parametrically defined by:

\[
\ell_{1,\theta}(t) = p_{1,\theta}t
\]
We first find the value of \( t \) for which each ray intersects the plane at unit distance from the camera. This requires:

\[
n_0 \cdot (p_{i,\theta}t - e_3) = 0
\]

Letting \( t_i \) be the solution for \( p_{i,\theta} \), we have:

\[
t_i = \frac{n_0 \cdot e_3}{n_0 \cdot p_{i,\theta}}
\]

We then let \( u_i \) be the coordinates of the intersection with this plane. We thus have:

\[
u_i = p_{i,\theta} \frac{n_0 \cdot e_3}{n_0 \cdot p_{i,\theta}}
\]

Evaluating this result for each point gives the following values for the points:

\[
u_1 = \left( \frac{w}{2(a + h \sin \theta)} \left( \frac{1}{a - h \sin \theta} - \frac{1}{a + h \sin \theta} \right) \right)
\]

\[
u_2 = \left( \frac{w}{2(a - h \sin \theta)} \left( \frac{1}{a + h \sin \theta} + \frac{1}{a - h \sin \theta} \right) \right)
\]

The \( x \) and \( y \) components of each point correspond to the \( x \) and \( y \) location in the cameras frame view. We can use these points to find the angle of the line connecting these two points from vertical, denoted by \( \alpha \):

\[
\tan \alpha = \frac{|u_{1,x} - u_{2,x}|}{|u_{1,y} - u_{2,y}|}
\]

\[
= \frac{w}{2} \left( \frac{1}{a - h \sin \theta} - \frac{1}{a + h \sin \theta} \right)
\]

\[
= \frac{h \cos \theta}{2a \cos \theta} - \frac{1}{a - h \sin \theta} + \frac{1}{a + h \sin \theta}
\]

\[
= \frac{w h \sin \theta}{2 a h \cos \theta} = \frac{w}{2d} \sin \theta
\]

Letting \( \beta = \frac{2d}{w} \) and solving for \( \theta \), we have our equation for the camera angle:

\[
\theta = \sin^{-1} (\beta \tan \alpha)
\]

**B. Calculation of \( \beta \)**

We note that in our above analysis, we require knowledge of \( \beta \). Since this ratio remains fixed for a curling arena, we need only evaluate it once. We derive a method of calculating it from two camera angles at a known distance from the camera.

Consider two camera angles, \( \theta_1 \) and \( \theta_2 \). We assume that we know the distance from the point directly underneath the camera and the point on the ice the camera is pointed at. We let these distances be \( y_1 \) and \( y_2 \) for \( \theta_1 \) and \( \theta_2 \), respectively. We let \( \lambda = \frac{y_2}{y_1} \). Let \( \alpha_1 \) and \( \alpha_2 \) be the angles of the bumpers from vertical in the camera image.

Since \( y_i = d \tan \theta_i \), we know that:

\[
\tan \theta_2 = \lambda \tan \theta_1
\]

**C. Calculation of the Scale**

Finally, we need to determine what the scale in the \( y \) direction is. That is, we want to determine the ratio \( \frac{\ell_0}{\ell} \) where \( \ell_0 \) is the \( y \) length of the at the same distance from the camera with no rotation and \( \ell \) is the length with the camera at an angle of \( \theta \). We assume that we know the field of view of the camera in the \( y \) direction, denoted \( \gamma_y \). Again, \( d \) is the height of the camera and \( a \) is the distance from the camera, so that \( d = a \cos \theta \).

This gives us the relations:

\[
\ell_0 = 2a \tan \gamma_y = \frac{2d \tan \gamma_y}{\cos \theta}
\]

\[
\ell = d \tan (\theta + \gamma_y) - d \tan (\theta - \gamma_y)
\]
Fig. 4. Visual representation of algorithm showing how the different components work together

We can use this to find the ratio between the distances:

$$\frac{\ell}{\ell_0} = \frac{d \tan(\theta + \gamma_y) - d \tan(\theta - \gamma_y)}{2d \tan(\gamma_y) \cos^2 \theta}$$  \hspace{1cm} (24)

Canceling some terms and simplifying using the relation

$$\tan(u \pm v) = \frac{\tan u \pm \tan v}{1 \mp \tan u \tan v},$$

we get the form:

$$\frac{\ell}{\ell_0} = \frac{\cos \theta(1 + \tan^2 \theta)}{1 - \tan^2 \theta \tan^2 \gamma_y}$$  \hspace{1cm} (25)

III. THE ALGORITHM

As discussed, there are four main components in the algorithm for tracking. We give a detailed description of the inner workings of each component.

A. Rectification

Rectification is the first component of the algorithm applied to the frames. This removes the perspective from the images and normalizes the scale so that the images are easier to work with in the subsequent components. There are three main steps in performing the rectification: binarization, angle calculation, and image warping. To determine the perspective in the image, we rely on the bumpers on the side of the ice sheet. Binarization is used to select out the bumpers from the image. Angle calculation determines the angle of these bumpers and from this, infers the angle of the camera. Finally, we perform an image warp to remove the perspective induced by this camera angle and normalize the scale of the image in the y direction.

1) Binarization: Inspection of the images reveals that the bumpers have a distinctive blue color. This suggests that color binning would prove effective in isolating the bumpers from the rest of the image.

To train our color binning algorithm, we took 11 frames from the videos at different positions on the ice and manually drew in the boundaries of the bumpers to create a mask. We bin the colors in 16 different bins for each color component, giving 4096 different bins. We count the number of pixels in each bin that are bumpers and those which are not bumpers. Let $c_i$ be the count of bumper pixels in bin $i$ and $n_i$ be the count of non-bumper pixels in bin $i$. In the standard implementation of the algorithm, we would mark a bin as a bumper if $c_i > n_i$.

Note the use of the strict inequality. This is to cover the case in which $c_i = n_i = 0$ since we do not want to mark a bin as a bumper if $c_i = 0$.

We generalize this algorithm slightly by introducing a bias, $b$, and mark a bin as a bumper if $c_i > bn_i$. Thus, we can tune the classification to avoid either false positives or false negatives. If we set $b > 1$, this marks less bins as bumpers and thus reduces the false positive rate. If we set $b < 1$, this marks more bins as bumpers and reduces the false negative rate. In practice, we prefer to reduce the number of false positives since we have no trouble identifying the bumpers but tend to have a high number of false positives which make it more difficult to isolate the bumpers. We have good results setting a bias of $b = 1.5$.

To further increase the accuracy of the binarization, we introduce constraints on the properties of the regions. We perform region labeling and look at the eccentricity, length and orientation of the regions. Since the bumpers are long and slender, we expect a high eccentricity. We thus set a threshold and reject all regions below this threshold. In practice, we choose a threshold of 0.98. Furthermore, based on inspection of the images, we know that there is a specific range of values for their orientation, so we impose this additional constraint on the regions. We reject orientations outside of $[75^\circ, 95^\circ]$ and $[-95^\circ, -75^\circ]$ where the angle is taken from horizontal. Finally, we expect the regions to be quite long, so we look at the length of these regions and reject those below a threshold.

To be more robust, we find the longest region and reject regions which are less than a given fraction of this length.
We have had good performance using a threshold of \( \frac{1}{2} \) the maximum length.

2) Angle Calculation: Once we have binarized the image and selected out the bumpers, we take the orientation of each region. We let \( \phi_1 \) and \( \phi_2 \) be the angle of each region with respect to horizontal. To determine the angle of the camera, we require \( \alpha \), which is the angle from vertical. In theory, we should have \( |\phi_1| = |\phi_2| \), but in practice, we note a slight difference in these angles. Thus, we will average these angles together in determining \( \alpha \) and use:

\[
\alpha = 90^\circ - \frac{|\phi_1| - |\phi_2|}{2}
\]

We also require knowledge of \( \beta = \frac{2d}{w} \), where \( d \) is the camera height, and \( w \) is the width of the ice sheet. We can find an estimate of \( \beta \) using two frames from the video at different locations on the ice using Equation 21. Since this value is of high importance, we want to estimate it with a high degree of accuracy. Thus, in calculating \( \beta \), we do not use the estimated angles of the bumpers from our program, but rather visually inspect the images, find a large number of points along the bumpers and perform linear regression to estimate their angle. We can find \( \lambda \) if we choose our images carefully so that we have a known ice feature in the center of the image and use the distance of these features from the button. Once we have an estimate of \( \beta \), this allows us to calculate the angle that the camera is at while looking at the image using Equation 14.

We also need to find an estimate of the field of view, \( \gamma_y \). Letting \( w_p \) and \( h_p \) be the width and height of the frame in pixels, it is clear that:

\[
\gamma_y = \tan^{-1}\left(\frac{h_p}{w_p} \tan \gamma_x\right)
\]

Thus, if we can find \( \gamma_x \), we can easily compute \( \gamma_y \). To estimate \( \gamma_x \), we first estimate the \( x \) coordinates of the endpoints of the bumpers. We compute the centroid of each bumper detected in the binarization, which we denote as \((c_x, c_y)\), then find the endpoints as:

\[
\begin{align*}
x_{i, \text{top}} &= (c_y - 1) \tan \phi_i \\
x_{i, \text{bot}} &= (c_y - h_p) \tan \phi_i
\end{align*}
\]

We use the middle of these points to determine the width, in pixels of the sheet of ice, \( w_s \), as:

\[
w_s = \frac{|x_{1, \text{top}} - x_{2, \text{top}}| + |x_{1, \text{bot}} - x_{2, \text{bot}}|}{2}
\]

Our field of view estimate is then:

\[
\gamma_x = \sin^{-1}\left(\cos \theta \frac{h_p}{\beta w_s}\right)
\]

This allows us to use Equation 25 to get the ratio \( s = \frac{t}{t_0} \).

3) Image Warp: At this point, we know the scale of the image as well as the orientation of bumpers and their endpoints. We want to warp this image to an image with a width of \( w_p \) and a height appropriate to the scale we computed, \( s \). Thus, we want a new height of \( sh \frac{w_p}{w_s} \). We can easily compute a projective transformation that maps the endpoints of the bumpers to the corners of an image with the desired size. We then simply apply this transformation to the image and crop appropriately to get the rectified image.

B. Frame Tracking

This component is responsible for determining which portion of the ice the camera is looking at. To do this, we look for known features of the ice in the current frame. We take 5 distinct features from the ice, including logos, and find their location on the ice sheet. We can then look for the best match against these templates to determine which portion of the ice we are looking at.

1) Matched Filter: We perform all processing on the rectified images from the previous stage. Since we normalize the scale of the images in rectification, the logos on the ice are approximately the same size in all frames. This makes matched filtering a good candidate for finding the ice features. This is implemented as a convolution of a 180° rotation of each image with the frame. In order to reduce the computational complexity, we first scale the images and templates by a factor of \( \frac{1}{4} \).

2) Scale by Distance: Using only the matched filter can cause some errors in identifying ice features. To alleviate this, we need a way of rejecting false matches. We do this by using the previous location of ice features. We assume that the features do not move much from frame to frame and use the previous location of the features to predict where they should be located in this frame. We thus scale down the response for locations which are further from the expected position of the filter. If we let \( f[x, y] \) be the image (with mean removed) and \( t_i[x, y] \) be the \( i \)th template (with mean removed) and \( y_{est} \) be the estimated location of the \( i \)th feature, we compute the response:

\[
c_i[x, y] = \frac{t_i[-x, -y] \ast f[x, y]}{||t_i||^2 \epsilon_2(1 + (y - y_{est})^2)}
\]

Thus, we weight locations which are far from their expected position less. We do not care about the expected \( x \) location since the logos are repeated horizontally in the image so we
have multiple possible \( x \) positions. In practice, we find that using only the \( y \) position suffices. Note that we divide by \( \|t_i\|_2^2 \) so that we do not weight images with a higher norm more highly than those with smaller means and rather look at the percentage of pixels that are good matches.

3) Best Response: Once we have the responses, \( c_i \), we can proceed to find the best match. Specifically, we let:

\[
(x_i, y_i) = \arg \max_{x,y} c_i[x, y]
\]

\[
i^* = \arg \max_{i} c_i(x_i, y_i)
\]

Thus, \( i^* \) is the index of the best matching ice feature and \( (x_{i^*}, y_{i^*}) \) is the location of its best match. We output this index and location.

C. Stone Tracking

For stone tracking, we need only track the position of the stone within the current frame. We use a simple two part method of doing this: we first use a mean shift algorithm to get a rough estimate of the location then a circular Hough transform to get a more refined position. Since the mean shift method is robust against occlusion, this maintains the stone’s position even when it is obstructed by the sweepers [2], [3].

1) Mean Shift: We use the algorithm described in [2]. We let \( \{x_i\}_{i=1}^n \) be the pixel locations of a circular region centered around the stone. In our implementation, we use a constant radius of 25 pixels for this region. We choose this location by having the user identify the initial location of the stone in the video. Let \( b(x) \) bin the pixels based on color into \( m \) bins of size \( 2^k \). In practice, we use \( k = 3 \). First we compute an estimate of the pmf of our stone in the initial frame using:

\[
\hat{q}_u = C \sum_{i=1}^n k(\|x_i\|^2)\delta[b(x_i) - u]
\]

where \( C \) is chosen so that \( \hat{q}_u \) sums to unity and \( k(x) = 1/\sqrt{2\pi}e^{-x^2/2} \) is the Gaussian kernel. Then consider any position in the image, \( y_0 \), and the set of pixel locations in a circular region centered at \( y_0 \), \( \{x'_i(y_0)\}_{i=1}^n \). Then in the target image, we can take the pmf of this region, using:

\[
\hat{p}_u(y_0) = C \sum_{i=1}^n k(\|x'_i(y_0)\|^2)\delta[b(x'_i(y_0)) - u]
\]

Again choosing \( C \) to normalize the pmf. Our goal is to choose \( y_0 \) to maximize the Bhattacharyya coefficient:

\[
\rho(y_0) = \sum_{u=1}^m \sqrt{\hat{p}_u(y_0)\hat{q}_u}
\]

We utilize the iterative algorithm described in [2] to find the \( y_0 \) that maximizes \( \rho(y_0) \) and output this as our estimate of the stone location.

2) Hough Transform: We now have an estimate of where the stone is located, but this estimate is subject to noise. We can refine this using a circular Hough transform. Since the image is rectified, the circular top of the stone remains circular throughout all frames which makes the circular Hough transform very effective. We slice out a region around the estimate from the mean shift algorithm and perform a circular Hough transform to find the stone. If we let \( r \) be the radius of the stone found from the previous location, we use a slice which is a square of side length \( 6r \) and look for circles with radius in the range \( [0.75r, 1.75r] \). We also set a threshold for the distance between the circles found with the Hough transform and the estimate from mean shift. If the circles are within that threshold we keep them, otherwise we throw them away. If there are circles left, we set the estimated location to the location of the circle in the Hough transform and use the radius of this circle. Otherwise, we keep the radius from the previous location and use the location estimate from mean shift. In practice, we have found a distance threshold of 40 works well.

D. Localization

At this point, we know the stone’s location in the frame as well as the location of a feature point in the frame. Since we know the location of the the features on the frame, this allows us to find the exact portion of the ice that we are looking at. Furthermore, we know the location of the stone within the current frame. Since we have modified the scales during rectification so that they remain constant across the different frames, this allows us to easily translate the the stone’s location in the frame to a location on the curling sheet.

IV. Results

We have applied the curling shot tracking program to several videos and observed the results. Included is a sample frame from these videos to give an idea of the performance. Visually, we can see that the algorithm is quite effective at finding the ice features and stone. We are able to combine these results together into a video to show how it performs. Inspection of these videos demonstrates that the algorithm smoothly tracks the current camera view and plots accurate positions for the stone locations.

V. Future Work

This project provides a means of tracking curling stone positions on the ice sheet. This capability offers many interesting directions for future work. The most basic of these directions is to offer shot prediction. The output of our program provides a list of the position of the stone in each frame. Given this data, one could apply a curve fitting program to predict what path the stone will follow. We could also utilize velocity information to predict when the stone will stop. To estimate the stopping time would require analysis of many different curling shots to help figure out a relation between velocity and distance traveled.

The above analysis assumes that there is no sweeping of the stones so that they all behave identically. While it is possible to find enough shots in which there is no sweeping to perform such an analysis, this somewhat limits the scope of the program. We could generalize this kind of analysis by tracking the position of the brooms and inferring if there is sweeping or not. Then, given an accurate prediction program as described above, we could determine how sweeping affects
the path of the stone. Analysis of many shots in this fashion should offer a fairly accurate quantification of the effect that sweeping has on the path and distance of a stone.

There are many applications of a program which can predict the effect that sweeping has on a stone. It can be used by live broadcasts to enhance commentator analysis. It could be used by curlers wishing to advance their game by helping them to better understand the effect that their sweeping has and to give them a realistic picture of how much they can hope change their shots using sweeping. Finally, it could be used as a tool for skips to help them decide when to call for sweeping of shots. This kind of analysis could give them a picture of what the shot will look like if their teams sweeps and what it would look like if they do not. This would let them decide whether they want to call for sweeping or not.

REFERENCES

