Improving OpenCV's CAMShift Algorithm with Curve Fitting

Andreas Brauchli
Department of Information and Computer Science
University of Hawai'i at Mānoa
anreasb@hawaii.edu

Paul E. Soulier
Department of Information and Computer Science
University of Hawai'i at Mānoa
psoulier@hawaii.edu

Abstract— **Object tracking in a computer vision system is concerned with identifying one or more specific objects through a series of sequential and related images. This paper investigates the application of curve fitting and trajectory prediction to existing motion tracking algorithms, such as the CAMShift filter, with the objective of improving accuracy, performance, and extracting useful contextual information related to the path of the tracked object. We present an improvement to OpenCV's CAMShift algorithm by applying curve fitting techniques to an object's projected trajectory. Our research showed tangible improvements were achievable with respect to tracking reliability in specific scenarios with the primary challenge being accurate and reliable data extrapolation; interpolation methods do not provide suitable data outside the data set. We found that traditional curve fitting performs poorly for data extrapolation and propose an alternative approach using function linearization of multiple path templates that may offer more promising results. The methods and theories described herein are equally applicable to other object tracking methodologies and are not limited to the CAMShift algorithm. Furthermore, the concepts and techniques developed for trajectory prediction have other practical application to active vision systems such as video-based servoing where trajectory information can be used for real-world interaction. The data obtained indicates strong evidence that the use of contextual information of an object's path can be successfully incorporated into analytically based tracking techniques to improve the overall capability of the tracking system.**

Keywords—computer vision; object tracking; motion tracking; curve fitting; path prediction; trajectory prediction; CAMShift; Mean Shift

I. INTRODUCTION

As humans, we have the remarkable ability not only to follow an object, but to instinctively anticipate its future location. Even when it is partially obscured or completely occluded for a period of time, we have the uncanny aptitude to effortlessly predict where an object of interest will be at a later point in time. The value of this skill as it pertains to people and animals is self-evident and is equally relevant to computer vision systems for many of the same reasons.

In many of the existing practical approaches to object tracking, identification of an object is largely based on finding patterns within an image that have the visual characteristics of the target object. These methods, however, are unable to disambiguate between multiple objects with similar visual attributes in various scenarios. Furthermore, significant changes in lighting or environment that sufficiently alter the visual qualities of the object can cause practical methodologies to lose track of the intended target[1] resulting in inaccurate computations.

Object tracking in computer vision systems can be used for a variety of purposes such as surveillance and monitoring or video-based servoing. In addition to improving tracking accuracy and performance, the contextual information generated by path prediction can also be used in the implementation of the aforementioned applications. For example, as software-based automation of vehicles become increasingly prevalent, mechanisms that employ path prediction to detect potential collisions could easily be envisioned. Additionally, the use of extrapolated path prediction information in an active vision systems could conceivably be useful in providing the ability for a system to accurately change camera position to keep an object within the sensor's field of view.

Combining contextual information such as trajectory prediction with techniques like the CAMShift filter [2] has the potential to produce more capable object tracking systems. Our goal is to show that path prediction can be used to create tangible improvements to the capabilities of existing algorithms. This work and associated results are based on the OpenCV [7] implementation of the CAMShift algorithm. Additionally, we create a generalized approach to extrapolating path information that can be used with any object tracking algorithm for a variety of visual object tracking applications.

The article is organized as follows: after this introduction Section II provides relevant background information and related works. Section III presents the method developed for extrapolating path prediction information. Section IV presents the results of our work. Section V describes areas of future work related to the methodologies present herein. Section VI provides the specific contributions made by each of the authors. Finally, section VII concludes the paper.

II. RELATED WORK

Our algorithm improves upon CAMShift [2], which in turn is an extension to MeanShift [3] object tracking algorithm. MeanShift is an iterative approach for object tracking that

[1] Particularly CAMShift may not notice when an object was lost and simply start tracking the

works on an object-probability image[2] and a search window and shifts the search window until the window's overall summed value is maximized and thus best covers the tracked object. The object-probability image is created from a histogram of the object to be tracked such that the image pixel's value is the percentage of the matching histogram bin relative to all other bins this pixel belongs to. CAMShift improves on MeanShift by additionally re-scaling and rotating the search window instead of simply moving it. CAMShift stands for Continuously Adaptive Mean Shift.

To the best of our knowledge, there have been no publications on applying curve fitting to improve object tracking in general.

III. Path Trajectory Prediction Methodology

This section describes the problems encountered when using extrapolation from curve fitting techniques based on numerical analysis and our proposed method of path prediction. Data extrapolation is the process of extending a function beyond the scope of available data. In order to extrapolate data, a curve is first fit to given data points to interpolate the values. The curve is then evaluated at the next frame point outside the scope. Traditional curve fitting approaches are ill-suited to the task of extrapolation.

A. Coordinate Frame Separation

Real-world object tracking scenarios follow complex, often erratic paths that cannot be easily represented by a mathematical model. This is due to a shifting camera viewpoint or field of view, different lighting that impacts finding the tracked object's center, obstacles, *etc.* Additionally, when the path history of an object is analyzed, it will frequently produce a path that violates function uniqueness when fitting a mathematically modeled curve to the path. To illustrate this issue, consider the path depicted in Figure 1a.

The manner in which the object moves would be impossible to construct a mathematical model that could represent this path. To address this issue, the object's path is decomposed into the constituent x-coordinate and y-coordinate components, each with respect to time - the current sequence frame. By performing this decomposition, two distinct data sets are obtained, each of which is guaranteed to abide by the definition of a function. This allows modeling the data with curve fitting and interpolation methods. Figure 1b shows the path broken into the respective x and y coordinate components. Note that while the y-component varies linearly with time, the x-component follows the path of a quadratic polynomial.

B. The Curve Fitting Problem

Different curve fitting algorithms were used to evaluate their ability to effectively extrapolate data. The methods investigated were cubic splines, smoothing splines, least square polynomial regression and finally weighted least squares polynomial regression. All these methods share the common feature of accurately interpolating data points but the extrapolated trajectory error exponentially grows after the last data point - especially for non-linear data points. Our final implementation uses weighted least squares fitting with

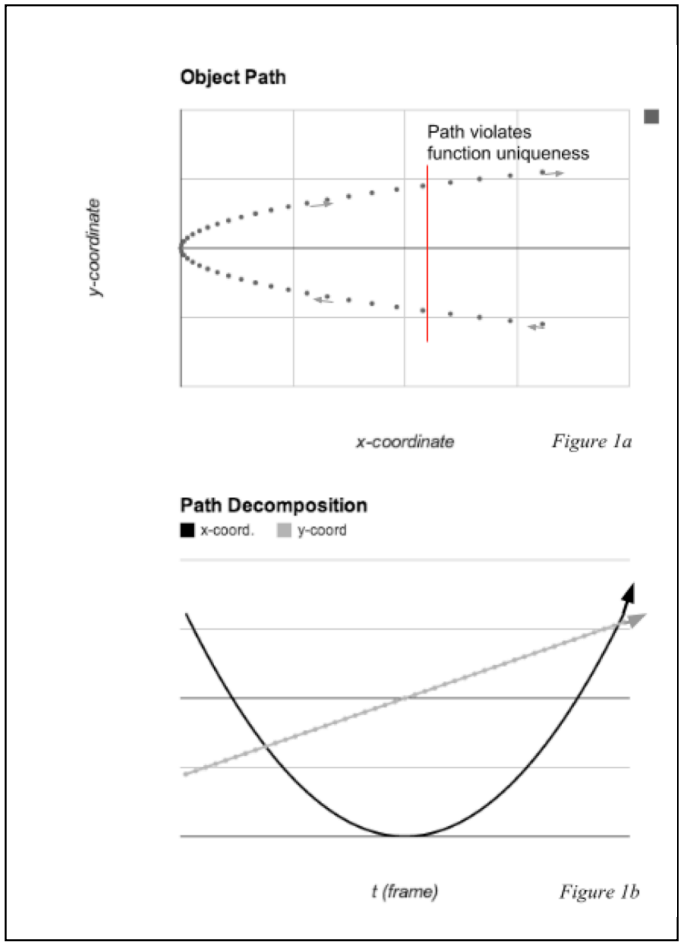

Fig. 1. Path decomposition

polynomials of quadratic degree due to the lower exponential factor.

Cubic splines match every data point is perfectly and thus creates an unnecessary curvy spline due to small shifts in the object's determined center point. To counter this effect, smoothing splines were considered. Instead of simply passing through every data point, a path smoothness constraint is added. Smoothing splines generally produce a better result than the cubic splines but the smoothness constraint results in a purely linear prediction. As with other approaches, splines are generally not concerned with points outside the given data set, thus making them a poor fit for use in path prediction.

Lagrange interpolation with a fixed degree polynomial of cubic order was first considered but never implemented due to Runge's phenomenon which already yields large swings between the data points. Also, it is considered too noise sensitive as it passes exactly through each data point.

Least squares fitting fits a fixed size polynomial curve to the data set by minimizing the overall deviation at every point. Generally, the higher the polynomial degree, the worse the prediction becomes because of the exponential nature of the curve. Linear and quadratic curves yield the best results. Additionally, better extrapolation results are also obtained when the system is overconstrained as this eliminates noise.

[2] The higher the pixel's value, the more probable it is that the object is located there.

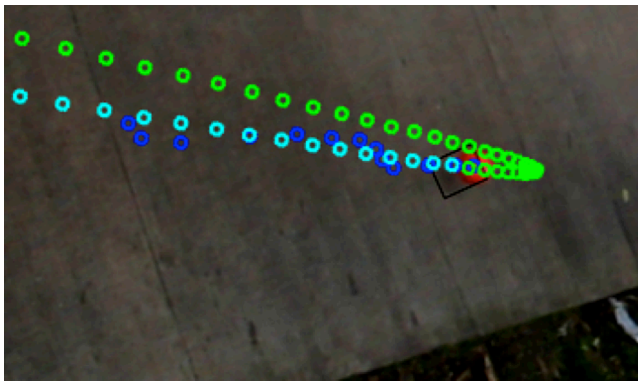

Fig. 2. Path history and computed trajectory of tracked object.

We thus reverted to polynomials of quadratic degree after experimenting with higher order polynomials. Figure 2 shows an orange ball tracked by the black-outlined search window. Its locations in previous frames is shown by dark blue circles. The fitted quadratic least squares path can be seen in the light blue circles and the predicted locations appears in green. We see that the curve finds its minimum after seven frames, after which predictions become exponentially worse.

Figure 2 also shows the result of generating a predicted curve using the described methods. In the image, the pink ball is the object being tracked. The dark blue circles represent the actual path traveled. The green and light blue circles show the predicted path. As can be seen, the prediction is accurate only for a very short distance before shifting exponentially in the wrong direction. This behavior is the result of the curve fitting algorithms only fitting a curve to the known data. The curve that is fit, however, is not representative of the path beyond the known data. The reason for this can be seen in Figure 3. The red and blue lines are plots of actual path data. The fit curves use data to the left of the gray dashed line and are represented by the green and yellow paths. Two observations can be made. First, the curve generation is extremely accurate for the data that was used to produce the polynomial approximation of the known data. Second, the polynomial is only valid within the given data set. Beyon the given data set (everything to the right of the gray line), the predicted curve simply follows the polynomial – which is entirely wrong with respect to extrapolating the future trajectory of the ball. Rather than the ball coming to a complete stop (this can be seen as the flattening out of the actual path in the red and blue lines), the prediction indicates the ball will infinitely accelerate.

To account for suboptimal pruning, described in the following subsection, weighting more recent data points achieves a better prediction at the cost of increasing noise when the last points are particularly noisy. We evaluated different weighting strategies of which a weight increase of 20% for every datapoint. We did not evaluate weighting points after a period of occlusion. We do, however, suggest to increase the weighting by 20% for every missed frame as the trajectory may have changed during the transition through the occlusion.

C. *History Pruning*

As paths can change radically over time due to external influences, such as the object hitting an obstacle and bouncing off, the curve is unable to represent those complex paths. The whole path is thus segmented into parts following a more natural motion. As our paths are separated into the two (t, x) and (t, y) components, the history is pruned individually for each of those components. It is generally desirable to prune history only when necessary as not having any history will lead to a constant extrapolation value whereas even a short history provides significantly better extrapolation in the pruned dimension. We present two approaches to solving the pruning problem in this section: The first approach uses the mathematical understanding of the changing path and the second uses a practical error-based approach. Our implementation uses the practical approach for the reasons stated hereafter.

The mathematical approach is based on the understanding that complex paths exceeding the polynomial's degree cannot be represented exactly - and mostly not even closely - by the degree-limited polynomial. We thus need to segment these paths into the longest possible representable path. The path length influences how accurate the prediction will be with longer paths generally yielding more precise extrapolation. We find that separating path segments at their point of inflection yields the smoothest results as the curve is linear at the point of inflection, thus working well with a small point history, and

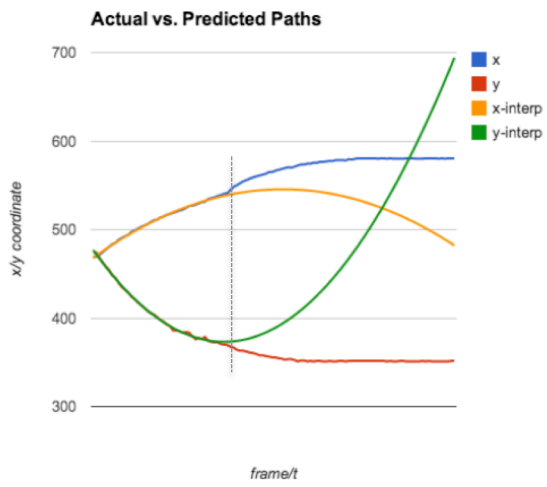

Fig. 3. Curve fitting results applied to path history.

there will be a certain point history available once the curve reaches the next local minimum or maximum. The mathematical condition for an inflection point is that the second derivative is zero and the highest non-zero coefficient is of an odd power. The primary issue with this approach is accurately extracting the inflection points from discrete path history data. Small fluctuations that occur from a changing center of mass in the tracked object often results in many relatively insignificant inflection points. This noise would need to be filtered out for this method to be useful. A possible approach that has yet to be investigated is to minimize the impact of the noise by using a least squares polynomial

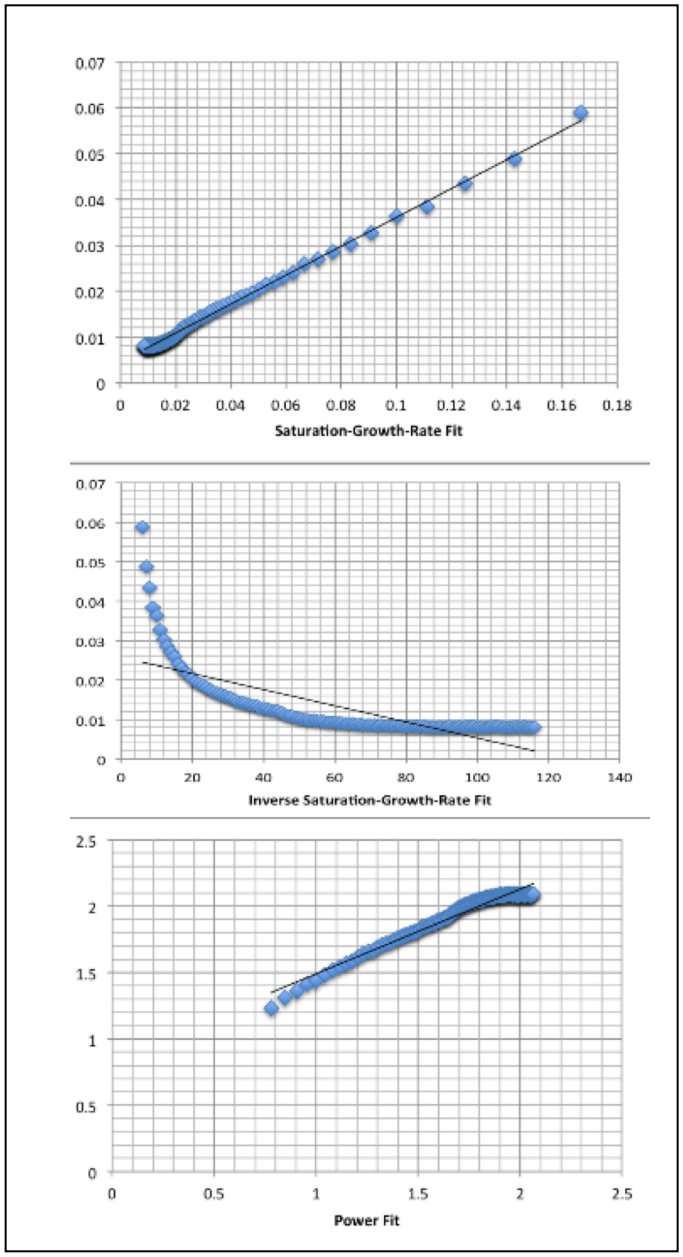

Fig. 4. Template linearization plots of sample data.

regression on the history which generates a curve that does not necessarily pass through each data point. Inflection points could theoretically be obtained from the fitted curve.

The practical approach simply examines the path data for a local minimum or maximum values and discard older data from the those points. This approach also discards data with very small fluctuations; again due to a shift of the center of mass of the tracked object. To account for this, we added an additional constraints to the path history discard mechanism to only discard if the difference exceeds a certain configurable threshold.

We find that while an inflection point approach, while theoretically ideal and may prune the history at optimal places, is unnecessarily complicated and adds additional computational effort that is not required. The practical approach was implemented as it was easier to implement and more efficient; it only needs two differences and a multiplication.

D. Linearized Path Templates

In real-world scenarios, the path an object follows is often complex and attempting to develop a single model that can accurately describe and meaningfully predict any path is extremely challenging. This section suggests an alternative approach based on fitting multiple possible path templates to the current path history data with the goal of finding an equation that best resembles the historical data as well as offering a reliable estimate on the object's future locations.

As stated above, object paths are generally quite varied in nature. However, when subjectively analyzing coordinate separated data, common traits of motion become evident. The path decomposition provides motion in a single direction. Since these motions are generally created by physical objects, they tend to follow nature's laws for motion. As a result, objects can be categorized into the following states of motion: not moving, moving with a constant speed, accelerating, or asymptotically decelerating. While obviously these categories cannot account for all motion, they are a reasonable approximation for most. Given these categories, a method is needed to determine how accurately each template represents the current path history, how likely it is to represent future position, and how it compares relative to the other templates. To accomplish this, equations are selected that have similar characteristics to each of the observed behaviors. The path templates used are based on the following general equations:

Linear Equation:

$$y = ax + b \tag{1}$$

Saturation-Growth-Rate Equation:

$$y = a\frac{x}{b+x} \tag{2}$$

$$\frac{1}{y} = \frac{b}{ax} + \frac{1}{a} \tag{3}$$

Inverse Saturation-Growth-Rate Equation:

$$y = \frac{b}{ax} + \frac{1}{a} \tag{4}$$

$$\frac{x}{y} = ax + b \tag{5}$$

Power Equation:

$$y = ax^b \tag{6}$$

$$\log(y) = b\,log(x) + \log(a) \tag{7}$$

Equations 1, 2, 4, and 6 are the path templates that represent common trajectories. To evaluate how closely a

template resembles the actual data and to allow a comparison to other non-linear paths, linearization of non-linear functions is used [1]. Equations 3, 5, and 7 are the linearizations (equation 1 is already linear and does not need an inversion). The path history is linearized by applying the same transformation to each data point. Since each data point can be expressed in a transformed linear form, linear regression can be used to determine how closely the path history data resembles a line. Transformed data points that closely match a line also closely resemble the non-linear, untransformed equation. All of the path template equations are expressed in a linear form. As a result, a comparison can be made between the relative accuracy of the fit to each equation providing a quantitative measurement regarding which template provides the most representative fit *e.g.* by using Linear Least Squares and minimizing the standard deviation.

IV. RESULTS

After the practical evaluation of all implementations discussed in (III), we find that combining curve fitting together with CAMShift only works well in a small number of cases such as near linear motion (with or without acceleration) or objects passing through partial occlusion. It does however not work for full occlusion because CAMShift adapts itself to track the background behind the last known position. To accommodate for this, we would need to examine changes in the histogram of the currently tracked window and not initiate CAMShift in frames where the object is unlikely to occur. Running CAMShift in a frame where the object is not visible is likely to change the center of mass in an unrecoverable way.

A. Test Cases

To validate our results, we are considering test cases where a moving object is tracked from a static camera position. We considered the following test cases to compare and validate our results against the unmodified CAMShift algorithm:

1. Straight moving object
2. Accelerating object
3. Object hitting and bouncing off an obstacle
4. Multiple similar objects in field of view
5. Two similar object in field of view crossing in close proximity to each other
6. Object passing through partial occlusion
7. Object passing through full occlusion

In our test cases, the tracked object is mostly a bright pink tennis ball. We chose this object because it is well distinguishable from our test backgrounds. Our backgrounds are chosen to be natural but with minimal noise.

The first case is covered by the tennis ball rolling in a straight direction on a concrete floor. The ball slowly decelerates during its mostly straight path. In our second case, we observe a falling tennis ball over three stories. The third case is similar to the first but the ball hits the wall and bounces off. The next case is again similar to the first but this time with two balls rolling parallel to each other. Case five is two tennis balls crossing their paths closely together. This case is particularly interesting because CAMShift erroneously switches from the tracked ball to the other as their paths cross. Test case six is a ball falling that is partially occluded by branches. Finally, case seven is a cardboard box with an orange sticky-note that is slid in a straight path passing behind a paper bag. Orange does not occur in the rest of the scene.

From all the above test cases, trajectory prediction offered the most noticeable improvement to test case 5. As mentioned, without trajectory prediction, the CAMShift algorithm ultimately tracks the wrong object as the two similar objects pass in close proximity to each other. By specifying where the CAMShift algorithm should begin its search based on the predicted path of the ball, we were able to influence the algorithm enough to reliably track the correct object.

B. Solution Analysis

To evaluate how well each prediction method is working, we developed a statistics class. The class instance is fed with the actual data point on every frame as well as the prediction for the next n=40 points. When enough data points are available (n), the prediction is evaluated against the actual value. The statistics object then outputs the average and deviation for each level of prediction (1..n). Since the prediction is a 2D point, the average and deviation is measured over the difference of $\sqrt{x^2 + y^2}$ between the predicted and actual point. This allowed for efficient analysis and debugging the various approaches that were attempted.

C. Linearized Path Template Results

While the final project did not fully implement this technique, sufficient data was obtained to demonstrate its potential for further study. To validate the effectiveness of the Linearized Path Template method, a data set containing a ball being tracked as it rolled down a sidewalk was extracted and analyzed for the x-coordinate component. This data can be seen as the yellow line in Figure 3 and shows the ball is moving to the right of the screen (increasing x-value), decelerating, and finally stopping. The data shown in Figure 4 is a scatter plot of the x-coordinate relative to time for the linear inversions of the sample data set.

As can be seen, the plot for the Saturation-Growth-Rate equation shows a strong correlation to a linear function. The scatter plots in Figure 4 show the actual data with linearization applied. The plots for the Inverse-Saturation-Growth-Rate and Power Equation do not exhibit this same linear relation. As expected, based on the known trajectory, the Saturation-Growth-Rate template shows the most accurate fit. Visually, this is easy to see. In order for vision system to deduce this result, a quantitative method must be used. The use of linear regression enables the *coefficient of determination* to be computed and represents accuracy of the fit. A value of 1 implies a perfect fit. Table I summarizes the computed coefficient of determination for each of the path templates from the sampled data. As can be seen, the Saturation-Growth-Rate is not only the best choice of the four templates, but also is

very close to the ideal value of 1 indicating that it is highly probable the trajectory will have a similar shape.

TABLE I. LINEARIZED PATH TEMPLATE COEFFICIENT OF DETERMINATION RESULTS

Linearized Path Template	*Coefficient of Determination*
Linear	0.866160
Saturation-Growth-Fit	0.996826
Inverse Saturation-Growth-Fit	0.540392
Power	0.962531

These results show a strong indication that this methodology will produce an accurate model of trajectory and can provide a mechanism to easily determine which path template is best. Also of value is this technique can also determine if no template produces a good fit by selecting a threshold for the coefficient of determination.

V. FUTURE WORK

Our algorithm is implemented atop CAMShift but is held completely apart such that CAMShift is not modified by our implementation. A working final product would integrate motion prediction into the motion tracking algorithm. We suggest a high-level object tracking algorithm that is composable from different probabilistic methods such as optical flow, motion tracking, physics *etc.* that could each be independently enabled to best fit the tracked object.

We would like to further pursue certain aspects of the Linear Path Template method by completing a fully functional prototype and extending the template model beyond linearized functions. For example, cyclic motion (such as a ball spinning at the end of a rope) could be represented as a sinusoidal path when the x and y components are considered separately.

VI. CONTRIBUTIONS BY THE AUTHORS

The project was performed in a collaborative manner and most of the work was done jointly. We discussed solutions before implementing them and received feedback afterwards. Efforts that can be categorized include refactoring OpenCV's *camshiftdemo* and the Least Squares solver. The working C++ source code is available from [9]. Demo sequences can be obtained from [10].

The demo refactoring was designed and written by Paul Soulier. It dissects the base demo into the multiple inheritance framework that abstracts CAMshift as one possible object tracker. This will make it easier to plug in a different algorithm for object tracking. Included in the refactoring is an improved command line argument parser.

The Least Squares solver was written by Andreas Brauchli. It is designed as a C++ class template that works on arbitrary numeric types. Also a class template argument is the maximal polynomial degree that the resulting function will have. When not enough data is available, the polynomial degree is reduced. The solver also evaluates the resulting function at any point within (interpolation) or outside of the data set (extrapolation).

VII. CONCLUSIONS

Object tracking is a highly challenging, multi-faceted problem with many real-world applications. Tracking an object based solely on the visual characteristics of the target is insufficient in various scenarios when attempting to achieve consistently reliable results. This paper has presented trajectory prediction as an incremental improvement to motion tracking that is generally applicable to most existing object tracking algorithms. Object tracking is a challenging task consisting of multiple sub-task and motion tracking is one of these tasks that can help in achieving a more robust tracking solution. We have found that curve fitting only works well when the object is following a linear path of motion, but does not produce accurate results when applied to more complex paths. Preliminary data shows that Linearized Path Templates would likely offer significant improvements over a simple curve fitting approaches such as polynomial regression or splines.

It is our hope that the Linearized Path Template method will be further refined by subsequent research and that the proposed concepts can be incorporated into future object tracking methodologies. While we would have liked to make more progress such as working prediction after full occlusion, we are glad to have learned much on the intrinsics of curve fitting as it pertains to motion tracking.

REFERENCES

[1] Chapra, S. C., & Canale, R. P. (1998). Numerical methods for engineers.

[2] G.R. Bradski. Computer vision face tracking as a component of a perceptual user interface. In Workshop on Applications of Computer Vision, pages 214-219, Princeton, NJ, Oct. 1998.

[3] K. Fukunaga and L. Hostetler. "The estimation of the gradient of a density function, with applications in pattern recognition." *Information Theory, IEEE Transactions on* 21.1 (1975): 32-40.

[4] Andreopoulos, A., & Tsotsos, J. K. (2013). 50 Years of Object Recognition: Directions Forward. *Computer Vision and Image Understanding*.

[5] Wikipedia, (12/2013). "Spline" [Online]. Available: http://en.wikipedia.org/wiki/Spline_%28mathematics%29.

[6] Wikipedia, (12/2013). "Smoothing Spline" [Online]. Available: http://en.wikipedia.org/wiki/Smoothing_spline.

[7] OpenCV, (12/2013). "OpenCV Developer Documentation" [Online]. Availalble: http://opencv.org.

[8] Wikipedia, (12/2013). "Point of Inflection" [Online]. Available: http://en.wikipedia.org/wiki/Inflection_point.

[9] A. Brauchli, P. Soulier, (12/2013). "Code Repository" [Online]. Available: https://github.com/abrauchli/camshift

[10] A. Brauchli, P. Soulier, (12/2013). "Google drive demo video folder" [Online]. Available: https://drive.google.com/a/hawaii.edu/folderview?id=0B3Woloi8oDioMXRXMzNMbkhIM2s&usp=sharing