Optical Tracking Using Projective Invariant Marker Pattern Properties

Robert van Liere, Jurriaan D. Mulder
Department of Information Systems
Center for Mathematics and Computer Science
Amsterdam, the Netherlands
{robertl,mullie}@cwi.nl

Abstract

In this paper, we describe a new optical tracker algorithm for the tracking of interaction devices in virtual and augmented reality. The tracker uses invariant properties of marker patterns to efficiently identify and reconstruct the pose of these interaction devices. Since invariant properties are sensitive to noise in the 2D marker positions, an off-line training session is used to determine deviations in these properties. These deviations are taken into account when searching for the patterns once the tracker is used.

1 Introduction

Optical tracking for head-tracking and interaction devices in Virtual and Augmented Reality can provide a valuable alternative over other tracking methods like magnetic, gyroscopic, and mechanical trackers. Advantages of optical tracking are that it allows for wireless ‘sensors’, it is less susceptible to noise, and it allows for many objects to be tracked simultaneously. To avoid complex and computationally expensive image processing, optical tracking systems often use specific markers that can easily be detected in an image. Several systems use retro-reflective material in combination with infra-red lighting. The image processing algorithm then simply comes down to finding light blobs in an otherwise dark greyscale image. These blobs however, do not contain any information for identification, i.e. in the images it is not known which blob of pixels belongs to which marker.

A common approach in optical tracking of interaction devices is to equip the devices with multiple markers in a known 3D configuration. Marker identification is then accomplished by reconstructing all possible 3D marker positions and searching for the desired 3D marker configuration. This is however, a combinatorially expensive operation. All possible 3D marker positions have to be constructed from the 2D marker positions in the stereo images with the use of stereo corresponding and epipolar geometry. However, since no marker identification is present in the 2D images, stereo corresponding will result in many false matches and therefore many false 3D marker positions. This is especially the case when multiple input devices are used and the 2D images become highly cluttered with markers.

In this paper we describe a method for designing and tracking marker patterns with projective invariant properties. Patterns consist of four collinear or five coplanar markers. The tracking method makes use of the invariant properties of the patterns to search and identify candidate marker patterns in the 2D images. This significantly reduces the search space for the recognition and reconstruction of the 3D marker positions.

Projection invariant properties are sensitive to noise in the 2D marker positions in the image. Therefore, deviations in the properties of the marker patterns are determined in an off-line training session. These deviations are taken into account when searching for the patterns once the tracker is in application mode.

We have applied our optical tracking algorithm for interaction in the Personal Space Station, our near-field VR/AR environment [1]. This environment makes use of graspable input devices to perform direct 3D interaction. Retro-reflective markers are pasted in collinear or coplanar patterns onto the devices to be used. The tracking system is then trained to recognize these patterns after which the devices can be applied for interaction.

The paper is organized as follows. In the next section we discuss related work. In section 3 we review a number of concepts that are fundamental to our tracking method. In section 4 we give a technical description of the steps involved in our tracking method. In section 5 we show how the method performs on a selection of interaction devices. Finally, in section 6 we discuss the pros and cons of the method.

2 Related Work

For years, the field of computer vision has used invariant theory to solve many vision problems [2, 3]. Although much of the work in this area is theoretical, computer vision research has also focused on making invariants robust for practical applications where noisy images are used. Identifying the model best representing the data in a database is usually referred to as the *indexing problem*.

Optical tracking systems using retro-reflective markers attached to an input device have been developed in other VR/AR systems; eg. Dorfmüller [4] and Ribo et al. [5]. Both systems first solve the correspondence problem between points in left and right image of the stereo vision system. Fixed distances between retro-reflective markers in a triangle structure are used to resolve ambiguities that arise after correspondence problem. A best fit method is used on all distances of 3D-points to find the sought triangle. Our approach differs from these approaches in that we use projective invariant properties to search for a pattern. The 3D position of the pattern will be computed only after it is found in both images.

3 Preliminaries

In this section we briefly review four known concepts which are used for searching, detection and identification of invariant marker patterns. The governing concept is the notion of a cross ratio and using this as a metric in a practical system.

3.1 The Cross Ratio Invariant

Given the projections of 4 collinear points (labeled as A, B, C, D), the cross ratio is the real number defined by:

$$\frac{|AC| \;/\; |BC|}{|AD| \;/\; |BD|}$$

where $|AC|$ is the length of the line segment from point A to C. A remarkable property of the cross ratio is that it is invariant under perspective projections, as is illustrated in Figure 1. Although the relative distances between the projected points on the line changes, the cross ratio remains constant.

- Permutation of point labellings

  The computation of the cross ratio depends on the order of the four points. They can be considered in $4! = 24$ different orderings.

  Instead of computing the cross ratio of each possible ordering, we use *projective and permutation* p^2*-invariants*, introduced by Meer et al.[6]. p^2-Invariants obtain representations of point sets that are insensitive to projective transformations and permutations of the labeling of the set. Thus, independent of how the labels of the four points are chosen, when the cross ratio is used as argument in the p^2-invariant, the same value is obtained. We denote this invariant as **J** (λ), in which λ is the cross ratio, and **J** is the projective and permutation invariant function.

  The appendix briefly discusses the details of the projective/permutation p^2-invariant.

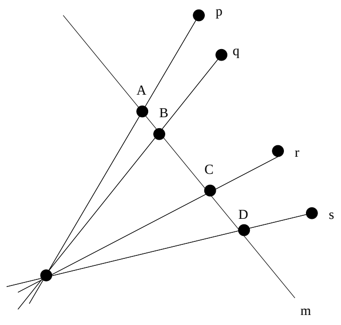


Figure 1. The cross ratio of four collinear points is projective invariant. No matter where line M is placed, the cross ratio of their projections remains constant.

- Sensitivity

  The cross ratio is very sensitive to noise; i.e. to positional errors in the locations of the points. Small errors on the locations of the markers in the image can drastically change the value of the invariant. For example, assume (in Figure 1) that the distances between A and B and C and D are 5, 10, 5 respectively. The cross ratio in this case would be $\frac{15/10}{20/15} = 1.125$. If the positions of B and C are perturbed by one, the cross ratio could be $\frac{16/12}{20/16} = 1.067$

  The sources of noise originate from a combination of varying lighting conditions, varying illumination of the retro-reflective markers, (partially) incorrect camera parameters, and to a lesser extent the numerical errors made during computations. The noise sensitivity is transferred to the p^2-invariant.

The five marker coplanar case is somewhat more complex, but also relies on these concepts. In this case, there are two projective invariants, λ_1 and λ_2. The p^2-invariant **J** function is a five dimensional vector.

A detailed discussion on cross ratios and its properties can be found in standard textbooks on computer vision or projective geometry; eg [7, 8].

3.2 Computing Collinearity

Accurately computing the collinearity of three points is central for computing the cross ratio robustly. The following method provides a metric for the collinearity of three points [9].

Assume three homogeneous 2D points x_1, x_2 and x_3, $x_i = (x_1^i, x_2^i, 1)$. To test for collinearity, we define the moment matrix M_{123} as $M_{123} = \sum x_i x_i^t$. The smallest eigenvalue of the moment matrix measures how collinear the three points are; i.e. the linear dependency of their vectors. We denote the smallest eigenvalue of the moment matrix M_{123} as ev_{123}.

3.3 Convex Hull Projections

A projective transformation which exists between two images preserves the convex hull of a coplanar point set. If two five point configurations are to correspond, then the number of points on their projected convex hulls must also correspond. A convex hull of five points may contain 3, 4 or 5 of the points.

Two properties exist on the projected convex hulls. First, the number of points on the hulls must be the same. Second, for points lying on the convex hull, neighborhood relations are preserved.

3.4 Epipolar Geometry

In optical tracking with stereo image pairs, each marker in the left image will have a corresponding point in the right image. Epipolar geometry can be used to search for a corresponding marker in the two images. This is illustrated in Figure 2. A point in the left image generates a line in the right on which its corresponding point must lie. The search for correspondences is thus reduced from a complete image to a line. This epipolar constraint arises because, for image points corresponding to the same 3D point, the image points, 3D point, and optical centers are coplanar.

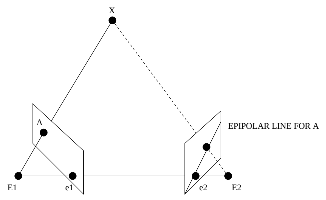


Figure 2. Epipolar geometry. Point X is projected onto point A in the left image. The corresponding point in the right image is on the epipolar line.

4 Method

Our tracking system comprises two different modes: *training* mode and *application* mode. When a new marker pattern is put on an input device, the tracking system is put into training mode and the marker pattern is trained. By waving the input device in the tracking space, the tracker registers the pattern's projective invariant properties and the deviations that occur in those properties due to noise. Furthermore, it registers the pattern's 3D layout, such as the distances between the different markers, including the deviations in these properties due to noise.

All deviations are stored in a pattern database, containing the properties of all trained patterns. A check is made to ensure that the obtained properties are unique, and do not overlap with previously trained patterns. If this might occur, the pattern will have to be changed slightly to obtain unique properties.

Once all patterns are trained, the tracker can be put in application mode and the tracker will search for all trained patterns using the properties and deviations as stored in the database. If a pattern is found, the tracker will return the reconstructed 3D position and orientation of the pattern.

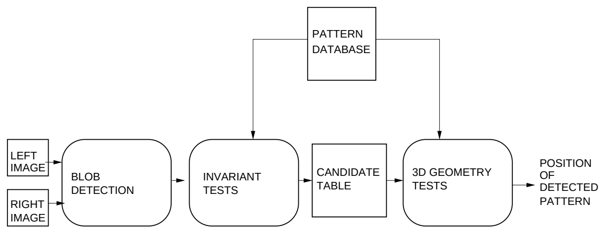


Figure 3. System overview. Two images and the pattern database are used as input. The candidate table stores point set configurations that pass the invariant tests.

The search algorithm consists of two steps, see Figure 3. In the first step, the invariant tests are applied to sets of four or five marker positions as found by the blob detection. This step results in candidate sets of marker points that match the properties of a trained marker pattern. In the second step, 3D geometric constraints are used to select the best candidate. This is realized by pairwise checking each candidate set obtained in the left image with the candidate sets obtained from the right image. The candidate set with the best match to the 3D geometric information of the trained pattern is selected as the final result.

In the next sections we describe each test in more detail.

4.1 2D Tests.

- Collinearity test

In four point patterns, all points must be collinear so the collinearity test is used to quickly remove all point configurations that are not. During training, ev_i^{max}, the maximum smallest eigenvalue of the moment matrix for all three point configurations, is stored in the pattern database. Due to noise ev_i^{max} is not equal to zero, but measures how collinear three points of the pattern in the database are. When searching for a four point pattern i in application mode, all four point configurations whose moment matrix have a smallest eigenvalue less than ev_i^{max} will pass the collinearity test. Any four point configuration which have a smallest eigenvalue larger than ev_i^{max} do not pass the test.

In five point patterns, no three points should be collinear so here the collinearity test is used to quickly remove all point configurations that are collinear. During training, ev_i^{max}, the maximum smallest eigenvalue of the moment matrix for all three point configurations, is stored in the pattern database. During application, ev_i^{max} is used as a lower bound to test for collinearity. Any five point configuration which have a smallest eigenvalue smaller than ev_i^{max} do not pass the test.

- Invariant interval test

  During training, the p^2-invariant of each pattern is computed and the minimum and maximum values of their p^2-invariant computations are stored in the pattern database. The interval for pattern i is denoted as $\left[J_i^{min}, J_i^{max}\right]$. During application, all point configurations whose p^2-invariant are within the interval will pass the test. Point configurations whose p^2-invariant are not within the trained interval do not pass the test.

- Convex hull type test

  During training of a five point pattern, the number of points of the projected convex hull is stored. In application mode, all five point configurations whose convex hull has the same number of points will pass the test. Point configurations that do not have the same number of points on their projected hull will not pass the test.

- Search space reduction with epipolar geometry

  Epipolar geometry can be used to reduce the search space for a corresponding marker pattern. Once candidate configurations have been found in the left image, only those points close to the epipolar lines of the left candidates are taken into consideration when searching in the right image.

  In well calibrated stereo image, points that are in correspondence lie on the same epipolar line. However, when noise is included, a corresponding point need not to lie exactly on the epipolar line. For example, when a high intensity blob in an image represents a point, it can be the case that the 2D point (computed as the center of mass of the high intensity blob) can be displaced somewhat with respect to the epipolar line. Thus, in practice, instead of searching for points only on the epipolar line, a search algorithm should search a neighborhood around the epipolar line.

  During pattern training we store the maximum offset from the epipolar line that corresponding points may have. The offset for pattern i is denoted as R_i^{max}. During application, after finding a candidate in one image, we restrict the search for points in the neighborhood defined by the offset.

4.2 3D Tests.

- Volume test

  During training, we store the maximum 3D volume spanned by 5 point coplanar patterns. Since coplanar points imply a volume equal to zero, this metric defines an upper bound of how much noise in the 3D points is tolerated before candidate patterns are rejected. During application, all candidate five point configurations are tested to verify if their 3D volume are below the maximum trained volume bound.

- Distance test

  During training, we store the minimum and maximum 3D distances between the points of a pattern in the database. This metric defines an interval of how much noise in the 3D distances is tolerated before candidate patterns are rejected. During application, all candidate point configurations are tested to verify if their 3D distances are contained in the 3D intervals.

  The candidate pair with the best fit (using a least squares method of their 3D distances and comparing this to the pattern in the database) is chosen to be the correct pattern.

5 Results

We have implemented and tested the tracking algorithm in our Personal Space Station (PSS) near-field environment [1]. The PSS consists of a mirror in which stereoscopic images are reflected. The user reaches under the mirror to interact with the virtual world. Two cameras are used to track the space in which the interaction takes place. The interaction space in the PSS is approximately a 50 cm x 50 cm x 50 cm volume. this volume is comprised in both cameras' field of view. The tracking space is illuminated by rings of IR LEDs mounted closely around the camera lenses. Retro-reflective markers are applied to all devices to

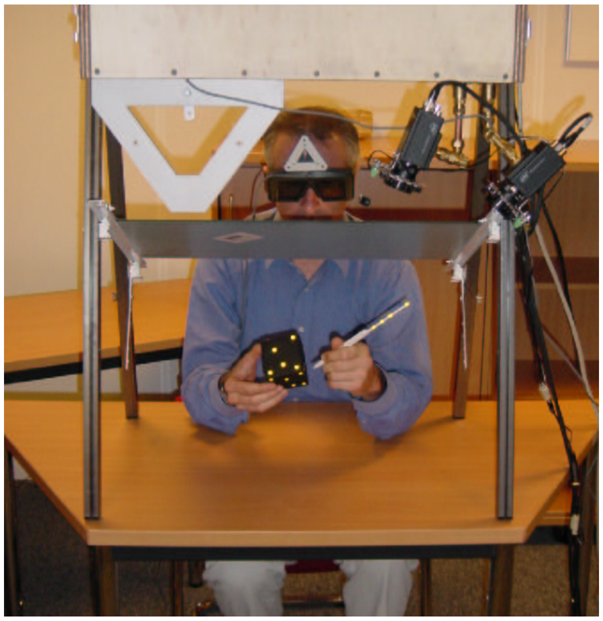


Figure 4. The pen and cube device. A collinear pattern is pasted on a 8 cm rod that is mounted on a plastic pen. Coplanar patterns are pasted on each side of a wooden cube.

be tracked. IR light is reflected by the markers into the lens such that, after thresholding, blobs of white pixels can be found in the acquired image.

Figure 4 shows a photo of two input devices in the PSS. On the right is a plastic pen with a rod mounted on the end of the pen. The pattern is defined as four retro-reflective markers, placed on the rod. The rod is mounted such that the user can hold the pen comfortably without occluding the markers. On the left is a 6x6x6 cm wooden cube. Five markers are placed on each side of the cube. Markers are small circles with a diameter of 0.5 cm and are separated at a minimum distance of 1 cm. The pen and cube are typically used together for two handed input. For example, the cube device is used in the non-dominant hand to position and orient a tethered virtual object, while the pen is used in the dominant hand for precise interaction with the object.

Figure 5 shows three snapshots during an interactive session with the pen and cube devices[1]. The top half of each snapshot is the image taken from the left camera, the bottom half is the image from the right camera. Detected patterns are colored, while points of non-detected patterns are drawn as small white points. Each point is labeled with an identification. In the left snapshot (9 left points and 9 right points), the pen and one side of the cube are detected. In the middle snapshot (15 left points and 14 right points), the pen and two sides of the cube are detected. The left image contains one point on the cube which is not in the right image. In the right snapshot (14 left points, 9 right points), the pen and one side of the cube are detected. The left image contains five points which the tracker detects a pattern, but there is no corresponding pattern in the right image.

[1]The submitted low quality MPEG video shows the complete interactive session.

	Left Image			Right Image		
configs	126	1365	1001	60	33	7
collinear	29	100	59	36	9	1
invariant	4	7	3	14	2	1

Table 1. Total number of tested configurations and the number of configurations that pass the collinearity and invariant interval tests in the input data sets when searching for the marker pattern on the pen device. Input data is taken from the images in figure 5.

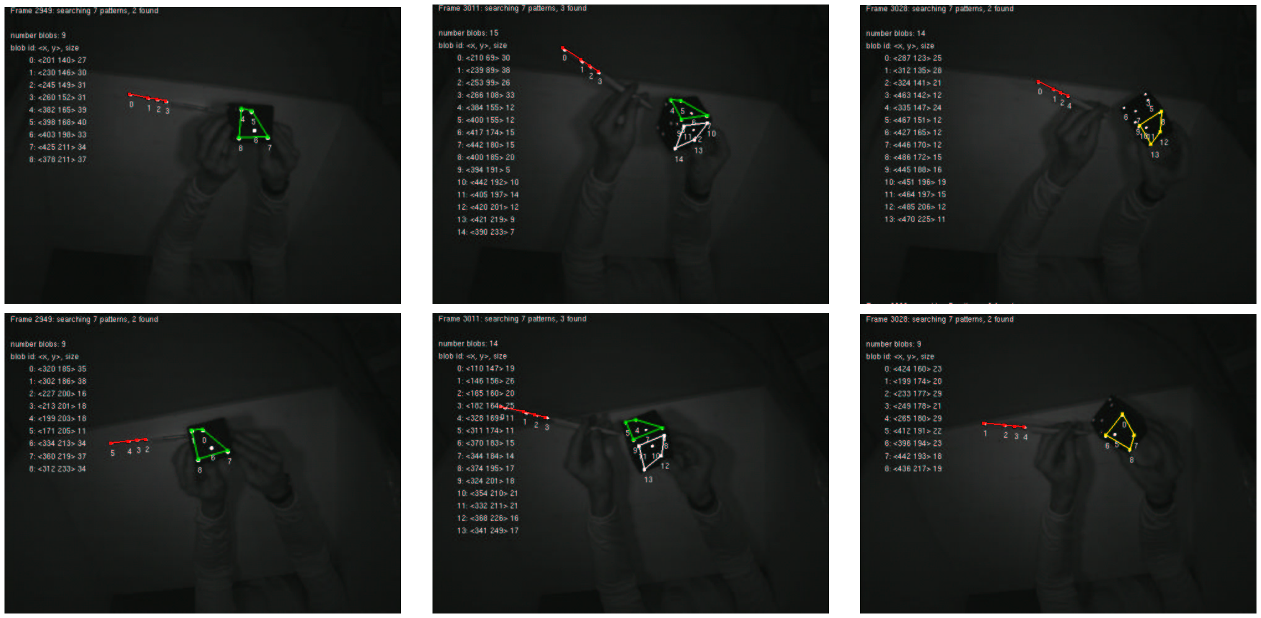
Table 1 lists the number of configurations that pass each test for the point sets in the left and right image. The numbers in the tables are for detecting the collinear pattern (drawn in red in the snapshots in Figure 5). The first row in the table for the left image gives the total number of four point configurations that were tested. The second and third row shows the number of point configurations that passed the collinear test and invariant interval test. The configurations that pass the invariant interval test are placed in the candidate table.

The table for the right image shows the number of four point configurations that were tested. Only points in the neighborhood of the epipolar lines of left image candidate configurations are used to search for right image configurations.

	Left Image			Right Image		
configs	126	3003	2002	30	322	6
collinear	1	648	252	2	36	3
invariant	1	8	4	2	6	2
hull	1	5	2	2	6	2

Table 2. Number of configurations that pass the collinearity, invariant interval, and convex hull tests when searching for the marker patterns on the cube device. In the middle column two patterns are found, while in the left and right column one pattern is found.

Table 2 lists the number of configurations that were tested to detect the coplanar patterns. The first row gives the



```
Frame 2949: searching 7 patterns, 2 found

number blobs: 9
blob id: <x, y>, size
      0: <201 140> 27
      1: <230 146> 30
      2: <245 149> 31
      3: <260 152> 31
      4: <382 165> 39
      5: <398 168> 40
      6: <403 198> 33
      7: <425 211> 34
      8: <378 211> 37
```

```
Frame 3011: searching 7 patterns, 3 found

number blobs: 15
blob id: <x, y>, size
      0: <210 69> 30
      1: <239 89> 38
      2: <253 99> 26
      3: <266 108> 33
      4: <384 155> 12
      5: <400 155> 12
      6: <417 174> 15
      7: <442 180> 15
      8: <400 185> 20
      9: <394 191> 5
      10: <442 192> 10
      11: <405 197> 14
      12: <420 201> 12
      13: <421 219> 9
      14: <390 233> 7
```

```
Frame 3028: searching 7 patterns, 2 found

number blobs: 14
blob id: <x, y>, size
      0: <287 123> 25
      1: <312 135> 28
      2: <324 141> 21
      3: <463 142> 12
      4: <335 147> 24
      5: <467 151> 12
      6: <427 165> 12
      7: <446 170> 12
      8: <486 172> 15
      9: <445 188> 16
      10: <451 196> 19
      11: <464 197> 15
      12: <485 206> 12
      13: <470 225> 11
```

```
Frame 2949: searching 7 patterns, 2 found

number blobs: 9
blob id: <x, y>, size
      0: <320 185> 35
      1: <302 186> 38
      2: <227 200> 16
      3: <213 201> 18
      4: <199 203> 18
      5: <171 205> 11
      6: <334 213> 34
      7: <360 219> 37
      8: <312 233> 34
```

```
Frame 3011: searching 7 patterns, 3 found

number blobs: 14
blob id: <x, y>, size
      0: <110 147> 19
      1: <146 156> 26
      2: <165 160> 20
      3: <182 164> 25
      4: <328 169> 11
      5: <311 174> 11
      6: <370 183> 15
      7: <344 184> 14
      8: <374 195> 17
      9: <324 201> 18
      10: <354 210> 21
      11: <332 211> 21
      12: <368 226> 16
      13: <341 249> 17
```

```
Frame 3028: searching 7 patterns, 2 found

number blobs: 9
blob id: <x, y>, size
      0: <424 160> 23
      1: <199 174> 20
      2: <233 177> 29
      3: <249 178> 21
      4: <265 180> 29
      5: <412 191> 22
      6: <396 194> 23
      7: <442 193> 18
      8: <436 217> 19
```

Figure 5. Three snapshots of images from the left (top row) and right (bottom row) camera. Detected collinear patterns (pen device) are drawn in red. The convex hull of detected coplanar patterns (cube device) are drawn in different colors.

total number of five point configurations that were tested. The second row shows the number of configurations that passed the collinear test for coplanar patterns. The second row shows the number of configurations that passed the invariant interval test. Finally, the convex hull test is performed.

The table for the right image shows the number of five point configurations that were tested. Here too, only points in the neighborhood of the epipolar lines of left image candidate configurations are used to search for right image configurations.

Point configurations that pass all tests in both images are placed in the candidate table. Candidates are then checked with the 3D distance test to determine which point configuration best fits the pattern in the database.

There were no false detections of any of the seven patterns during the interactive session. The marker pattern on the pen device is detected correctly in every frame. In seven of the 3000 frames no coplanar patterns were found. The reason for this is that there are a few angles in which the cube can be held such that both cameras cannot detect complete marker patterns. The retro-reflective markers do not reflect sufficient light at these angles. These lighting circumstances require that a lower threshold is used for blob detection.

The tracker runs on a modern single CPU PC equipped with two Leutron Vision PictPort H4D dual channel frame grabbers and two Leutron Vision LV-7500 progressive scan CCD-cameras. Two Computar H0612FI lenses with a focal length of 6 mm and an F number of 1.2 are fitted to the cameras. Images are grabbed at 30 frames per second. The tracker runs at approximately 28 frames per second, for input data sets similar to the images given above.

6 Discussion

In the previous sections we have described a method for detecting marker patterns based on projective invariant properties. We now discuss some pros and cons of the method.

- Robustness

  The robustness of the pattern tracker when processing noise corrupted data is characterized by two factors: the probability of detection, and the number of false detections. These two measures are intimately correlated; increasing the probability of detection (by increasing threshold intervals) will inherently increase the number of false detections. False detections should be avoided at all costs, since they will cause the associated input device to be tracked at an incorrect position. The robustness of the tracker is thus closely related to the training of projective invariant marker pattern properties.

- Lighting conditions

Noise originates mostly from a combination of lighting conditions, illumination of the retro-reflective markers, and camera parameters. A requirement for the PSS is that it can be used under normal working conditions, including TL-lighting. Patterns may need to be re-trained when office conditions vary significantly, since the width of intervals depend on the noise introduced by the varying lighting conditions during training.

- Occlusion

  Occlusion is an inherent problem in optical tracking. However, there are some cases in which if the cross ratio is known and not all points of a trained pattern are visible, then the remaining point can be reconstructed. For example, if three points of a collinear pattern are known, then the fourth point can be reconstructed if the cross ratio of the pattern in the database is known.

- Complexity analysis

  The total number of tests to be performed is dependent on the positions of the points in the input sets. Seemingly simple sets may require many tests, while more complex sets may require only a few tests. This makes a thorough complexity analysis on the tracking algorithm difficult. Therefore, we can only provide experimental results to illustrate the performance of the system.

- Device construction

  Input devices are easy to construct. All that is needed is to paste collinear or coplanar marker patterns on the device. The training session will store all projective invariant marker pattern properties.

  Some caution must be taken in determining the inter-marker distance for markers in the pattern. The inter-marker distance must be large enough so that there is least one pixel that seperate the blobs in the image. In this way, each blob in the image can be distinguished and 2D points can be determined.

- Stereo correspondence vs indexing

  Using projective and permutation p^2-invariants to search for patterns is more efficient than searching patterns in 3D after performing the correspondence on points.

  Assume N points in the left and right images and that a 4 point pattern is to be searched.

  In the case that there is a 1-1 correspondence for all points, then searching in 3D will need to perform $\binom{N}{4}$ comparisons with the 3D model of the pattern; i.e. every 4 point permutation will need to be compared with the pattern in the database.

  The p^2-invariant will need to perform $\binom{N}{4} 1/4!$ comparisons with the 2D model, since the p^2-invariants are permutation independent.

  In the case that the correspondence is not 1-1 then searching in 3D will need to perform many more comparisons with the 3D model. In the worst case, $\binom{N}{4}^2$ comparisons will be needed.

  The worst case of the p^2-invariant case will require $2\binom{N}{4} 1/4!$ comparisons.

  The combinatorics assume a naive searching method and both approaches can be optimized in many ways. However, when point clouds have a high density, for instance when multiple devices are used in a small working volume, searching for a pattern with p^2-invariant representations is very efficient.

7 Conclusion

In this paper, we have described a new and robust optical tracker algorithm for tracking interaction devices. The tracker uses projective invariant properties of marker patterns to efficiently identify and reconstruct the pose of these interaction devices. The tracking system comprises two different modes: training mode and application mode. Since input data can contain noise, the off-line training session is used to determine deviations in the invariant and 3D properties. These deviations are taken into account when searching for patterns once the tracker is used in application mode.

References

[1] J.D. Mulder and R. van Liere. The personal space station: Bringing interaction within reach. In S. Richer, P. Richard, and B. Taravel, editors, *Proceedings of the Virtual Reality International Conference, VRIC 2002*, pages 73–81, 2002.

[2] Forsyth D, J. Mundy, and E. Grosso. Transformational invariance – a primer. In *Proceedings of the British Machine Vision Conference*, pages 1–6, 1990.

[3] J. Mundy and Z. Zisserman, editors. *Geometric Invariance in Computer Vision*. The MIT Press, 1992.

[4] K. Dorfmüller. Robust tracking for augmented reality using retroreflective markers. *Computers and Graphics*, 23(6):795–800, 1999.

[5] M. Ribo, A. Pinz, and A. Fuhrmann. A new optical tracking system for virtual and augmented reality applications. In *Proceedings of the IEEE Instrumentation and Measurement Technical Conference*, volume 3, pages 1932–1936, Budapest, Hungary, 2001.

[6] P. Meer, R. Lenz, and S. Ramakrishna. Efficient invariant representations. *IJCV*, 26(2):137–152, 1998.

[7] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. The MIT Press, 1993. ISBN 0-262-06158-9.

[8] M. Penna and R. Patterson. *Projective geometry and its applications to computer graphics*. Prentice-Hall, Inc., 1986. ISBN 0-13-730649-0.

[9] K. Kanatani. Computational projective geometry. *CVGIP*, 54(3):333–348, 1991.

Appendix

The tracking algorithm in the paper uses projective and permutation p^2-invariants, as published by Meer et al.[6]. In this appendix, we briefly describe the p^2-invariant of four collinear points.

The cross ratio of four collinear 2D points A, B, C, D is defined as

$$\lambda = \frac{|AC| \,/\, |BC|}{|AD| \,/\, |BD|}$$

where $|AC|$ is the length of the line segment from point A to C. It is known that the cross ratio of any permutation of the four points will yield only six values:

$$\begin{aligned}
\lambda_1 &= \lambda, \\
\lambda_2 &= 1/\lambda, \\
\lambda_3 &= \lambda - 1/\lambda, \\
\lambda_4 &= \lambda/(\lambda - 1), \\
\lambda_5 &= 1/(1 - \lambda), \\
\lambda_6 &= 1 - \lambda
\end{aligned}$$

A p^2-invariant defined by Meer et al. is computed as a linear combination of

$$J_1(\lambda) = (\lambda^6 - 3\lambda^5 + 3\lambda^4 - \lambda^3 + 3\lambda^2 - 3\lambda + 1)/(\lambda^2(\lambda - 1)^2)$$

and

$$J_2(\lambda) = (2\lambda^6 - 6\lambda^5 + 9\lambda^4 - 8\lambda^3 + 9\lambda^2 - 6\lambda + 2)/(\lambda^2(\lambda - 1)^2)$$

as

$$J(\lambda) = J_2(\lambda)/J_1(\lambda)$$

Thus, independent of how the labels of four points are chosen, when the cross ratio λ is used as argument in J[.], the same value is obtained.

The method is generalized to compute the p^2-invariant of a point set in the n-dimensional projective space. See [6] for more details.