The Homography as a State Transformation Between Frames in Visual Multi-Target Tracking

Jacob H. White  
*Brigham Young University, jacob.white@byu.net*

Randal W. Beard  
*Brigham Young University, beard@byu.edu*

Follow this and additional works at: [https://scholarsarchive.byu.edu/studentpub](https://scholarsarchive.byu.edu/studentpub)  
Part of the [Electrical and Computer Engineering Commons](https://scholarsarchive.byu.edu/electricalandcomputerengineeringcommon)

**BYU ScholarsArchive Citation**  
[https://scholarsarchive.byu.edu/studentpub/261](https://scholarsarchive.byu.edu/studentpub/261)
The Homography as a State Transformation Between Frames in Visual Multi-Target Tracking

Jacob H. White* and Randal W. Beard†
Brigham Young University, Provo, UT 84602

I. Introduction

There are many applications where an unmanned aerial vehicle (UAV) is required to visually track a ground-based target using an on-board camera. In these applications, successful target tracking often requires that past estimates of the target be transformed into the current image frame so that moving targets can be correctly identified [1]. If the target is moving on an approximately planar surface, then the coordinate transformation can be accomplished using the homography matrix.

Figure 1 shows the relevant geometry. Suppose that $P$ is the target of interest on the ground and that $P$ lies on a planar surface with unit normal vector $\hat{n}$, where $\hat{n}$ points into the earth. Furthermore, suppose that the UAV captures two images of $P$ at time indices $k-1$ and $k$. The coordinate frame of the camera at time index $k-1$ is denoted $F^{k-1}$ and the coordinate frame of the camera at time index $k$ is denoted $F^k$. Let $P^{k-1}$ be the target’s position at expressed in frame $F^{k-1}$, and let $P^k$ be the target’s position expressed in frame $F^k$. Let $d_{k-1}$ be the perpendicular distance from the camera to the ground plane, at time $k-1$ and let $d_k$ be the distance from the camera to the ground plane at time $k$. The relative pose between the camera frames $F^{k-1}$ and $F^k$ is given by the vector $t_{k-1/k}$ from $F^{k-1}$ to $F^k$ and the rotation matrix $R_{k-1}^k$ that transforms vectors expressed in $F^{k-1}$ to vectors expressed in $F^k$. We will use a superscript to denote the frame in which a vector is expressed. Therefore $\hat{n}^{k-1} \in \mathbb{R}^3$ is $\hat{n}$ expressed relative to the basis vectors of $F^{k-1}$.

With reference to Figure 1 we have

$$P^k = R_{k-1}^k P^{k-1} + t_{k-1/k}^k.$$  

Observing that

$$d_{k-1} = (\hat{n}^{k-1})^T p^{k-1},$$

we get that

$$P^k = R_{k-1}^k P^{k-1} + \frac{t_{k-1/k}^k}{d_{k-1}} \left( (\hat{n}^{k-1})^T p^{k-1} \right)$$

$$= \left( R_{k-1}^k + \frac{t_{k-1/k}^k}{d_{k-1}} (\hat{n}^{k-1})^T \right) p^{k-1}. \quad (1)$$

*Research Assistant
†Professor, Department of Electrical and Computer Engineering, beard@byu.edu, AIAA Associate Fellow.
Fig. 1 The geometry for the derivation of the homography matrix between two camera poses

Let $p^k$ represent projection of $P^k$ onto the normalized image plane of camera $k$. In other words $p^k$ is the location of $P$ in the image plane of camera frame $F^k$. If $P^k = [X^k\ Y^k\ Z^k]^\top$, then

$$p^k = \begin{bmatrix} X^k \\ Z^k \\ Y^k \\ 1 \end{bmatrix}.$$  

Therefore, Equation (1) can be written as

$$(\frac{Z^k}{Z^{k-1}}) p_k = \left(R^k_{k-1} + \frac{t^k_{k-1/k}(\hat{n}^{k-1})^\top}{d^k_{k-1}}\right)p_{k-1}.  \quad (2)$$

Defining the scalar $\gamma_k \triangleq Z^k/Z^{k-1}$ we obtain

$$\gamma_k p_k = \left(R^k_{k-1} + \frac{t^k_{k-1/k}(\hat{n}^{k-1})^\top}{d^k_{k-1}}\right)p_{k-1}. \quad (3)$$
The matrix

\[ H_k \doteq \left( R_{k-1}^k + \frac{t_{k-1/k}^k (\hat{p}_{k-1}^k)^\top}{d_{k-1}} \right) \]  

(4)
is called the Euclidean homography matrix at time \( t_k \). Equation (3) demonstrates that the homography matrix \( H_k \) transforms the pixel location of \( P \) at time index \( k - 1 \) into the pixel location of \( P \) at time index \( k \). The homography is defined up to a scale factor \( \gamma_k \); any scaling of the matrix results in an equivalent matrix. In practice, the scale factor \( \gamma_k \) is determined as the inverse of the third element of \( H_k p_{k-1} \), so as to ensure that the third element of \( p_k \) is one.

The homography between frames can be estimated by first detecting features in one image and tracking them using optical flow or matching them to features in the next image. A homography is calculated from these feature pairs using random sample consensus (RANSAC) \([3]\), or least median of squared (LMedS) \([4]\). These algorithms randomly pick 4-point subsets, calculate a homography hypothesis from each subset, score the hypotheses, and pick the best one. The hypothesis is then refined using an iterative method on only the inlier points \([5]\). There are readily-available open-source implementations of these algorithms, such as the \texttt{findHomography} function in OpenCV \([6]\).

Video tracking algorithms from a UAV typically represent the target’s position \( p \), velocity \( v \), and position covariance \( \Sigma \) in the image plane. These states are updated using a Kalman filter \([7]\), or in the case of multiple targets or uncertain measurement association, a Probabilistic Data Association Filter (PDAF) \([8]\). The Kalman filter prediction step advances the target state forward in time. However, after the Kalman filter prediction step, the target’s state is still represented in the previous camera frame. This state must also be transformed into the current camera frame. The purpose of this technical note is to show how to correctly transform the image plane projection of the target’s position, velocity, and covariance from one frame to the next frame.

II. The Homography as a Non-Linear Transformation

While the homography is a matrix, and matrices define linear transformations, the homography is not a linear transformation due to the scale factor \( \gamma \). To see this, represent the homography as a block matrix,

\[
H = \begin{bmatrix}
H_1 & h_2 \\
2x2 & 2x1 \\
\hline
h_3^\top & h_4 \\
1x2 & 1x1
\end{bmatrix}.
\]

(5)

Solving for \( \gamma \) gives

\[
\gamma \begin{bmatrix}
p^2 \\
1
\end{bmatrix} = \begin{bmatrix}
H_1 & h_2 \\
\hline
h_3^\top & h_4
\end{bmatrix} \begin{bmatrix}
p^1 \\
1
\end{bmatrix},
\]

(6)

and
\[ p_2 = \frac{H_1 p_1 + h_2}{\gamma} = \frac{H_1 p_1 + h_2}{h_3 p_1 + h_4}. \] 

(7)

To simplify notation, let us represent this transformation as the function

\[ g(p_1) = \frac{H_1 p_1 + h_2}{h_3 p_1 + h_4}. \] 

(8)

This equation is equivalent to (3), but more clearly shows the non-linear elements of the transformation. This will be useful in subsequent sections when deriving the target velocity and covariance transformations in the image plane.

III. Transforming the Target’s Velocity

We will now derive a method for transforming the target’s velocity between frames. Define the target’s velocity in the image plane as the net velocity of the target, after the ego-motion of the UAV has been subtracted. Thus according to this definition, if an object is stationary it will also have zero velocity in the image plane, regardless of the motion of the UAV.

The equation to transform the target’s position in the image plane is given in (8). However, transforming the target’s velocity in the image plane using this same equation would add a non-zero offset to the velocity of \( \frac{h_2}{h_3 p_1 + h_4} \) whenever the camera translates or rotates. Thus homogeneous coordinates should not be used when transforming the target’s velocity. The desired transformation for the velocity should instead be a \( 2 \times 2 \) matrix that only stretches and rotates the velocity, rather than a \( 3 \times 3 \) matrix. Similarly, the position covariance of the target is a \( 2 \times 2 \) matrix and also requires a \( 2 \times 2 \) matrix to stretch or rotate it into the next frame.

The temptation is to ignore the non-linear component of the transformation, and simply take the derivative of (6), which results \( H_1 \), the upper-left \( 2 \times 2 \) sub-matrix of the homography matrix. This can produce correct results under certain special cases, as will be discussed later in the paper. However, the problem with this approach is that it ignores the scale factor \( \gamma \). For example, multiplying the homography matrix by two gives an equivalent homography, but since the submatrix is also multiplied by two, it will produce a different velocity!

An improved technique is to represent the velocity with two points and subtract the difference of the transformed points to obtain the transformed velocity. Suppose the position of the target in the first image is given by vector \( p^1 \) and its velocity by \( v^1 \). In \( \delta_t \) seconds, the target will be at position \( q^1 = p^1 + \delta_t v^1 \). This point can be transformed into the second frame using (8) to give \( q^2 \). The difference between these two points in the second frame is a good approximation of the target’s velocity in the second frame. This gives

\[ v^2 \approx \frac{g(p^1 + \delta_t v^1) - g(p^1)}{\delta_t}. \] 

(9)
The smaller the time difference used in calculating (9), the more accurate the approximation will be. If we take the limit as $\delta_t$ approaches 0, we obtain

$$v^2 = \lim_{\delta_t \to 0} \frac{g(p^1 + \delta_t v^1) - g(p^1)}{\delta_t}.$$  \hspace{1cm} (10)

Observe that (10) is the definition of a calculus derivative, in the direction $v^2$. Thus the correct approach to obtain an exact answer is to calculate the derivative of the homography transformation.

By differentiating (8) using the quotient rule we obtain

$$G(p) = \frac{\partial}{\partial p} g(p).$$ \hspace{1cm} (11)

$$= \frac{\delta}{\partial p} \frac{H_1 p + h_2}{h_3 p + h_4}$$ \hspace{1cm} (12)

$$= \left( h_3 p + h_4 \right)^{-1} \left( H_1 (H_1 p + h_2) h_3^T - H_1 p h_3^T \right).$$ \hspace{1cm} (13)

This transforms the image plane velocity of the target to the next frame as

$$v^2 = G(p) v^1.$$ \hspace{1cm} (14)

We can now return to the simple, but mathematically incorrect submatrix method mentioned above, to determine under what conditions it will produce the same answer as (14). Suppose the velocity is transformed to the next frame according to the equation

$$v^2 \overset{?}{=} H_1 v^1.$$ \hspace{1cm} (15)

This equation will be equivalent to (14) if the matrices $G$ and $H_1$ are equal. Using the definition of $G$ in (13) we find that there are two conditions that must hold for the matrices $G$ and $H_1$ to be equal. First, the vector $h_3$ must be zero, and second, the scalar $h_4$ must be one.

In order for the vector $h_3$ to be zero, the camera’s motion must meet certain conditions. The homography matrix is the sum of a rotation matrix and a outer product from the translation vector and plane normal, as shown in (4). In order for rotation-contributed term of $h_3$ to be zero, the rotation matrix must be a rotation about the optical axis, or z axis, of the camera. Any other rotation will cause a large difference in the result. In order for the translation-contributed term of $h_3$ to be zero, either the translation in the z direction must be zero, or the plane normal vector x and y components must be zero. In other words, translating the camera in the x and y directions is allowed, but translating the camera in the z
direction is only allowed if the camera is pointed directly at the plane.

The second condition for (14) and (15) to be equivalent is that \( h_4 \) must equal 1. Since the homography is defined up to a scale factor, \( h_4 \) will only be 1 if the scale factor happens to be exactly the right number to make \( h_4 \) be 1. This can occur by coincidence, or if the library used to calculate the homography intentionally scales the homography so that \( h_4 \) is 1. As it turns out, OpenCV’s \texttt{findHomography} function happens to scale the homography so that \( h_4 \) is 1. This enables (14) and (15) to be equivalent if the conditions described above are also met.

IV. Transforming the Target’s Covariance

![Diagram of the target’s position, velocity, and position covariance in the image plane](image)

*Fig. 2 The target’s position, velocity, and position covariance in the image plane*

Suppose that the position of the target in frame \( k-1 \) is represented by the random vector \( X^{k-1} \), with mean \( \mu^{k-1} \) and covariance \( \Sigma^{k-1} \) (see Figure 2). Using the definition of \( g(\cdot) \) given in (8), any realizations of the random vector \( X \) can be transformed into the current frame \( k \) by

\[
X^k = g(X^{k-1}).
\]  

(16)

Using a first-order Taylor series approximation of \( g \) evaluated at the mean \( \mu^{k-1} \) of the random vector \( X^{k-1} \) gives

\[
g(X^{k-1}) \approx g(\mu^{k-1}) + \frac{\partial}{\partial z} g(z) \bigg|_{z=\mu^{k-1}} (X^k - \mu^{k-1})
\]

\[
\approx g(\mu^{k-1}) + G(\mu^{k-1}) (X^{k-1} - \mu^{k-1}),
\]
where $G$ is defined in (13). With this linear approximation of the homography transformation, we can calculate the covariance $\Sigma^k$ in the current frame, giving

$$
\Sigma^k = E[(X_k - \mu_k)(X_k - \mu_k)^T]
\approx E\left[\left(g\left(\mu^{k-1}\right) + G\left(\mu^{k-1}\right)(X^{k-1} - \mu^{k-1})\right)\left(g\left(\mu^{k-1}\right) + G\left(\mu^{k-1}\right)(X^{k-1} - \mu^{k-1})\right)^T\right]
= E\left[G\left(\mu^{k-1}\right)(X^{k-1} - \mu^{k-1})\left(G\left(\mu^{k-1}\right)(X^{k-1} - \mu^{k-1})\right)^T\right]
= E\left[G\left(\mu^{k-1}\right)(X^{k-1} - \mu^{k-1})(X^{k-1} - \mu^{k-1})^TG^\top\left(\mu^{k-1}\right)\right]
= G\left(\mu^{k-1}\right)E\left[(X^{k-1} - \mu^{k-1})(X^{k-1} - \mu^{k-1})^TG^\top\left(\mu^{k-1}\right)\right]
= G\left(\mu^{k-1}\right)\Sigma^{k-1}G^\top\left(\mu^{k-1}\right).
$$

(17)

This produces a mathematically-correct approximation to the true covariance transformation. However, as mentioned before it can be tempting to use the upper-left $2 \times 2$ sub-matrix of the homography matrix

$$
\Sigma^k \approx H_1\Sigma^{k-1}H_1^\top.
$$

(18)

This is not mathematically based and can cause the problems described in the previous section.

A simple Matlab script was used to illustrate the difference between using the covariance transform (19) and the covariance transform (18). For this plot, 20,000 points were sampled from a covariance distribution. Each of these points were then transformed to the next frame using a homography matrix generated from an out-of-the-plane y-axis rotation of $\frac{\pi}{8}$ radians (22.5 degrees). The results are shown in Figure 3. The sample covariance of the points in the first frame is shown in dashed red and is nearly identical to the true distribution covariance. The sample covariance of the points in the second frame is shown in dashed blue. The expected covariance calculated using covariance transform (18) is shown in green. The transformed covariance is very close to the sample covariance. On the other hand, the covariance calculated using (19) is very different from the sample covariance.

The small difference between expected distribution and the measured covariance is because the partial derivative is not constant over the entire distribution. Thus the covariance propagation is not exact. However, if the covariance is small enough, the effects of the non-linearities are reduced and the covariance after the transformation becomes very close to the true covariance. If certain applications expect large covariances, better results may be obtained with the transformations used in the unscented Kalman filter (UKF) and the particle filter (PF). These filters are much better at dealing with non-linearities. However, for our use cases, the Kalman filter (KF) transformation is accurate enough.
Fig. 3 The true covariance $\Sigma_1$ (solid red) with almost identical measured covariance (dashed red) are transformed by a y-axis rotation of $\frac{\pi}{8}$ radians (22.5 degrees). Calculating the expected covariance of the transformed points using the sub-matrix method (cyan) is much less accurate than using the derivative method (green).
V. Video Results

The derivative \([14], [18]\), submatrix \([15], [19]\), and the two-point \([9]\) methods were also tested on six video sequences captured by a hand-held camera. Each video sequence shows a target which starts near the center of the field of view. Three of the video sequences show translations in each of the three unit vector directions. The other three videos show camera rotations about each of the three axes. The homography is calculated using OpenCV’s findHomography function. The homography is used to update the target’s position, velocity, and position covariance in the image plane. Note that because OpenCV scales the homography so that the bottom-right entry of the homography matrix is always one, it is possible to obtain identical results for certain transformations listed at the end of section III. To make the results easier to see, the Kalman filter time and measurement updates were omitted. In other words, the target in the video is actually stationary, but the velocity calculations done on the video sequence are still valid.

The results are shown in Figure 4 and at https://youtu.be/ixyq1qa9m70. The position and velocity are shown by arrows and the covariance is shown by an ellipse representing the 1-sigma bound. The first frame of each video sequence is shown on the left and the last frame is shown on the right. The derivative, submatrix, and two-point methods are shown in green, cyan, and red respectively. Notice that all three methods give similar results when the translation is in the x, y, and z directions. As mentioned earlier, the z direction translation can give similar results if the camera is pointed at the plane. All three methods also give similar results for rotations about the camera’s z axis. However, the three methods give different results for rotations about the x and y axes. The derivative and two-point give good results, but the submatrix method has large unexpected stretching and shrinking of the velocity and covariance estimates as the camera rotates.

VI. Matlab UAV Simulation Results

A Matlab UAV simulation was used to test the position, velocity, and covariance transformations defined above, within a Kalman filter target tracker. In this simulation, a target is traveling at a constant velocity in the world frame, while being followed by a UAV. A gimbaled camera is mounted on the UAV and rotates to keep the target in the center of the field of view. In addition to keeping the target in the center of the field of view, the camera is also rotating slowly about its optical axis at an approximately constant angular rate. Both the UAV and gimbal are controlled using simple PID controllers. However, the UAV gains are not well tuned, resulting in significant oscillations. Figure 5 shows the UAV represented with a red asterisk and the target’s position represented with a blue asterisk. Figure 6 shows the world position of the UAV and the target.

At each time step, the camera returns the x and y pixel locations of the target in the image plane. The position and velocity of the target are updated using a Kalman filter. However, at each time step after the Kalman filter time update, the position and velocity must be transformed into the next camera frame. All of the methods transform the target’s position into the next frame using \([8]\), but the velocity and covariance are transformed into the next frame using...
Fig. 4 Comparison of three methods of transforming points, velocities, and covariances
different transformations. The resulting estimates and their error are shown in Figure 7. For a baseline comparison, no transformation is performed on the velocity and covariance (blue). This results in significant error as the camera rotates about its optical axis. The second comparison is the submatrix method, which transforms the velocity and covariance using (15) and (19). This transformation is fairly accurate when the UAV is at a constant attitude. However, when the UAV tips forward or backward, there are large spikes in the error.

The third comparison uses the derivative of the homography transformation. This method transforms the velocity and covariance using (14) and (18). This method is the most accurate. However, there is still error because the target’s state is represented as a constant-velocity target in the image plane. Constant velocity in the world plane is not constant velocity in the image plane. For example, imagine a camera pointed at the horizon as a motorcycle drives past. The motorcycle may be moving at a constant velocity, but its velocity in the image plane get smaller and smaller as its distance to the camera increase.

The last comparison also uses the derivative of homography transformation to transform the velocity and covariance as shown in (14) and (18). However, instead of performing the Kalman filter time update in the image plane, the states are temporarily transformed into the world ground plane to perform the update. This transformation is also accomplished using a homography. This method has no error for the entire simulation because the correct homography state transformation is used and the target model is exact.

VII. Conclusion

In this work we have derived a method of transforming target states represented in the image plane between consecutive frames of a video sequence. We have demonstrated the accuracy of this state transformation on both a hand-held video sequence and in a Matlab target tracking simulation. We have also shown the potential pitfalls when transforming states between frames using the homography and how these pitfalls can affect performance. Future work includes developing improved models to estimate target depth from a monocular camera in order to track in 3D without the flat-earth assumption.

Acknowledgment

This work has been funded by the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation Industry/University Cooperative Research Center (I/UCRC) under NSF award Numbers IIP-1161036 and CNS-1650547, along with significant contributions from C-UAS industry members.

References

Fig. 5  Matlab UAV simulation
Fig. 6  UAV and target world position over time
Fig. 7  The target’s position and velocity in the camera frame in pixels, estimated using a Kalman filter


