On Continuous Time Lucas-Kanade Tracking using Lessons from Adaptive Control

Project report for the Fall 2021 offering of ENEE765

Levi Burner

Abstract

Tracking objects in the visual-field is required for numerous computer vision and robotics applications. The Kanade-Lucas-Tomasi (KLT) tracker is a workhorse that allows this to be done quickly and efficiently. The basic idea is that a template image of a tracked object (say a ball, or a picture on the wall) should be equal to some subset of the incoming pixels from a camera. However, because the camera's available on today's market are fundamentally discrete time, the KLT tracker considers only discrete time optimization, without necessarily considering the dynamics of the camera under motion. Because of this, algorithms using a KLT tracker may require an internal optimization loop to "converge" before emitting state estimates. This can result in lag, and there is always the problem of deciding when the optimization has "converged". Towards alleviating this issue, this project considers a simplified version of the problem with the restriction that all methods used must be able to evaluated as part of a continuous time system. To make the problem tractable, several simplifying assumptions are made, including no rotation, motion along only the Z axis, and that the image patch tracked that is parallel to the imaging plane. With this framework setup, Newton-Raphson Flow and continuous time-regression (as used for deterministic system identification) are used to simultaneously identify the four states and one unknown parameter of the system. Namely, the states are the X, Y, Z position of the camera and the velocity along the Z axis. The single system parameter is the position of the plane along the worlds Z axis. The method was shown to work in simulation, however a rigorous proof of stability is still lacking. However, future work may find sufficient conditions for stability and some suggestions for accomplishing this are given.

Key words: Newton-Raphson Flow, System Identification, Lucas-Kanade tracking, Optical-Flow, Observability after Perspective Transformation

1 Introduction

The Kanade-Lucas-Tomasi (KLT) tracker is a workhorse in computer vision that allows tracking of points or "patches" across a sequence of images. While the algorithm has been well studied in discrete time, in the patch-tracking case it relies on an internal optimization loop that must "converge" before the arrival of the next frame [1–3]. The number of iterations the optimizer must complete is typically unknown apriori due to unknown magnitude of motion of the tracked patched, and convergence rates that depend on the texture of the patch. Further, heuristics such as "small" magnitude of the parameter updates are often used as the stopping condition for the optimizer [3]. Thus, there is much to be done in understanding mathematically rigorous conditions which ensure the KLT's algorithms reliability.

There is a striking historical parallel that can be drawn between the KLT situation and the story of the "MIT gradient rule". The gradient rule was known to work well in practice, but sufficient conditions for correct operation were not rigorously understood. When the gradient rule was deployed on an experimental aircraft, there were many

⋆ Special thanks to Professor Krishnaprasad for both teaching ENEE765 and encouraging exploration of this unconventional idea.

Email address: lburner@umd.edu (Levi Burner).

successful flights. However, eventually the gradient rule failed, leading to loss-of-life of the pilot [4]. Similarly, the KLT tracker is essential to many robotics applications, and has been tested extensively, but a rigorous understanding of necessary and sufficient conditions for stability are lacking.

To work towards solving this fundamental problem, this project sought a continuous time extension of the Lucas-Kanade (LK) tracker, for static, planar scenes, and known three dimensional acceleration of the camera. We call it the LK tracker because Tomasi's contribution was to introduce the criteria for selecting patches to track and when to stop tracking them. The literature frequently seems to refer to the LK-tracker when the patch selection process is not being discussed (as is the case here).

Originally, the goal was to employ a mathematical approach inspired by the Sondhi and Mitra's seminal work on adaptive filtering and the insightful mathematical analysis provided by Khalil on Model Reference Adaptive Control. By doing so it was hoped that necessary and/or sufficient conditions on the acceleration of the camera, and bounds on the "learning-rate" required for 1st or 2nd order gradient flows, to achieve successful "tracking" of a patch in a planar scene would be found [5,6].

A lot of the time spent on this project was put towards attempting to lay down the mathematical framework required to realize the above goals. However, due to the complicated and nonlinear nature of the equations that model the camera, it was not satisfactorily worked out (I have stacks of notes to prove it). In order to finish with something on time, it became necessary to significantly restrict the scope of the system considered. Specifically to the case of a planar scene, perpendicular to the optical axis of the camera, where the camera moves only along the Z axis, and feedback is provided by both the camera and linear accelerometer (gravitational bias removed). Thankfully, considering this greatly simplified form turned out to be enough to gain insights useful for future work. I am convinced that with a little more work it would not be hard to extend the methods to full three dimensional motion with gravitational bias in the acceleration measurements. Further, the case of a plane slanted in a single direction should not be terribly difficult to work out. However, the case of a plane slanted in any direction remains illusive. Further, the proof of stability of the even the simplest form of the problem still needs more work, and this is likely to be the hardest part. Such a proof of stability should ensure ensure convergence of the observed states and estimated plane parameters to their true values. This includes the metric depth as well the relation between all points of the patch at all times.

Regardless, the continuous time Lucas-Kanade problem is still exciting and novel because it is more than just a mathematical curiosity. By formulating the problem in continuous time, numerical methods will be able to estimate solutions to the resulting differential equation. The bounds on the quality of the resulting approximation may provide rigorous justification for the required frame-rate of a traditional camera given appropriate bounds on trajectory and the bandwidth of scene texture. Further, such analysis when extended to the stochastic setting could allow understanding the best trajectory for a camera to follow in order to achieve a minimum variance estimate of position and homography. This is nothing more than a minimum-variance control problem with a camera. It seems likely that the trajectory will depend on the geometry and the texture of the scene the camera is moving in. Thus, I hope to be able to continue working on this topic in the future.

In the Method's section that follows, the mathematical framework for the imaging system is introduced first. Then the adaptation of the Lucas-Kanade tracker to continuous time is done using Newton-Raphson flow [4]. It is shown that image information alone is not enough to determine the state and the planes parameters. Thus, continuous-time regression is introduced to determine the parameters of the planar patch under consideration using theory from class which is also found in [7]. With the mathematical foundations in place, it shown how to integrate everything above into a Luenberger style observer and provide a brief discussion on how a proof of stability might proceed. The method's section concludes with describing a simulation used to validate the proposed method and ensure it is not "totally wrong". Afterwards, typical Results, Discussion, and Conclusion sections appear.

2 Methods

2.1 Formulation

2.1.1 Imaging Model

Consider a camera pointed at a planar patch in a scene as illustrated by Figure 1. The camera is assumed to move along the Z axis only, while the the optical axis points in the Z direction. In practice, this assumption of no-rotation

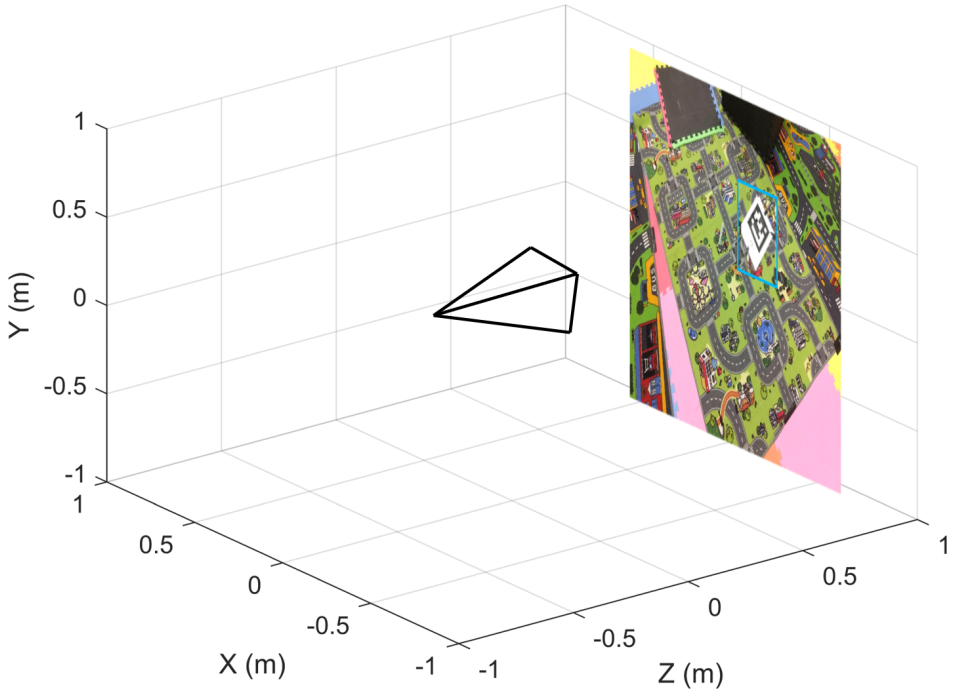

Fig. 1. The above figure shows the coordinate system and the "tracked patch" relative to the initial camera position.

is a reasonable assumption because over short time periods inexpensive gyroscopic sensors can be used to compensate for orientation changes.

Let the set of 3D points associated with the planar patch be given by the set M. Let the planes normal be given by $[0, 0, 1]$. This restriction forces the plane to be parallel to the imaging plane. While this simplifying assumption reduces accuracy in practice, it helps keep the following derivations from becoming too complicated to produce insights. Finally, let d be the depth of the plane at $X_1, X_2 = 0$. Then the set M is defined by

$$M = \left\{ X \in \mathbb{R}^3 \,|\, X_3 = d \right\}. \tag{1}$$

Let the camera's positional trajectory be given by $T(t) \in \mathbb{R}^3, \forall t \geq 0$ and assume it to be continuous. Further, let $T(0) = 0$ meaning that the camera is assumed to start at the origin. Then it is easy to verify that position of a point X on the planar patch in the camera frame is $X - T(t)$. Then, assuming a pinhole model, a point X on the patch M projects onto point $x \in \mathbb{R}^2$ on the image plane. Define the function $x(t, X)$ to represent the position of this projection.

$$x(t, X) \coloneqq \frac{1}{X_3 - T_3(t)} \underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{K} [X - T(t)] \tag{2}$$

Where K is called the camera "intrinsics" matrix and encodes the center of the imaging surface and the focal length. For this project, it is assumed $K = I_{3\times 3}$. This can be safely assumed in practice if the camera is calibrated. Notice that $x(t, X) \in \mathbb{R}^3$ with $x_3(t, X) = 1, \forall t \geq 0$. This definition is common in computer vision and is called a "homogeneous coordinate". Also notice that clearly $T_3(t) \neq 0, \forall t$ else the equation is ill-posed. In practice $\exists \epsilon > 0$ s.t. $T_3(t) \geq \epsilon, \forall t$ else the camera cannot "see" the patch.

It is desirable to write the right hand side of this projection equation entirely in terms of $x(0, X), T, d$. This development would mean that the position of all points of the planar patch in the imaging plane could be understood

in terms of their initial position in the imaging plane, the position of the camera, and the planes parameters. In computer vision the result is called a perspective transformation, however it is not usually written in terms of the minimum number of physical quantities, so it is re-derived here [3,8].

Because K is always invertible, $x(0, X)$ can be brought in through the relation $X_3 x(0, X) = X$ revealing

$$x(t, X) \coloneqq \frac{1}{X_3 - T_3(t)} \left[X_3 x(0, X) - T(t) \right] \tag{3}$$

However since $X_3 = d$

$$x(t, X) \coloneqq \begin{bmatrix} \frac{d}{d - T_3(t)} & 0 & \frac{-T_1(t)}{d - T_3(t)} \\ 0 & \frac{d}{d - T_3(t)} & \frac{-T_2(t)}{d - T_3(t)} \\ 0 & 0 & 1 \end{bmatrix} x(0, X) \tag{4}$$

This expression is closely related to the "affine homography" from computer vision. However, the affine homography is usually not written in terms of real quantities as above and it is allowed to have arbitrary values in all entries except the bottom row.

For convenience define

$$\begin{aligned} p_1(t) &\coloneqq \frac{d}{d - T_3(t)} \\ p_2(t) &\coloneqq \frac{-T_1(t)}{d - T_3(t)} \\ p_3(t) &\coloneqq \frac{-T_2(t)}{d - T_3(t)} \\ p(t) &\coloneqq \begin{bmatrix} p_1(t) & p_2(t) & p_3(t) \end{bmatrix}^T \\ W(t, p, x) &\coloneqq \begin{bmatrix} p_1(t) & 0 & p_2(t) \\ 0 & p_1(t) & p_3(t) \\ 0 & 0 & 1 \end{bmatrix} x \end{aligned} \tag{5}$$

.

Then $x(t, X)$ can be expressed concisely as

$$x(t, X) = W(t, p, x(0, X)) \tag{6}$$

.

Thus, the projection of points X on the planar patch in the imaging plane ($x(X, t)$) at all times is a continuous function of their initial point on the imaging plane, $x(t, X)$, the planes depth d, and the position of the camera. In computer vision W is called a "warp" function.

Define the image intensity function $I(t, x)$ to be the the brightness of the scene projected onto pixel x at time t. Using W, $I(t, x)$ can be written in terms of $I(0, x)$ for all x corresponding with a point on the patch X.

$$I(t, W(t, p, x(0, X))) = I(0, x(0, X)) \tag{7}$$

This constraint is known as the "brightness constraint" in computer vision. Optimizing this constraint using the parameters p is the basis of the Lucas-Kanade tracker. Commonly $I(0, x(0, X))$ is called the "template" image. Define $I_T(x) \coloneqq I(0, x(0, X))$ to represent this template image.

2.1.2 A continuous time extension of Lucas-Kanade

Here the Lucas-Kanade tracker for the forward additive case is derived using our notation. The Lucas-Kanade method attempts to find the parameters p through Gauss-Newton iteration over the squared l_2 norm of error in the brightness constraint over the patch. Specifically the cost is

$$J(p) \coloneqq \frac{1}{2} \iint_{X \in M} \left[I(t, W(t, p, x(0, X))) - I_T(x(0, X)) \right]^2 dX \tag{8}$$

.

Minimization proceeds by iteratively minimizing J using the linearization of $I(t, W(t, p, x(0, X)))$. Let $\tilde{J}$ represent the cost with this substitution, then

$$\tilde{J}(\Delta_p) \coloneqq \frac{1}{2} \iint_{X \in M} \left[I(t, W(t, p, x(0, X))) + \left(\frac{\partial I(t, W(t, p, x(0, X))}{\partial x} \right)^T \frac{\partial W(t, p, x(0, X))}{\partial p} \Delta_p - I_T(x(0, X)) \right]^2 dX \tag{9}$$

.

The notation becomes bulky quickly. If the function arguments are dropped it can be understood more clearly

$$\tilde{J}(\Delta_p) \coloneqq \frac{1}{2} \iint_{X \in M} \left[I + \nabla_x I \frac{\partial W}{\partial p} \Delta_p - I_T \right]^2 dX \tag{10}$$

.

The minimizing Δ_p^* is then

$$\Delta_p^* \coloneqq \arg\min_{\Delta_p} \tilde{J}(\Delta_p) = - \left[\left(\nabla_x I \frac{\partial W}{\partial p} \right)^T \left(\nabla_x I \frac{\partial W}{\partial p} \right) \right]^{-1} \iint_{X \in M} \left(\nabla_x I \frac{\partial W}{\partial p} \right)^T \left[I - I_T \right] dX \tag{11}$$

.

Lucas-Kanade methods then proceed by iterating $p_{k+1} = p_k + \Delta_{p_k}^*$ until $\| \Delta_{p_k}^* \|$ is small.

Historically, Gauss-Newton iteration is used because it has been found empirically to be much faster than other methods while remaining just a reliable [3]. Additionally, direct gradient descent is also known empirically to not always solve the problem. Unsurprisingly, in behind-the-scenes experiments it was found that direct gradient descent in continuous time on p did not work satisfactorily either. The natural extension of Guass-Newton iteration is the Newton-Raphson flow as explained by Professor Sean Meyn in his new textbook [4]. In our case it is given by

$$\frac{dp}{dt} \coloneqq - \left[\frac{\partial^2 J}{\partial p^2}^T \frac{\partial^2 J}{\partial p^2} \right]^{-1} \frac{\partial J}{\partial p} = - \left[\left(\nabla_x I \frac{\partial W}{\partial p} \right)^T \left(\nabla_x I \frac{\partial W}{\partial p} \right) \right]^{-1} \iint_{X \in M} \left(\nabla_x I \frac{\partial W}{\partial p} \right)^T \left[I - I_T \right] dX \tag{12}$$

.

2.1.3 The relation between p and T

The system identification and observer design that follows will attempt to directly determine T and the planes parameter d. Thus, if the Newton-Raphson flow is to be used on p then $\dot{T}$ and $\dot{d}$ must be determined in terms of $\dot{p}$ since p constrains T and d. Referring back to the definition of p (5),

$$\begin{aligned}\dot{p}_1(t) &\coloneqq \frac{d}{dt}\frac{d}{d - T_3(t)} \\ \dot{p}_2(t) &\coloneqq \frac{d}{dt}\frac{-T_1(t)}{d - T_3(t)} \\ \dot{p}_3(t) &\coloneqq \frac{d}{dt}\frac{-T_2(t)}{d - T_3(t)}\end{aligned} \tag{13}$$

Some manipulation reveals

$$\begin{aligned}\dot{d}p_1 + d\dot{p}_1 - \dot{T}_3 p_1 - T_3\dot{p}_1 &= \dot{d} \\ \dot{d}p_2 + d\dot{p}_2 - \dot{T}_3 p_2 - T_3\dot{p}_2 &= -\dot{T}_1 \\ \dot{d}p_3 + d\dot{p}_3 - \dot{T}_3 p_3 - T_3\dot{p}_3 &= -\dot{T}_2\end{aligned} \tag{14}$$

.

This is a linear algebra problem, thus,

$$\dot{p} = \frac{1}{d - T_3}\begin{bmatrix} 1 - p_1 & 0 & 0 & p_1 \\ -p_2 & -1 & 0 & p_2 \\ -p_3 & 0 & -1 & p_3 \end{bmatrix}\begin{bmatrix}\dot{d} \\ \dot{T}\end{bmatrix} \tag{15}$$

.

Clearly, $\dot{p}$ does not uniquely determine $\dot{d}$ and $\dot{T}$. Indeed three things cannot uniquely determine four things. Further, it is well-known that minimizing quanitites in the imaging space cannot uniquely determine the scale of depth. An additional constraint is needed to uniquely determine $\dot{d}$ and $\dot{T}$. Further, some assurance is needed that this constraint will not cause d and T to converge to their true values.

2.1.4 Continuous time regression using acceleration measurements

If an inertial sensor is mounted to the camera, then it can be assumed $\ddot{T}$ is known. Technically in practice only $\ddot{T} + g$ where g is the gravitational acceleration in the camera frame is known. However, for the purposes of this project g is neglected. Previous work that is currently under review suggests that g should also be able to be determined with a few extensions to the theory below.

If this project were not restricted to a continuous time dynamical system, then would be easy to optimize the following constraint between $\ddot{T}_3$ and p_1

$$\begin{aligned} & T_3(t) = T_3(t_0) + \dot{T}_3(t_0)t + \int_{t_0}^{t}\int_{t_0}^{\sigma}\ddot{T}_3(\sigma_2)d\sigma_2 d\sigma \\ \iff & d\frac{d - T_3(t)}{d} = d\frac{d - T_3(t_0)}{d} - \dot{T}_3(t_0)t - \iint \ddot{T}_3(\sigma_2)d\sigma_2 d\sigma \\ \iff & p_1(t)d = p_1(t_0)d - \dot{T}_3(t_0)t - \iint \ddot{T}_3(\sigma_2)d\sigma_2 d\sigma \end{aligned} \tag{16}$$

Due to the double integral of $\ddot{T}_3$, which is measured value and associated with noise, in practice it is important to consider a finite window of previous measurement history. Thus, to optimize the above constraint in a continuous time estimator, one would need to calculate a suitable norm (say the 2-norm) over a window of time and set $\dot{d}$ to the negative gradient of that cost with respect to d. This can be done, but it is not easy to do in a continuous time. A more elegant solution comes from considering the deterministic identification problem using regressor vectors. The derivation proceeds in a manner similar to the method presented in-class.

Consider,

$$\begin{aligned} p_1 &= \frac{d}{d - T_3} \\ \iff d\frac{d^2}{dt^2}\left(\frac{1}{p_1} - 1\right) &= -\ddot{T}_3 \end{aligned} \tag{17}$$

.

For convenience, define $m \coloneqq \frac{1}{p_1} - 1$. Then, taking the laplace transform reveals

$$s^2 dm(s) = -\ddot{T}(s) \tag{18}$$

.

If both sides are multiplied by the stable transfer function given by $1/(s+\lambda)^2, \lambda > 0$, then

$$\begin{aligned} d\frac{s^2}{(s+\lambda)^2}m(s) &= \frac{-1}{(s+\lambda)^2}\ddot{T}_3(s) \\ \iff d\frac{(s+\lambda)^2 - 2\lambda s - \lambda^2}{(s+\lambda)^2}m(s) &= \frac{-1}{(s+\lambda)^2}\ddot{T}_3(s) \\ \iff dm(s) &= \frac{-1}{(s+\lambda)^2}T(s) + \frac{2\lambda s + \lambda^2}{(s+\lambda)^2}m(s) \end{aligned} \tag{19}$$

.

Letting the impulse response of $(2\lambda s + \lambda^2)/(s+\lambda)^2$ be given by h_1 and $1/(s+\lambda)^2$ by h_2. Then assuming all initial conditions are zero

$$0 = d\left[m(t) - (h_1 * m)(t)\right] + (h_2 * \ddot{T}_3)(t) \tag{20}$$

.

Where $*$ denotes convolution in time. Thus, a constraint between the measured quantity $\ddot{T}_3$ and d has been found. An instantaneous cost on this constraint can be defined as

$$J_a(t,d) \coloneqq \frac{1}{2}\left(d\left[m(t) - (h_1 * m)(t)\right] + (h_2 * \ddot{T}_3)(t)\right)^2 \tag{21}$$

.

Then setting $\dot{d}$ to the negative gradient of J_a would be performing gradient descent on d so that the "acceleration" of p_1 is scaled to the "acceleration" measured $\ddot{T}_3$. The low-pass filters are essentially taking the place of the norm over a window of time considered earlier.

Thus, the resulting update rule for $\dot{d}$ has been found

$$\dot{d} = -\frac{dJ_a}{dd} = -\Big(d\left[m(t) - (h_1 * m)(t)\right] + (h_2 * \ddot{T}_3)(t)\Big) * \left[m(t) - (h_1 * m)(t)\right] \tag{22}$$

.

2.1.5 $\dot{T}$ and $\dot{d}$ are now determined by the losses

Considering, everything above, (15) provides a method to optimize the brightness constraint with Newton-Raphson flow through $\dot{T}$ and $\dot{d}$. However, that constraint did not uniquely determine $\dot{T}$ and $\dot{d}$. So, by further considering p_1 and $\ddot{T}_3$ another constraint was found revealing:

$$\begin{bmatrix} -\frac{dJ_a}{dd} \\ \dot{p} \end{bmatrix} = \underbrace{\frac{1}{d - T_3}\begin{bmatrix} -(d - T_3) & 0 & 0 & 0 \\ 1 - p_1 & 0 & 0 & p_1 \\ -p_2 & -1 & 0 & p_2 \\ -p_3 & 0 & -1 & p_3 \end{bmatrix}}_{G^{-1}(d,T)} \begin{bmatrix} \dot{d} \\ \dot{T} \end{bmatrix} \tag{23}$$

It is easy to verify that the above matrix is always full rank and invertible because p_1 cannot equal zero (that would require T_3 to be infinite). Thus, $\dot{T}$ and $\dot{d}$ are uniquely determined by Newton-Raphson flow on the brightness constraint and gradient descent on the regression between d, $\ddot{T}_3$, and p_1. Call this matrix G^{-1}.

2.2 The observer

The previous discussion really only considered the static case (where the camera is not moving). Thus, a larger framework is needed to integrate the above constraints in a method that adjusts $\hat{T}$ and $\hat{d}$ such that $\hat{T} \to T$ and $\hat{d} \to d$. A straightforward method do so is inspired by a Luenberger observer.

Let the "corrections" from the continuous time Lucas-Kanade and continuous-time regression be given by

$$\frac{d}{dt}\begin{bmatrix} \hat{d} \\ \hat{T} \\ \hat{\dot{T}} \end{bmatrix} = \begin{bmatrix} 0 \\ \hat{\dot{T}} \\ \ddot{T} \end{bmatrix} + LG(\hat{d}, \hat{T}) \begin{bmatrix} -\frac{dJ_a}{dd} \\ \dot{p} \end{bmatrix} \tag{24}$$

Clearly L is the classical observer gain. Additionally, while G and everything to the right of it are highly nonlinear, their purpose is simply to match the template image I_t with the incoming image I while scaling d so that T_3/d matches the measured acceleration. Considered this way, $G(d,T)\begin{bmatrix} -\frac{dJ_a}{dd} & \dot{p} \end{bmatrix}^T$ is actually a descent direction on the observed output error in the same way that outputs of a linear system, $y - \hat{y}$, provide descent direction.

Since the motion along the X, Y axis is zero, a more specific form of the proposed observer is given by

$$\frac{d}{dt}\begin{bmatrix} \hat{d} \\ \hat{T} \\ \hat{\dot{T}}_3 \end{bmatrix} = \begin{bmatrix} 0 \\ \begin{bmatrix} 0 \\ 0 \\ \hat{\dot{T}}_3 \end{bmatrix} \\ \ddot{T}_3 \end{bmatrix} + LG(d,T) \begin{bmatrix} -\frac{dJ_a}{dd} \\ \dot{p} \end{bmatrix}, \quad L = \begin{bmatrix} l_1 & 0 & 0 & 0 \\ 0 & l_2 & 0 & 0 \\ 0 & 0 & l_2 & 0 \\ 0 & 0 & 0 & l_2 \\ 0 & l_3 & 0 & 0 \end{bmatrix} \tag{25}$$

Such a structure mimics the Luenberger observer for a two dimensional integrator when feedback is available only from the highest level state. In that case the second column of L, without the zero entries, would be the Luenberger gain.

2.3 Some comments on stability

Global stability of the observer is out of the question. This is because the Lucas-Kanade tracker will only work if the parameters p resulting from T and d result in "looking at" at a position in the incoming image that is close to the template. This is easily verified in simulation.

Further, stability is dependent on the property of the brightness intensity function I. Clearly, I must be "sufficiently rich" for $\dot{p}$ to result in descent direction for $\hat{T}$ and $\hat{d}$. For example, if the planar patch is a single color, then $\dot{p} = 0$, $\forall t$ and there is no hope of convergence.

Additionally, stability is dependent on the excitation provided to the system. If acceleration is zero, then continuous time-regression has no-hope of finding d. However, as shown in recent work under review for a related problem, the only technical condition on acceleration was that it be non-zero. A similar relation should hold for this problem since the purpose of the continuous time regression is simply to scale d so that the curvature of T_3 matches the measured acceleration. As long as there is acceleration, there should be curvature to be matched.

Thus, there remains hope for proving some kind of stability, but it will have to be the subject of future work. Below, some basic ideas that if followed up on, would likely result in a sufficient condition for stability are presented.

Let the error $e(t)$ be defined as

$$e(t) \coloneqq \begin{bmatrix} d - \hat{d} \\ T - \hat{T} \\ \dot{T} - \hat{\dot{T}} \end{bmatrix} \tag{26}$$

.

Then considering the candidate LaSalle function

$$V(e(t)) = \frac{1}{2}\|e\|_2^2 \tag{27}$$

Then it is easy to see that $\dot{V}$ is given by

$$\dot{V} = e^T \dot{e} = e^T A e - e^T L G(\hat{d}, \hat{T}) \begin{bmatrix} -\frac{dJ_a}{dd} \\ \dot{p} \end{bmatrix}, \quad A \coloneqq \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{28}$$

Clearly, A is not Hurwitz. Regardless, in this form, it begins to look tractable to determine the stability properties of the entire system as well as the required hypothesis. One hypothesis that is likely to be needed is as follows.

Suppose the camera is initially looking at a sufficiently rich scene patch M such that the Lucas-Kanade tracker would converge if the camera is not moving. Rigorously this would mean:

$$\exists \delta \text{ s.t. } \forall \| p - \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \| < \delta, p \rightarrow \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \text{ as } t \rightarrow \infty \tag{29}$$

Then, in the case where the camera is moving, it stands to reason that if the camera has bounded velocity, and the trajectory is bounded away from the planar patch being viewed, then setting a high enough gain to the calculation of $\dot{p}$ will result in p remaining within the required neighborhood.

This restriction on p will restrict the space $\hat{T}$ and $\hat{d}$ live on to a neighborhood around one dimensional manifold. Further, since $\hat{T}$ and $\hat{d}$ are related by a a differentiable function (so long as T_3 is bounded away from d). It seems reasonable to expect that the continuous-time regression would then guide $\hat{T}$ and $\hat{d}$ to T and d respectively.

2.4 Experimental Evaluation

Even though a proof of stability was missing, it seemed wise to use simulation to determine if there was any hope of good results before investing significant time in a proof. To do so, the proposed observer was evaluated using Simulink. To simulate a "continuous time camera" (which does not exist on today's market), the trajectory $T_1 = 0$, $T_2 = 0$, $T_3(t) = 0.25 * \sin(t)$ and plane parameters $d = 1$ were used to continuously warp a single greyscale real-world image into what would be seen by a camera moving along that trajectory. The image was resized to 320x240 pixels and blurred with a 2D Gaussian filter with a standard deviation of 4 pixels.

The incoming image's gradients were approximated by the popular Sobel operator [8]. For clarity, the operation is repeated here. The kernels are assume to have a height and width of three pixels.

$$\nabla_x I \approx \frac{1}{8} \left[\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * I \quad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I \right] \tag{30}$$

The continuous time Lucas-Kanade tracker was configured to track a 57x41 pixel patch with top-left corner at pixel location 166x72 as illustrated by Figure 1. The spatial integral required for calculating the Lucas-Kanade correction vector was calculated discretely by summing over all the pixels in the patch.

The continuous time regressions transfer function parameter λ was set to 2π. The output of the continuous-time regression was not used for 2 seconds in order to minimize the transient effects of mis-match in the initial conditions.

The observer gains were set to $l_1 = 10,000$, $l_2 = 25$, and $l_3 = 2.5$.

The initial conditions for the observer were perturbed from their true values. The true initial condition was $d = 1$, $T = 0, \dot{T} = 0$. However, the initial condition of the observer were set to $\hat{d} = 1.1$, $\hat{T} = [-35, -10, -0.2]$, and $\dot{\hat{T}} = 0$.

The "ode45" solver was used with default parameters, except for configuring the max step size to 0.01 seconds, to simulate results over a 30 second interval. Configuring the step size to something "small", like 0.01 seconds, was necessary to get consistent numerical results.

3 Results

Figure 2 shows the results from the Simulink simulation. All three top-level states T quickly converge to be very close to their true values. This is due to the continuous-time Lucas-Kanade tracker quickly converging to "center" the incoming image on top of the template image. This process is visualized in Figure 3. There the initial incoming image, the template image and the difference is shown at time $t = 0$ and $t = 0.34$ as well as Lucas-Kanade tracker's loss J.

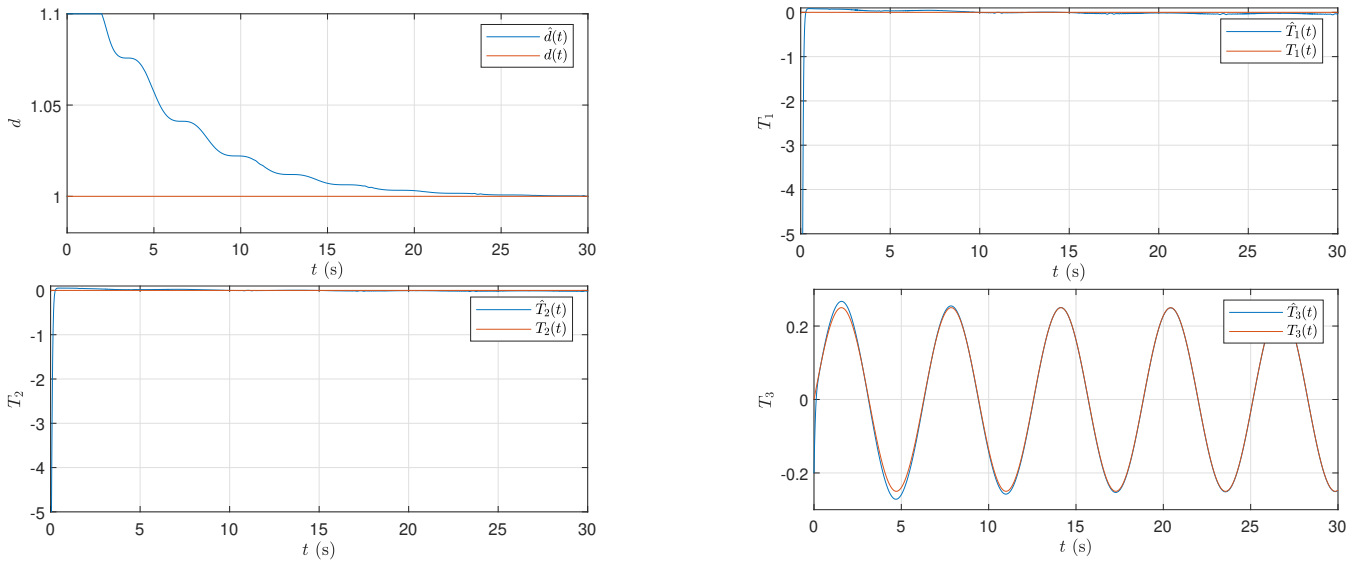

Fig. 2. The above plots show the convergence of $\hat{d}$ and $\hat{T}$ to the true values. Note that T_3 is the only time varying real quantity. The plot of $\hat{\dot{T}}_3$ is not included since it is known to converge due to convergence of T_3 and measurements of $\ddot{T}_3$.

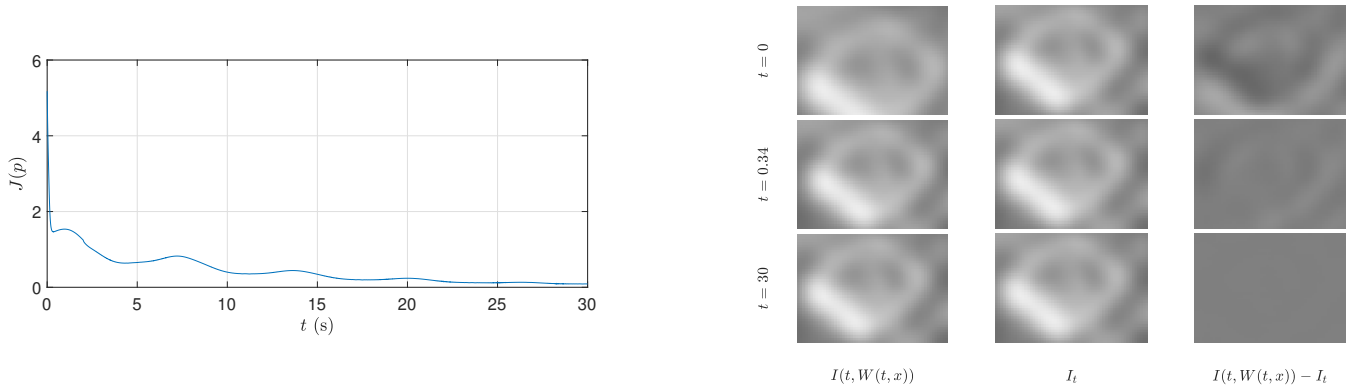

Fig. 3. (left) The LK trackers error summed over the template image. (right) The warped incoming image considered at three critical times and the difference between that and the template. The three critical times are the the initial time, the initial convergence to a local minima, and the final time. The difference images black levels have been shifted to grey so that both positive and negative errors can be visualized.

However, there is a constant multiplicative offset that slowly converges to 1 as d converges to the true value. The system parameter d, converges much more slowly than the other states. This is because it was necessary to use a very large gain (10,000) to get the convergence pictures. Higher values could result in instability. It was noticed that a very small step max timestep (0.01 seconds) was necessary to get stable results for the trajectory of $\hat{d}$.

4 Discussion

First the experimental results are discussed. As expected, the continuous time Lucas-Kanade tracker quickly converged to some local minima within approximately 0.34 seconds. This minima greatly improved the match between the incoming image and the template image as can be seen in Figure 3. Further, as indicated by the plots in Figure 2 it appears this convergence restricted $\hat{d}$ and $\hat{T}$ to some neighborhood around the true values immediately. Then, the continuous time regression for the planes parameter d slowly began to converge. As it converged, all three elements of $\hat{T}$ (particularly T_3 which is the estimate of Z position) continued to converge to their true values.

Because the Lucas-Kanade tracker is tracking a moving patch, it cannot "keep up" on its own until the planes parameter d becomes known. As a result, the slight mismatch between the incoming image and the template slowly

decreased in a manner qualitatively similar to the convergence of $\hat{d}$. Notice the pattern of "bumps" in J is very similar to the bumps in $\hat{d}$'s trajectory.

It should be noted that the template image shown in Figure 3 is "super blurry" compared to the original photo showed in Figure 1. Blurring this much was necessary to prevent the differential equations from becoming "too stiff" and taking excessively long to solve. In computer vision problems, it is not typically necessary to perform so much blurring to get a working system. However, blurring and using multi-scale pyramids is a common method for ensuring better convergence of LK trackers [8]. In this case, it would be of great interest to investigate this further to understand if this stringent "blurring" requirement is fundamental or simply required for the numerical methods.

While the Newton-Raphson flow was introduced in (12) for solving the brightness constraint in a manner analogous to Gauss-Newton iteration, it does not seem strictly necessary. In practice Gauss-Newton is known to perform well for the Lucas-Kanade problem [3], and in my own experiments gradient descent never worked at all. Both in the discrete time case in some previous work and some experiments for this project that took place behind the scenes. Understanding why regular gradient descent performs so poorly would be an interesting problem.

The continuous-time regression used to find d should also be able to easily extended to solve for d using motion along the X or Y axis by considering p_2 and p_3 respectively. Developing this would be a trivially simple extension.

In the end, all the numerical result match the predictions made in the "Comments on Stability" subsection. Because of this it seems likely that pursuing a proof of stability starting from (28), and including the hypothesis just below, will be a fruitful, or at least insightful direction forward. Further, the proof for the more general case of a tilted plane is likely to be very similar. Thus, the most pressing direction for future work is the stability proof.

One additional direction for future work, if a proof of stability is found, is to pursue the problem where the camera's velocity is bounded, but the bound is not known a-priori. It stands to reason that an "adaptive" Lucas-Kanade gain would help to solve such a problem. Unfortunately, this idea was thought of just a few days before the report became due, so it could not be pursued further.

5 Conclusions

A mathematical framework for a continuous time Lucas-Kanade tracker has been developed for the simplified case of a planar scene, parallel to the camera's imaging plane, and motion along the camera's optical axis. The primary development is to exchange Gauss-Newton iteration for Newton-Raphson flow. Additionally, continuous-time regression is used to regress the planes single parameter d. The results are combined in a Luenberger style observer. The method was validated in a simulation, however additional work is needed to understand sufficient conditions for stability. It is thought that proving stability in this simplified case will be insightful for the general case, and it is recommended that this direction be pursued prior to extending the above results to general motion of the camera or slanted patches.

Acknowledgements

This research was supported by NSF award DGE-1632976.

References

[1] B. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *Proceedings DARPA Image Understanding Workshop*, pp. 121–130, 1981.

[2] C. Tomasi and T. Kanade, "Detection and Tracking of Point Features," tech. rep., Carnegie Mellon University, 1991.

[3] S. Baker and I. Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework," Tech. Rep. 3, 2004.

[4] S. Meyn, *Control Systems & Reinforcement Learning*. Cambridge University Press, to appear ed., 2021.

[5] M. M. Sondhi and D. Mitra, "New Results on the Performance of a Well-Known Class of Adaptive Filters," *Proceedings of the IEEE*, vol. 64, no. 11, pp. 1583–1597, 1976.

[6] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, 2nd ed., 1995.

[7] K. J. Åström and B. Wittenmark, *Adaptive control*. Courier Corporation, 2013.

[8] R. Szeliski, *Computer Vision: Algorithms and Implementations*. Springer, 2010.

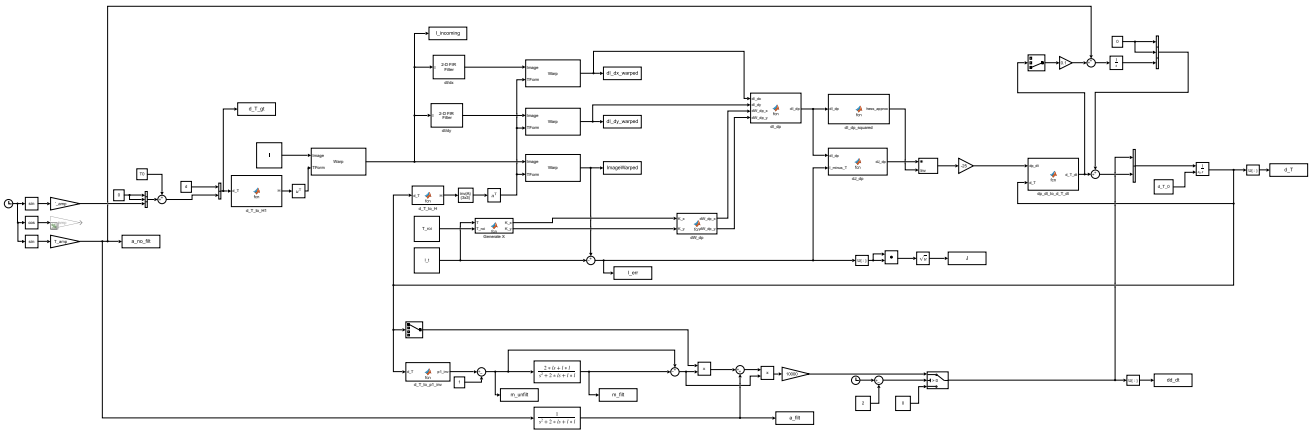

Fig. 4. The entire Simulink block diagram. It is a vector file so zooming in should work.

6 Appendix 1

For transparency, the complete block diagram model from Simulink, rendered in a vector format, is shown in Figure 4. The code listings for each Matlab function block in the simulation are also included.

6.1 Code listings for Matlab scripts and Matlab function blocks

6.1.1 Homography3params_move_2.m

```
clear
close all

I = double(rgb2gray(imread('test_small.jpg'))) / 255;
I = imgaussfilt(I,4);

T_roi = floor([666, 289, 896-666, 453-289]/4);
I_t = I(T_roi(2):T_roi(2)+T_roi(4)-1, T_roi(1):T_roi(1)+T_roi(3)-1);

% Plane initially at Z=1 meter
d = 1.0; % Plane is defined by Z = 1
% Camera moves sinusoidally on the Z axis from the origin
T0 = [0 0 0];
% Moves with this amplitude
T_amp = 0.25;

% Perturb the initial state estimate from the true value
d_T_0 = [1.1 -35 -10 -0.2];
%d_0 = d_T_0(1);
%T_0 = d_T_0(2:4);

% regressor filters frequency
l = 2*pi;

%sim('Homography3params_move_sim_2');
```

6.1.2 d_T_to_p.m

```
function p = d_T_to_p(d,T)
    p = [d/(d-T(3)),...
         -T(1)/(d-T(3)),...
         -T(2)/(d-T(3))];
end
```

6.1.3 d_T_to_H

```
function H = fcn(d_T)

p = d_T_to_p(d_T(1), d_T(2:4));

H = [p(1), 0, p(2);...
     0, p(1), p(3);...
     0, 0, 1];
```

6.1.4 Generate_X

```
function [K_x, K_y]= fcn(T, T_roi)

shape_T = size(T);

X = [1:shape_T(2)] + T_roi(1);
Y = [1:shape_T(1)] + T_roi(2);
[X, Y] = meshgrid(X,Y);

K_x = X;
K_y = Y;
```

6.1.5 dW_dp

```
function [dW_dp_x, dW_dp_y]= fcn(K_x, K_y)

dW_dp_x = zeros([size(K_x) 3]);
dW_dp_x(:, :, 1) = K_x;
dW_dp_x(:, :, 2) = 1;

dW_dp_y = zeros([size(K_y) 3]);
dW_dp_y(:, :, 1) = K_y;
dW_dp_y(:, :, 3) = 1;
```

6.1.6 dI_dp

```
function dI_dp = fcn(dI_dx, dI_dy, dW_dp_x, dW_dp_y)

dI_dp = dI_dx .* dW_dp_x + dI_dy .* dW_dp_y;
```

6.1.7 dI_dp_squared

```
function hess_approx = fcn(dI_dp)
num_p = 3;
hess_approx = zeros(num_p, num_p);
for i = 1:num_p
    for j = 1:num_p
        hess_approx(i, j) = sum(dI_dp(:, :, i).*dI_dp(:, :, j), 'all');
    end
end
```

6.1.8 dJ_dp

```
function dJ_dp = fcn(dI_dp, I_minus_T)

dJ_dp = reshape(sum(I_minus_T .* dI_dp, [1,2]), [1, 3]);
```

6.1.9 dp_dt_to_d_T_dt

```
function d_T_dt = fcn(dp_dt, d_T)

p = d_T_to_p(d_T(1), d_T(2:4));

d = d_T(1);
T1 = d_T(2);
T2 = d_T(3);
T3 = d_T(4);

A = [ 0  0 p(1);...
     -1  0 p(2);...
      0 -1 p(3)] / (d-T3);

b = [dp_dt(1); ...
     dp_dt(2); ...
     dp_dt(3)];

d_T_dt = linsolve(A, b);
```

6.1.10 d_T_to_p1_inv

```
function p1_inv = fcn(d_T)

p1_inv = (d_T(1) - d_T(4)) / d_T(1);
```