53rd IEEE Conference on Decision and Control
December 15-17, 2014. Los Angeles, California, USA

Randomized Algorithm for Estimation of Moving Point Position Using Single Camera

Dmitry Krivokon[1], Alexander Vakhitov[1]

Abstract— **Stochastic approximation algorithms (for example SPSA) provide a way to solve optimization problems in the presence of arbitrary but bounded disturbances. In this paper a problem of position estimation for a moving point using monocular projective observations is considered. We add random perturbations to camera position to produce an algorithm which makes estimates of point position demanding only that the point's velocity is bounded in time. This is superior to the methods currently available in the computer vision field which all consider very restricted cases of point movement (constant, movement in plane). We prove theoretical convergence of estimates and provide numerical simulation for the algorithm.**

I. Introduction

There are large number of applications where estimation of a moving point position is essential. Driver assistance systems rely on accurate knowledge of positions of pedestrians and other vehicles to prevent possible collisions [1]. Robotics systems require algorithms for reconstruction of surroundings including dynamic objects [2], [3] for successful navigation and obstacle avoidance. In computer vision this problem is generally called simultaneous localization and mapping (SLAM). There are a lot of well-known algorithms developed in the field [4], [5], [2], [3], [1]. For the most part ones that are based on using single camera consider only static scene and merely filter out moving object. Dynamic case is generally dealt either by augmenting observational system with special kinds of sensors like radars or multiple cameras or by heavily restricting object movement. First approach is not really interesting for us because use of additional devices can be impractical and too expensive for many applications. Lets consider options available in the second approach. In [6] authors propose a solution for the case of an object moving with constant speed. Method presented in [7] copes with a case of objects moving in plane. Both possibilities greatly decrease practical applicability of reconstruction systems and allows their usage only in some narrow field of problems. We on the other hand propose an algorithm that allows estimation of a moving point position using single camera in the conditions of arbitrary movement demanding only that that movement is bounded in time.

We propose use of special types of camera motion and a filtering algorithm for the point's position estimation to give the monocular SLAM system a possibility to estimate depth of any kind of point (dynamic or static) moving with arbitrary trajectory. To do this, the theoretically grounded solution would be to use random motion of the camera, where the direction of motion will be independent from the observations of other objects' trajectories. This motion can for example be programmed and known, or it can be estimated using objects which are known to be static (because with our approach we can definitely distinguish between moving and static objects). The idea of random motion is inspired by an approach to linear plant identification using random perturbation [8]. It leads to a simple filtering algorithm with constant step size using a gradient approximation as in SPSA algorithms [9], [10]. This algorithm is also similar to a more general case described in [11], [12]. We analyze theoretically convergence properties of this stochastic filtering algorithm, estimate its mean error and show on simulations how the algorithm performs.

In the next section, we will briefly describe a problem and sketch our approach to it.

II. Problem Description

In our problem we consider a single moving camera. It's position and hence speed in each moment of time is supposed to be known. We also assume that we can control its movement to some extent, specifically we assume that we can add some perturbations to its positions. To make our assumptions more clear lets consider a case of a moving vehicle on which a camera is installed. Vehicle has its own speed and hence the camera too, we assume that this speed is present and we can't change it. Let's imagine now that the camera is set on some mechanical platform that allows its local movement, we control position of the camera inside this platform and based on that we simulate perturbations for our method. So in the end the camera has its own velocity, which is produced by a moving car, and velocity which is produced by our random perturbations. Later on in formal description of a problem we will separate these two types of velocities to make derivation more clear. Our camera observes a single point which can be static or moving. We don't restrict its movement to special cases as was discussed in introduction. It is assumed that projections of a point on a camera frame at each moment of time are known. Projections can be calculated using well known optical flow methods or explicit matching techniques based on distance in some feature space. Because of the nature of these projections they can contain some amount of noise which in general case has unknown characteristics. In such conditions a good solution

[1]Dmitry Krivokon and Alexander Vakhitov are with Faculty of Mathematics and Mechanics, St. Petersburg State University, 198504, Universitetskii pr. 28, St. Petersburg, Russia, Dmitry Krivikon is also with the Herzen State Pedagogical University of Russia. dmitry00@gmail.com, a.vakhitov@spbu.ru. The work was supported by RFBR (project 13-07-00250). The authors acknowledge the Russian Ministry of Education and Science (project 27.1835.2014/K).

for the problem would be some kind randomized stochastic optimization method [9], [10]. We will now give a short introduction to the main idea of our method.

A. Algorithm idea

As mentioned before our method is based on perturbation of a camera position. For illustrative purposes in this section we will consider initially static camera to which we add known random perturbation. Also, we will here restrict our problem only for simplicity to the problem of estimation of point's depth only. In some applications even this will be sufficient because ray which goes from optical center to the point can be reconstructed based on this point's projection on camera frame and from this ray and estimated depth of a point one can easily calculate its position. This section presents only rough idea on which our algorithm is based. Formal and consistent description will be presented in later sections.

Let's consider a simple of a camera with position $C_n = (\Delta_n^{(1)}, 0, 0)^T$ where $\Delta_n^{(1)}$ is our random perturbation. This camera observes a point $(X_n, Y_n, Z_n)^T$ and at each moment of time projection $(p_n^x, p_n^y)^T$ of this point on camera frame is present, such as

$$p_n^x = \frac{X_n - \Delta_n^{(1)}}{Z_n} + v_n^x,$$

$$p_n^y = \frac{Y_n}{Z_n} + v_n^y.$$

where (v_n^x, v_n^y) is unknown noise vector. Now lets take equation for p_n^x and multiply it by $\Delta_n^{(1)}$:

$$p_n^x \Delta_n^{(1)} = \frac{X_n \Delta_n^{(1)}}{Z_n} - \frac{\Delta_n^{(1)} \Delta_n^{(1)}}{Z_n} + v_n^x \Delta_n^{(1)}$$

As in most of SPSA-like algorithms we will generate $\Delta_n^{(1)}$ in such way that it will have zero expected value. And because we generate it randomly we can safely assume that it is independent of points's position and noise in projections. Having this lets take expected value of the above equation:

$$E\langle p_n^x \Delta_n^{(1)} \rangle = E\langle \frac{X_n \Delta_n^{(1)}}{Z_n} - \frac{\Delta_n^{(1)} \Delta_n^{(1)}}{Z_n} + v_n^x \Delta_n^{(1)} \rangle.$$

Because of the nature of $\Delta_n^{(1)}$ we have $E\langle \frac{X_n \Delta_n^{(1)}}{Z_n} \rangle = 0$ and $E\langle v_n^x \Delta_n^{(1)} \rangle = 0$ so now we can simplify everything to this:

$$E\langle p_n^x \Delta_n^{(1)} \rangle = E\langle \frac{\Delta_n^{(1)} \Delta_n^{(1)}}{Z_n} \rangle,$$

$$E\langle p_n^x \Delta_n^{(1)} \rangle = \frac{\Sigma_\Delta}{E Z_n},$$

where $\Sigma_\Delta = E\langle \Delta_n^{(1)} \Delta_n^{(1)} \rangle$. From last equation we can roughly conclude that if we average $p_n^x \Delta_n^{(1)}$ over time we will get inverse depth multiplied by some constant and hence we can estimate it. This derivation is intuitive basis of our algorithm. Having this we construct our method by considering a dynamical system where the observation is constructed from the point's projection at the current moment of time and some expression of the form $p_n^x \Delta_n^{(1)}$. This additional coordinate in the observation vector represents inverse depth. Then we apply a simple fixed-step gradient-like procedure to make estimates of system's state.

In next section a formal and elaborate description of our method will be presented alongside with a proof of convergence of the estimates for the gradient-like procedure.

III. Algorithm Formulation

A. Notation

We consider a dynamical system with a state $\theta_n \in \mathbb{R}^3$, where $n \in \mathbb{N}$ is discrete moment of time. $\theta_n^{(1)}, \theta_n^{(2)}$ are coordinates on the image plane of the intersection of the ray to from the camera center to the point with the image plane. $\theta_n^{(3)}$ is inverse depth of the point. If the euclidean coordinates of the point at the moment n in the camera frame are X_n, Y_n, Z_n, then $\theta_n^{(1)} = X_n/Z_n$, $\theta_n^{(2)} = Y_n/Z_n$, $\theta_n^{(3)} = 1/Z_n$. Inverse conversion is obvious and using it we can calculate object position.

The coordinate frame is defined by the camera. The point is supposed to be drifting freely in 3D space, so the point coordinates change due to the point motion can be denoted as $\xi = (\xi_n^{(1)}\ \xi_n^{(2)}\ \xi_n^{(3)})^T$. In the same time, at each moment, the camera, not changing its orientation, makes a move $\Delta_n = (\Delta_n^{(1)}\ \Delta_n^{(2)}\ \Delta_n^{(3)})^T$. Finally, we can formulate the evolution of point coordinates as

$$\begin{pmatrix} X_n \\ Y_n \\ Z_n \end{pmatrix} = \begin{pmatrix} X_{n-1} \\ Y_{n-1} \\ Z_{n-1} \end{pmatrix} + \Delta_n + \xi_n.$$

B. Algorithm

We assume following to be true for all $n \in \mathbb{N}$:

1) Point is always in the screen, so there exist constants W, H such that $|\frac{X_n}{Z_n}| < W$, $|\frac{Y_n}{Z_n}| < H$.
2) Point's depth lies in some bounded interval: $Z_n \in (Z_m, Z_M)$, and $Z_m, Z_M > 0$.
3) Point drift is bounded $|\xi_n^{(1)}| < D_1$, $|\xi_n^{(2)}| < D_2$, $|\xi_n^{(3)}| < D_3$.
4) Random camera motion has finite moments: $E|\Delta_n^{(i)}|^j < \delta_i^j$, for $i = 1 \ldots 3,\ j = 1, 2$.
5) Additive image noise has variances:

$$E|v_n^{(1)}|^2 < \sigma_{v1}^2,\ E|v_n^{(2)}|^2 < \sigma_{v2}^2,\ E|v_n^{(1)} - v_{n-1}^{(1)}|^2 < \sigma_d^2.$$

The following sequence of actions is proposed:

- do a random camera motion Δ_n
- do measurements (coordinates of the point's projection and the special third component for the depth estimation constructed by us) y_n
- update the estimates of point's true projection on camera and its inverse depth (the formula below)

We denote by y_n the vector of observations which are made at each step. Basic algorithm is of the form

$$\hat{\theta}_{n+1} = \hat{\theta}_n - \alpha(\hat{\theta}_n - y_n),$$

where $\alpha > 0$ is a step size coefficient.

We propose to do observations as follows:

$$y_n^{(1)} = \frac{X_{n-1} + \Delta_n^{(1)} + \xi_n^{(1)}}{Z_{n-1} + \Delta_n^{(3)} + \xi_n^{(3)}} + v_n^{(1)},$$

$$y_n^{(2)} = \frac{Y_{n-1} + \Delta_n^{(2)} + \xi_n^{(1)}}{Z_{n-1} + \Delta_n^{(3)} + \xi_n^{(3)}} + v_n^{(2)},$$

$$y_n^{(3)} = k_n \Delta_n^{(1)} \left(\frac{X_{n-1} + \Delta_n^{(1)}}{Z_{n-1} + \Delta_n^{(3)} + \xi_n^{(3)}} - \frac{X_{n-1}}{Z_{n-1}} + v_n^{(1)} - v_{n-1}^{(1)} \right),$$

where $k_n = |\Delta_n^{(1)}|^{-2}$.

Next we will prove a theoretical bound on mean error of estimates produced by such algorithm.

IV. Convergence of Estimates

In the following theorem, denote

$$C_2 = \sqrt{18 Z_M^4 Z_m^{-4} + (3 Z_m^{-2}(Z_M^2 + 1)(W^2 + H^2) + 2)^2},$$

$$\bar{C}_3 = \sigma_{v1}^2 + \sigma_{v2}^2 + \sigma_d^2 + 12 Z_m^{-2}((W^2 + H^2)(\delta_3^2 + D_3^2) + $$
$$+ \delta_1^2 + \delta_2^2 + D_1^2 + D_2^2),$$

$$C_4 = Z_m^{-1}((W + H)(\delta_3^1 + D_3^1) + D_1^1 + D_2^1 + Z_m^{-1}(\delta_3^1 + D_3^1)).$$

Theorem If $\alpha, \varepsilon > 0$ are chosen such that

$$1 - 2\alpha + \alpha^2 C_2 + \frac{\varepsilon^2}{2} \in (0, 1)$$

then there is an asymptotic bound for the algorithm estimates:

$$E \lim_n \|\hat{\theta}_n - \theta_n\|^2 < \frac{0.5\varepsilon^{-2}(2 - 4\alpha)C_4 + \alpha^2 \bar{C}_3}{2\alpha - \alpha^2 C_2 - 0.5\varepsilon^2}$$

Proof. Let us denote

$$g_n = \hat{\theta}_n - y_n.$$

We will also denote as $E_n\{\cdot\}$ the conditional expectation with respect to observation history from the beginning of algotihm execution up to the $n-1$ st observation, including it and not including the n-th observation:

$$E_n\{\cdot\} = E\{\cdot | y_{n-1}, y_{n-2}, \ldots\}$$

The error on the $n+1$-st step can be represented as:

$$\|\hat{\theta}_{n+1} - \theta_{n+1}\|^2 = \|\hat{\theta}_n - \theta_n - \alpha g_n + \theta_n - \theta_{n+1}\|^2 = \|\hat{\theta}_n - \theta_n\|^2 +$$
$$+ \alpha^2 \|g_n\|^2 + \|\theta_n - \theta_{n+1}\|^2 + \langle \hat{\theta}_n - \theta_n, -\alpha g_n \rangle +$$
$$+ \langle \hat{\theta}_n - \theta_n, \theta_n - \theta_{n+1} \rangle + \langle -\alpha_n g_n, \theta_n - \theta_{n+1} \rangle.$$

Each term will be analysed in the following text. First of all, we make three important simplifications:

1) Because $E_n y_n = E_n \hat{\theta}_{n+1}$ and $E_n\{\hat{\theta}_n - \theta_n\} = \hat{\theta}_n - \theta_n$,

$$E_n \langle \hat{\theta}_n - \theta_n, \theta_n - y_n \rangle = \langle \hat{\theta}_n - \theta_n, \theta_n - \theta_{n+1} \rangle.$$

2) Using the definition $g_n = \hat{\theta}_n - y_n$

$$E_n \langle \hat{\theta}_n - \theta_n, g_n \rangle = \|\hat{\theta}_n - \theta_n\|^2 + \langle \hat{\theta}_n - \theta_n, \theta_n - y_n \rangle.$$

3) Using the same fact,

$$E_n \langle g_n, \theta_n - \theta_{n+1} \rangle = E_n \langle \hat{\theta}_n - \theta_{n+1}, \theta_n - \theta_{n+1} \rangle =$$
$$= E_n \langle \hat{\theta}_n - \theta_n, \theta_n - \theta_{n+1} \rangle + \|\theta_n - \theta_{n+1}\|^2.$$

Therefore,

$$E_n \|\hat{\theta}_{n+1} - \theta_{n+1}\|^2 = (1 - 2\alpha)\|\hat{\theta}_n - \theta_n\|^2 +$$
$$+ (1 - 2\alpha)\|\theta_n - \theta_{n+1}\|^2 + \alpha^2 \|g_n\|^2 +$$
$$+ (2 - 4\alpha)\langle \hat{\theta}_n - \theta_n, \theta_n - \theta_{n+1} \rangle.$$

We need to bound the three terms (second, third and fourth) in the last formula.

1) $\|\theta_n - \theta_{n+1}\|^2$:

$$|\theta_n^{(1)} - \theta_{n+1}^{(1)}|^2 = \left(\frac{X_n}{Z_n} - \frac{X_n + \xi_n^{(1)} + \Delta_n^{(1)}}{Z_n + \xi_n^{(3)} + \Delta_n^{(3)}} \right)^2 =$$
$$= \left(\frac{X_n \xi_n^{(3)} + X_n \Delta_n^{(3)} - Z_n \xi_n^{(1)} - Z_n \Delta_n^{(1)}}{Z_n (Z_n + \xi_n^{(3)} + \Delta_n^{(3)})} \right)^2.$$

Because of independence of $\Delta_n^{(1)}$ and other random values appearing in the formula,

$$E_n |\theta_n^{(1)} - \theta_{n+1}^{(1)}|^2 \le \frac{\delta_1^2}{Z_m^2} + \frac{1}{Z_m}((W D_3 + D_1)^2 + W^2 \delta_3^2).$$

Analogously,

$$E_n |\theta_n^{(2)} - \theta_{n+1}^{(2)}|^2 \le \frac{\delta_2^2}{Z_m^2} + \frac{1}{Z_m}((H D_3 + D_2)^2 + H^2 \delta_3^2).$$

$$E_n |\theta_n^{(3)} - \theta_{n+1}^{(3)}|^2 \le \frac{1}{Z_m^2}(D_3^2 + \delta_3^2).$$

Finally for this term,

$$E_n \|\theta_n - \theta_{n+1}\|^2 \le \frac{1}{Z_m^2}(\delta_1^2 + \delta_2^2 + \delta_3^2 + D_3^2) +$$
$$+ \frac{1}{Z_m}((W D_3 + D_1)^2 +$$
$$+ (H D_3 + D_2)^2 + (W^2 + H^2)\delta_3^2) = C_0.$$

2) $\|g_n\|^2$:

$$E_n |g_n^{(1)}|^2 = E_n \left(\frac{\hat{X}_n}{\hat{Z}_n} - \frac{X_n + \Delta_n^{(1)} + \xi_n^{(1)}}{Z_n + \Delta_n^{(3)} + \xi_n^{(3)}} - v_n^{(1)} \right)^2 \le \sigma_{v1}^2 +$$

$$+ E_n((\hat{X}_n(Z_n - \hat{Z}_n) + \hat{Z}_n(\hat{X}_n - X_n) + (\hat{X}_n(\Delta_n^{(3)} + \xi_n^{(3)}) +$$
$$+ \hat{Z}_n(\Delta_n^{(1)} + \xi_n^{(1)}))(\hat{Z}_n(Z_n + \Delta_n^{(3)} + \xi_n^{(3)}))^{-1})^2.$$

Using the decompositions:

$$Z_n - \hat{Z}_n = Z_n \hat{Z}_n (\frac{1}{\hat{Z}_n} - \frac{1}{Z_n}),$$

$$X_n - \hat{X}_n = Z_n (\frac{X_n}{Z_n} - \frac{\hat{X}_n}{\hat{Z}_n}) + \frac{\hat{X}_n}{\hat{Z}_n}(Z_n - \hat{Z}_n)$$

and the inequality $(\sum_{i=1}^{N} c_i)^2 \leq N \sum c_i^2$ we get:

$$E_n|g_n^{(1)}|^2 \leq \sigma_{v1}^2 + 3\Bigg\{\frac{W^2}{Z_m^2}(Z_M^2+1)^2(\hat{\theta}_n^{(3)}-\theta_n^{(3)})^2+$$

$$+\frac{Z_M^2}{Z_m^2}(\hat{\theta}_n^{(1)}-\theta_n^{(1)})^2 + E_n((Z_n+\Delta_n^{(3)}+\xi_n^{(3)})^{-1}$$

$$(\frac{\hat{X}_n}{\hat{Z}_n}(\Delta_n^{(3)}+\xi_n^{(3)})+\Delta_n^{(1)}+\xi_n^{(1)}))^2\Bigg\}.$$

The last term itself can be bounded with $\frac{4}{Z_m^2}(W^2(\delta_3^2+D_3^2)+\delta_1^2+D_1^2)$, so we get

$$E_n|g_n^{(1)}|^2 \leq \sigma_{v1}^2 + 3\Bigg\{\frac{W^2}{Z_m^2}(Z_M^2+1)^2(\hat{\theta}_n^{(3)}-\theta_n^{(3)})^2+$$

$$+\frac{Z_M^2}{Z_m^2}(\hat{\theta}_n^{(1)}-\theta_n^{(1)})^2 + \frac{4}{Z_m^2}(W^2(\delta_3^2+D_3^2)+\delta_1^2+D_1^2)\Bigg\}.$$

Analogously for the second component,

$$E_n|g_n^{(2)}|^2 \leq \sigma_{v2}^2 + 3\Bigg\{\frac{H^2}{Z_m^2}(Z_M^2+1)^2(\hat{\theta}_n^{(3)}-\theta_n^{(3)})^2+$$

$$+\frac{Z_M^2}{Z_m^2}(\hat{\theta}_n^{(2)}-\theta_n^{(2)})^2 + \frac{4}{Z_m^2}(H^2(\delta_3^2+D_3^2)+\delta_2^2+D_2^2)\Bigg\}.$$

For the third component:

$$E_n|g_n^{(3)}|^2 = E_n|k_n\Delta_n^{(1)}(((\xi_n^{(1)}+\Delta_n^{(1)})Z_n-$$

$$-X_n(\xi_n^{(3)}+\Delta_n^{(3)}))^{-1}(Z_n(Z_n+\xi_n^{(3)}+\Delta_n^{(3)}))+$$

$$+v_n^{(1)}-v_{n-1}^{(1)})-\frac{1}{\hat{Z}_n}|^2 =$$

$$= E_n(k_n\Delta_n^{(1)}\Big(\frac{\xi_n^{(1)}Z_n - X_n(\xi_n^{(3)}+\Delta_n^{(3)})}{Z_n(Z_n+\xi_n^{(3)}+\Delta_n^{(3)})}+$$

$$+v_n^{(1)}-v_{n-1}^{(1)}\Big))^2 + E_n|\frac{1}{Z_n+\xi_n^{(3)}+\Delta_n^{(3)}}-\frac{1}{\hat{Z}_n}|^2.$$

For the last term we have

$$E_n|\frac{1}{Z_n+\xi_n^{(3)}+\Delta_n^{(3)}}-\frac{1}{\hat{Z}_n}|^2 \leq 2(\frac{1}{Z_m^2}(\delta_3^2+D_3^2)+$$

$$+(\theta_n^{(3)}-\hat{\theta}_n^{(3)})^2).$$

$$E_n|g_n^{(3)}|^2 \leq E_n v_{n-1}^{(1)}C_1 + \sigma_d^2(v_{n-1}) + 2\frac{k_n}{Z_m^2}(1+$$

$$+W^2(\delta_3^2+D_3^2)) + 2(\frac{1}{Z_m^2}(\delta_3^2+D_3^2)+(\theta_n^{(3)}-\hat{\theta}_n^{(3)})^2).$$

where we have denoted as C_1 some term independent of v_{n-1} and by $\sigma_d^2(v_{n-1}) = E_n(v_n^{(1)}-v_{n-1}^{(1)})^2$.
Finally,

$$\|g_n\|^2 \leq C_2\|\hat{\theta}_n-\theta_n\|^2 + C_1 v_{n-1}^{(1)} + C_3,$$

where

$$C_2 = \sqrt{18Z_M^4Z_m^{-4}+(3Z_m^{-2}(Z_M^2+1)(W^2+H^2)+2)^2},$$

$$C_3 = \sigma_{v1}^2+\sigma_{v2}^2+\sigma_d^2(v_{n-1}) + 12Z_m^{-2}((W^2+H^2)$$

$$(\delta_3^2+D_3^2)+\delta_1^2+\delta_2^2+D_1^2+D_2^2).$$

3) $\langle\hat{\theta}_n-\theta_n, \theta_n-\theta_{n+1}\rangle$:

$$E_n\langle\hat{\theta}_n-\theta_n,\theta_n-\theta_{n+1}\rangle \leq$$

$$\leq E_n\|\hat{\theta}_n-\theta_n\|\|\theta_n-\theta_{n+1}\| \leq E_n\|\hat{\theta}_n-\theta_n\|\sum_{i=1}^{3}|\theta_n^{(i)}-\theta_{n+1}^{(i)}|.$$

$$|\theta_n^{(1)}-\theta_{n+1}^{(1)}| \leq Z_m^{-1}(W(\delta_3^1+D_3^1)+D_1^1),$$

$$|\theta_n^{(2)}-\theta_{n+1}^{(2)}| \leq Z_m^{-1}(H(\delta_3^1+D_3^1)+D_2^1),$$

$$|\theta_n^{(3)}-\theta_{n+1}^{(3)}| \leq \frac{D_3^1+\delta_3^1}{Z_m^2},$$

$$E_n\langle\hat{\theta}_n-\theta_n,\theta_n-\theta_{n+1}\rangle \leq \|\hat{\theta}_n-\theta_n\|Z_m^{-1}((W+H)$$

$$(\delta_3^1+D_3^1)+D_1^1+D_2^1+Z_m^{-1}(\delta_3^1+D_3^1)) =$$

$$= E_n\langle\hat{\theta}_n-\theta_n,\theta_n-\theta_{n+1}\rangle \leq \|\hat{\theta}_n-\theta_n\|C_4$$

with

$$C_4 = Z_m^{-1}((W+H)(\delta_3^1+D_3^1)+D_1^1+D_2^1+Z_m^{-1}(\delta_3^1+D_3^1)).$$

$$E_n\|\hat{\theta}_{n+1}-\theta_{n+1}\|^2 \leq (1-2\alpha+\alpha^2C_2)\|\hat{\theta}_n-\theta_n\|^2+$$

$$+\|\hat{\theta}_n-\theta_n\|((2-4\alpha)C_4)+\alpha^2E_n\{C_0v_{n-1}+C_3\}.$$

Taking the unconditional expectation, we get

$$E\|\hat{\theta}_{n+1}-\theta_{n+1}\|^2 \leq (1-2\alpha+\alpha^2C_2)\|\hat{\theta}_n-\theta_n\|^2+$$

$$+\|\hat{\theta}_n-\theta_n\|((2-4\alpha)C_4)+\alpha^2\bar{C}_3 \leq$$

$$\leq (1-2\alpha+\alpha^2C_2+\frac{\varepsilon^2}{2})\|\hat{\theta}_n-\theta_n\|^2+\frac{(2-4\alpha)C_4}{2\varepsilon^2}+\alpha^2\bar{C}_3.$$

From the well-known geometric progression formula, which can be applied because

$$1-2\alpha+\alpha^2C_2+\frac{\varepsilon^2}{2} \in (0,1)$$

we get the asymptotic bound

$$\lim_{n\to\infty}E\|\hat{\theta}_n-\theta_n\|^2 = \frac{0.5\varepsilon^{-2}(2-4\alpha)C_4+\alpha^2\bar{C}_3}{2\alpha-\alpha^2C_2-0.5\varepsilon^2}.$$

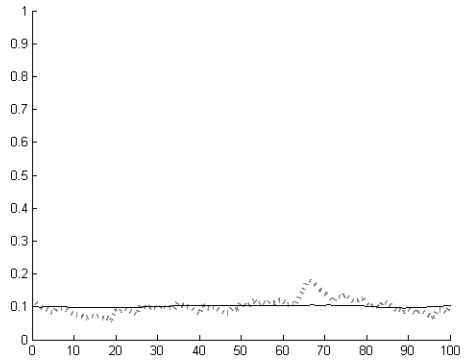

Fig. 1. Example of true trajectory (solid line) and algorithm's estimates (dotted line) for the simple case without point movement and small initial error.

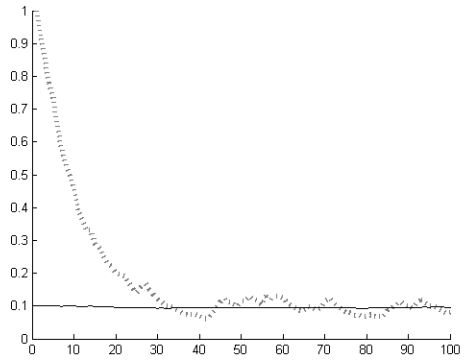

Fig. 2. Same setting as in Fig. 1 but with large initial error

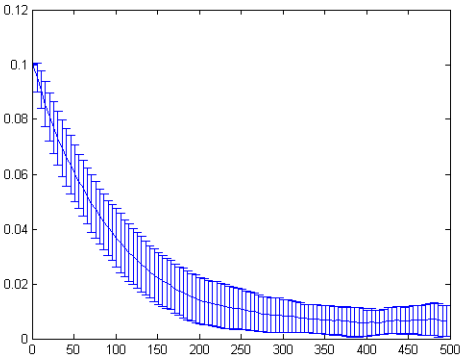

Fig. 3. Average inverse depth estimation error and its variance produced by algorithm in the case of point movement.

V. Simulation

A. Setting

In our simulation we consider single camera observing single point. Let's denote camera center at nth moment of time as C_n. For simplicity we don't assume any rotation of camera frame because it's irrelevant to our algorithm and any transformation of input caused by it can be removed in image domain beforehand. Camera center moves with constant speed V_n^C (this can be speed of a possible car on which camera is installed) according to the law:

$$C_n = C_{n-1} + V_n^C.$$

We add random pertrubation to camera position on each frame:

$$C_n = C_{n-1} + V_n^C - \Delta_n,$$

where Δ_n - is a random vector drawn from distribution on a sphere. We form vector Δ_n first by generating two random angles ϕ and θ using uniform distributions on ranges $[0, 2\pi]$ and $[0, \pi]$ accordingly. After that we set $\Delta_n = 0.1 * (sin(\theta)cos(\phi), sin(\theta)sin(\phi), cos(\theta))^T$. Another entity in our experiment is point which is observed by the camera. We denote its coordinates relative to camera frame as $P_n = (X_n, Y_n, Z_n)$. If point has its own velocity V_n^P than equation for coordinates update is:

$$P_n = P_{n-1} - V_n^C + V_n^P.$$

In all of our tests initial position of camera is set to $C_0 = (0,0,0)^T$ and point to $P_0 = (0,0,10)^T$. Measurement noise is uniform with expected value equal to zero and varies in range $[-0.001, 0.001]$ for both x and y coordinates of point projection on camera plane.

B. Results

In the most basic scenario we assumed that $V_n^C = 0$ and $V_n^P = 0$. We tested our algorithm in different conditions of initial estimate error. In case of small initial error algorithm successfully tracks inverse depth which can be seen on Fig. 1. In the case of large initial error there is a period of initial convergence to the true trajectory after which algorithm continues to track point position (Fig. 2). In this setting we set algorithm's parameters as follows. Initial estimate was set to $\theta_0 = (0,0,1.1)$, α was set to 0.1.

The most interesting scenario in our testing is when a camera and a point move in same direction and with the same speed. In this setting without adding random camera movement one wouldn't able to estimate point position in any possible way because. We set $VC_n = VP_n = (0.5,0,0)^T$. Fig. 3 shows averaged inverse depth estimation error along with its variance for our algorithm ($\alpha = 0.001$) Also in Fig. 4 we show error for estimates of X/Z to illustrate that our approaches successfully track it too. Initial estimate in this test was set to $\theta_0 = (0,0,0.2)$. It is clear that even in such complicated scenario our method produce reliable estimates of point position.

VI. Conclusions

We have presented an approach to reconstruct the trajectories of moving objects observed by a moving camera. The approach is based on random camera motions which are independent of any motion in the scene. The approach can be implemented either as a special device with programmed random camera motion, where the trajectory of a device is known a priori, or as a device moving in random directions, the motion of which can be reconstructed using static points in the scene with conventional monocular SLAM approaches.

In this paper we have shown theoretical arguments for the convergence of our method, as well as numerical simulation results for it. We show numerically that our approach succeeds in reconstruction of the true point trajectory in case of camera and point moving with same speed, as in real traffic flow.

In the future, we'd like to create a complete SLAM system built based on a proposed here principle, and we are planning to make more theoretical research in the field of optimal parameter choice for the algorithms. Also, we are considering using our method combined with modified Kalman filter to produce estimates of better quality.

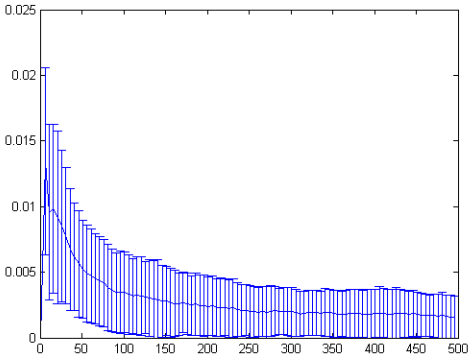

Fig. 4. Average estimation error for X/Z and its variance in the case of point movement.

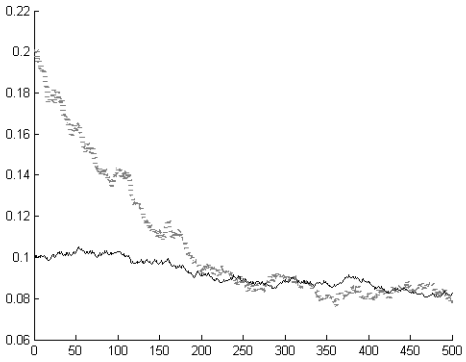

Fig. 5. Trajectory example for the case of point movement.

References

[1] Z. Huijing, C. Masaki, S. Ryosuke, S. Xiaowei, C. Jinshi, and Z. Hongbin. Slam in a dynamic large outdoor environment using a laser scanner. In *Robotics and Automation, IEEE International Conference on*, pages 1455–1462. IEEE, 2008.

[2] M.W.M.G. Dissanayake S.B. Williams, P. Newman and H.F. Durrant-Whyte. Autonomous underwater simultaneous localisation and map building. In *Robotics and Automation, Proceedings. IEEE International Conference on*, pages 1143–1150, 2000.

[3] J. Kim and S. Sukkarieh. Autonomous airborne navigation in unknown terrain environments. *Aerospace and Electroninc Syststems, IEEE Transactions on*, 40(3):1031–1045, 2004.

[4] A.J. Davison and D. Murray. Simultaneous localization and map-building using active vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):865–880, 2002.

[5] P.M. Newman. *On the structure and solution of the simultaneous localization and mapping problem*. University of Sydney, 1999.

[6] M. Han and T. Kanade. Reconstruction of a scene with multiple linearly moving objects. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 542–549. IEEE, 2000.

[7] P. Sturm. Structure and motion for dynamic scenesthe case of points moving in planes. *Computer VisionECCV 2002*, pages 507–509, 2002.

[8] A.T. Vakhitov, V. Vlasov, and O.N. Granichin. Adaptive control of siso plant with time-varying coefficients based on random test perturbation. In *American Control Conference (ACC), 2010*, pages 4004–4009. IEEE, 2010.

[9] O.N. Granichin. Procedure of stochastic approximation with disturbances at the input. *Automation and Remote Control*, 53(1):232–237, 1992.

[10] J.C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *Automatic Control, IEEE Transactions on*, 37(3):332–341, 1992.

[11] O.N. Granichin, L.S. Gurevich, and A.T. Vakhitov. Discrete-time minimum tracking based on stochastic approximation algorithm with randomized differences. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 5763–5767. IEEE, 2009.

[12] O.N. Granichin, V. Volkovich, and D. Toledano-Kitai. *Randomized Algorithms in Automatic Control and Data Mining*. Springer-Verlag: Heidelberg New York Dordrecht London, 2014.