Algorithms for the Visual Odometry

Alex Kreimer

Algorithms for the Visual Odometry

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

Alex Kreimer

Submitted to the Senate
of the Technion — Israel Institute of Technology
Tevet 5772 Haifa August 2017

This research was carried out under the supervision of Prof. Ilan Shimshoni and of Prof. Ehud Rivlin

Info about whether the contents of the thesis was published, about collaborators and the thesis author's contribution.

The Technion's funding of this research is hereby acknowledged.

Contents

List of Figures

Abstract

In this work we revisit the problem of visual odometry. Visual odometry is the process of estimating the motion of the camera by examining the changes that the motion induces on the images made by it. This work has two parts: the first part proposes a novel algorithm for the visual odometry. The approach we propose exploits a scene structure typical for that seen by a moving car and is suitable for use in either the stereo or the monocular setting. We recover the rotation and the translation separately, thus dealing with two separate, smaller problems. The rotation is estimated by means of the infinite homography. The rotation estimation algorithm operates on distant image points using the 3-D to partition them into the distant and the near-by ones. We start with an initial estimate and then refine it using an iterative procedure. After the rotation is compensated for, the translation is found by means of the 1-point algorithm in the stereo setting and epipole computation for pure translational motion in the monocular setting. We evaluate our algorithm on the KITTI dataset [GLU12]. The second part of this work explores a method to recover the scale of camera motion. The size of a translation vector for a single moving camera is not directly observable, although is desirable. Stereo, scene/camera prior assumptions were used in the past to recover the translation size. We argue that the required information is present in the images and explore a number of ways to learn it. We experiment with both "legacy" shallow learning methods and hand-crafted features as well as end-to-end learning methods based on the convolutional neural networks.

Chapter 1

Introduction

Visual odometry refers to the problem of recovering camera motion based on the images taken by it. This problem naturally occurs in robotics, wearable computing, augmented reality and automotive.

Wheel odometry, recovers the motion of the vehicle by examining and integrating the wheel turns over time. In a similar manner, visual odometry operates by estimating relative motion of the camera between subsequent images by observing changes in them. Later, these estimates are combined into a single trajectory. Just as in wheel odometry, visual odometry is subject to error accumulation over time. Contrary to wheel odometry, visual odometry is not affected by wheel slip in a rough terrain. Visual odometry is able to produce motion estimates with errors that are lower than those of the wheel odometry. Another advantage of visual odometry is that cameras are low cost and low weight sensors. All these make visual odometry a viable supplement to other motion recover methods such as global positioning systems (GPS) and inertial measurement units (IMUs).

Visual odometry becomes a harder problem as the amount of detail in the images diminishes. The images should have sufficient overlap and the scene needs to be illuminated. In the stereo setup, the scene must be static or the images taken at the same time. Also, the video processing incurs a computational burden.

Visual odometry is an active fields of research with a large amount of published work. We review only the most pertinent works. [SF11] provides a more complete survey.

Similar to [PPFM15] we partition visual odometry algorithms by four traits:

1. Feature-based vs direct

2. Global vs local

3. Filter based vs bundle adjustment based

4. Monocular vs stereo

Visual odometry algorithms use large number of corner detectors (e.g., Moravec [Mor80], Harris [HP87], Shi-Tomasi [ST94], Fast [RD06]) and blob detectors (e.g., SIFT [Low04], SURF [BTV06]). Corners are faster to compute and usually are better localized, while blobs are more robust to scale change. The choice of a specific feature point depends mainly on the images at hand. Motion estimation results for different feature points are presented in [Gov09]. In this work we choose Harris [HP87] corners, but this choice is not crucial. We view the feature point choice as a parameter, which needs to be determined from the data (e.g., by cross-validation).

The features are either tracked [HFF09] or matched [GZS11] (i.e., freshly detected in each new frame) between subsequent images. While the early works chose to track features, most of the current works detect and match them. The output of this stage are pairs of the image features, which are the projections of the same 3-D point.

Matched features are used as an input for a motion estimation procedure. Whether the features are specified in 2-D or 3-D, the estimation procedures, may be classified into 3-D-to-3-D [MS06], 3-D-to-2-D [GZS11] and 2-D-to-2-D [Nis]. Most of the early works were of the 3-D-to-3-D type. More recent works [Nis] claim that this approach is inferior to the latter two. Popular techniques that participate in most algorithms in some way are essential matrix estimation and (possibly) its subsequent decomposition [Nis], perspective 3-point algorithm [KSS11], and re-projection error minimization [GZS11].

Global methods [Kle07], [NLD11] keep the map of the environment and make sure that motion estimates are globally consistent with this map, while local methods do not. Some local methods [BYK13] also keep track of a (local) map, but the underlying philosophy is different: global vs local. Global methods usually more accurate since they make use of a vast amount of information (which, of course, comes at a computational price). Note that accuracy does not imply robustness, since outliers that made their way into the map may greatly skew subsequent pose estimates.

Methods that explicitly model system state uncertainty tend to use filtering mathematical machinery, e.g., [KAS10], [OMSM03], [KRD08]. Another alternative to maintain map/pose estimate consistency is to use the bundle adjustment approach [TMHF00]. Monocular systems [SG] make use of a single camera, while stereo systems [GZS11] rely on a calibrated stereo rig. In the monocular setup the translation of the camera may only be estimated up to scale, while in stereo all six motion parameters may be recovered. An additional advantage of the stereo setup is that more information is available at each step, which may be one of the reasons why stereo algorithms perform better.

Chapter 2

Infinite Odometry

2.1 Introduction

In this work we present a novel algorithm for camera motion estimation. The novelty of the algorithm is in camera rotation estimation procedure. We rely on the fact that for scene points that are infinitely far from the camera, the motion of the projected (image) points may be described by an homography (the infinite homography). For distant points this assumption is nearly true. Our algorithm starts by partitioning the scene points into two sets: distant and near-by. Then, camera rotation is estimated from the distant points and, subsequently, the translation is recovered from the near-by points.

We present two versions of the algorithm: one for the monocular and the other for the stereo settings. These versions differ in the way we partition points into the distant and the near-by ones and in the way the algorithms estimate translation.

With respect to the classification of the visual odometry methods given in the introduction, our work is local, feature based, stereo odometry. We do not use bundle adjustment, however the results of our algorithm may be subsequently improved with some form of bundle adjustment.

The outline of the our method:

1. Feature detection. We use Harris [HP87] corners.
2. Feature matching. The matching is done both across the stereo pair images as well as previous vs. current pair. We enforce epipolar constraint, chierality and use circle heuristics similar to [GZS11] to reject outliers.
3. Partition the scene points into two sets: distant and near-by.
4. Estimate the rotation of the camera from the distant points.
5. Estimate the translation of the camera from the near-by points.

We choose the work [GZS11] as our baseline (our implementation of their work). The results in the Section 2.4 show that on the KITTI dataset our rotation estimation method outperforms the baseline.

2.2 Preliminaries and Notation

2.2.1 Image Point Mapping Related to Camera Motion

Suppose the camera matrices are those of a calibrated stereo rig P and P' with the world origin at the first camera

$$\mathrm{P} = \mathrm{K}[\mathrm{I} \mid 0], \quad \mathrm{P}' = \mathrm{K}'[\mathrm{R} \mid \mathbf{t}]. \tag{2.1}$$

Consider the projections of a 3D point $\mathbf{X} = (X, Y, Z, 1)^T$ into the image planes of both views:

$$\mathbf{x} = \mathrm{P}\mathbf{X}, \quad \mathbf{x}' = \mathrm{P}'\mathbf{X}. \tag{2.2}$$

If the image point is normalized as $\mathbf{x} = (x, y, 1)^T$ then

$$\mathbf{x}Z = \mathrm{P}\mathbf{X} = \mathrm{K}[\mathrm{I} \mid 0]\mathbf{X} = \mathrm{K}(X, Y, Z)^T.$$

It follows that $(X, Y, Z)^T = \mathrm{K}^{-1}\mathbf{x}Z$, and:

$$\begin{aligned} \mathbf{x}' &= \mathrm{K}'[\mathrm{R} \mid \mathbf{t}](X, Y, Z, 1)^T && (2.3)\\ &= \mathrm{K}'\mathrm{R}(X, Y, Z)^T + \mathrm{K}'\mathbf{t} && (2.4)\\ &= \mathrm{K}'\mathrm{R}\mathrm{K}^{-1}\mathbf{x}Z + \mathrm{K}'\mathbf{t}. && (2.5) \end{aligned}$$

We divide both sides by Z to obtain the mapping of an image point $\mathbf{x}$ to image point $\mathbf{x}'$

$$\mathbf{x}' = \mathrm{K}'\mathrm{R}\mathrm{K}^{-1}\mathbf{x} + \mathrm{K}'\mathbf{t}/Z = \mathrm{H}_\infty\mathbf{x} + \mathrm{K}'\mathbf{t}/Z = \mathrm{H}_\infty\mathbf{x} + \mathbf{e}'/Z. \tag{2.6}$$

H_∞ is the infinite homography that transfers the points at infinity to the points at infinity. If $\mathrm{R} = \mathrm{I}$ (e.g., pure translation) the point $\mathbf{x}$ will undergo a motion along a corresponding epipolar line:

$$\mathbf{x}' = \mathbf{x} + \mathrm{K}'\mathbf{t}/Z = \mathbf{x} + \mathbf{e}'/Z. \tag{2.7}$$

If $\mathbf{t} = \mathbf{0}$ the motion of the point may be represented by a homology:

$$\mathbf{x}' = \mathrm{H}_\infty\mathbf{x}. \tag{2.8}$$

In a general case the mapping of an image point $\mathbf{x}$ into $\mathbf{x}'$ may be viewed as a two step process: transformation by a homology (a specialization of homography which has two equal eigenvalues) H_∞ which simulates a pure rotational motion of the camera followed by an offset along the epipolar line which simulates a pure translational motion of the camera, see Figure 2.1.

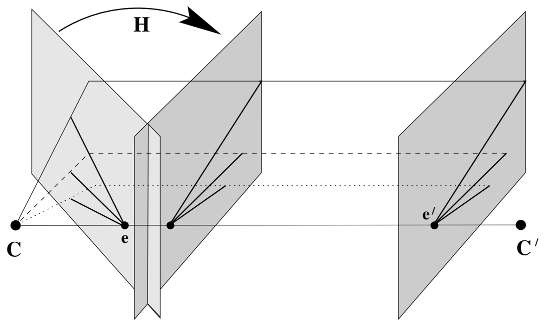


Figure 2.1: (Adapted from [HZ04]) The effect of the camera motion on the image points may be viewed as a two-step process: mapping by a homography H_∞ followed by a motion along the corresponding epipolar lines.

2.3 Motion Estimation

Our strategy to attack the problem is to separate it into two smaller sub-problems: rotation estimation and translation estimation. The algorithm relies on the ability to partition the scene points into two sets: the distant and the near-by ones. The distant points are used for rotation estimation while the near-by ones take part in the translation estimation.

First, the stereo algorithm is presented, followed by the monocular one. The main difference is in the translation estimation part. While it is possible to implement a stereo-like algorithm in the monocular setting as well, it suffers from the scale drift. Thus, we propose a different technique.

2.3.1 Stereo

Partitioning the points To partition the points in the stereo case we hard-threshold their Z-coordinates (the threshold is a parameter of the algorithm). The depth of the points was computed by a stereo triangulation.

Rotation Estimation: We use distant points to estimate rotation R (i.e., near-by points do not take part in rotation estimation). As Eq. (2.6) states:

$$\mathbf{x}' = \mathrm{KRK}^{-1}\mathbf{x} + \mathrm{K}\mathbf{t}/Z. \tag{2.9}$$

The total motion of the feature point in the image plane may be viewed as a two-step process (the order is not important): transformation by homography $\mathrm{H}_\infty = \mathrm{KRK}^{-1}$ followed by displacement along the line defined by the epipole e' and the point $\mathrm{H}_\infty\mathbf{x}$. The magnitude of the displacement along the epipolar line depends on the the camera translation and the inverse depth of the point, see Figure 2.2.

Our estimation algorithm consists of initialization and non-linear refinement.

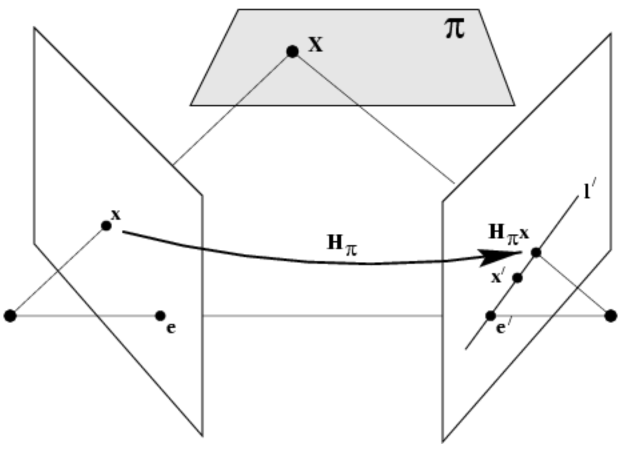


Figure 2.2: (Adapted from [HZ04]) The homography H_π transfers the point onto a corresponding epipolar line. The displacement along the epipolar line depends on the inverse depth of the point and the camera translation magnitude.

Initialization: to compute the initial estimate of the rotation parameters we assume that for the distant points (s.t., $\|\mathbf{t}\|/Z \ll \|\mathrm{H}_\infty \mathbf{x}\|$):

$$\mathbf{x}' \approx \mathrm{KRK}^{-1}\mathbf{x}. \tag{2.10}$$

This assumption is justified by the fact that for the distant points the displacement along the epipolar line is small. We multiply both sides of Eq. (2.10) by K^{-1} and denote $\mathbf{u}' = \mathrm{K}^{-1}\mathbf{x}$ and $\mathbf{u} = \mathrm{K}^{-1}\mathbf{x}$:

$$\mathbf{u}' = \mathrm{K}^{-1}\mathbf{x}' \approx \mathrm{RK}^{-1}\mathbf{x} = \mathrm{R}\mathbf{u}. \tag{2.11}$$

Since $\mathbf{u}$ and $\mathbf{u}'$ are projective quantities, only their directions are of importance, we normalize them to unit length and denote normalized quantities by $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{u}}'$ respectively. We choose a sample of n points ($n = 3$) and stack them as columns of matrices $\tilde{\mathrm{U}}$ and $\tilde{\mathrm{U}}'$ respectively. We search for R that solves the following minimization problem:

$$\underset{\mathrm{R}}{\operatorname{argmin}} \|\tilde{\mathrm{U}}' - \mathrm{R}\tilde{\mathrm{U}}\|_2. \tag{2.12}$$

Eq. (2.12) is known as the absolute orientation problem (see e.g., [Hor87]) and its solution provides an initial estimate for the subsequent non-linear optimization problem.

Refinement: The idea of the refinement is this: the residual vector $\mathrm{H}_\infty\mathbf{x} - \mathbf{x}'$ may be viewed as a sum of a vector orthogonal to the epipolar line and the vector parallel to it. We search for the camera rotation that ignores the parallel component (we view it as a "legal" bias) while trying to minimize the orthogonal one. We define the point residual as the orthogonal distance to the corresponding epipolar line and minimize the sum of squared residuals for all points. We do so, because, as Eq. (2.6) suggests, after we compensate for a rotation, the point is still allowed to move along the epipolar line.

Consider the objective:

$$\mathrm{R}(\mathrm{v},\theta) = \underset{v,\ \theta}{\operatorname{argmin}} \sum_{i=1}^{N} {r_i}^2 \quad \text{s.t.} \quad r_i = n_i \cdot (\mathbf{x}'_i - H_\infty(v,\theta)\mathbf{x}_i) \\ \text{where } n_i = (\mathrm{F}\mathbf{x}_i)_\perp \tag{2.13}$$

$\mathrm{H}_\infty(v,\theta) = \mathrm{KR}(\mathrm{v},\theta)\mathrm{K}^{-1}$ is the homography that transforms the image points in Eq. (2.6). It depends on a known camera intrinsic matrices K and a rotation matrix R. We choose to parameterize the rotation by an angle θ and an axis v. F denotes the fundamental matrix that corresponds to two subsequent stereo rig poses and is computed elsewhere. $\mathrm{F}\mathbf{x}_i$ denotes the epipolar line that corresponds to $\mathbf{x}_i$ in the second image and $(\mathrm{F}\mathbf{x}_i)_\perp$ is the normal to this line. We solve the minimization problem (2.13) by means of the Levenberg-Marquardt optimization algorithm.

To make the estimation robust we wrap the initialization procedure into the RANSAC iterations. We choose the strongest consensus estimate and its support set as an input for the solution of the Eq. (2.13).

Translation Estimation (1-point algorithm) To estimate the translation we use only the near-by points. First, we triangulate these points in the previous stereo pair to obtain their 3-D locations, and then iteratively minimize the sum of reprojection errors into the current frame.

The reprojection of point $\mathbf{X} = (X, Y, Z, 1)^T$ into the current left image is given by:

$$\pi^{(l)}(\mathbf{X}; \mathbf{t}) = K\Big[\mathrm{R} \mid \mathbf{t}\Big]\mathbf{X}. \tag{2.14}$$

and the reprojection of the same point into the current right image (b is the baseline of the stereo-rig) is given by:

$$\pi^{(r)}(\mathbf{X}; \mathbf{t}) = K\Big[\mathrm{R} \mid \mathbf{t}\Big](\mathbf{X} - (b, 0, 0, 0)^T), \tag{2.15}$$

We use the Levenberg-Marquardt algorithm to iteratively minimize the sum of squared reprojection errors (starting from $\mathbf{t} = \mathbf{0}$):

$$\|\mathbf{x}' - \pi^{(l)}(\mathbf{X}; \mathbf{t})\|^2 + \|\mathbf{x}' - \pi^{(r)}(\mathbf{X}; \mathbf{t})\|^2. \tag{2.16}$$

There are three unknown parameters, since $\mathbf{t} = (t_x, t_y, t_z)^T$, thus a single 3-D point provides enough constraints to determine $\mathbf{t}$.

2.3.2 Mono

The stereo setup has an advantage over the monocular one in the sense that it provides the algorithm with more information (e.g., the calibration and the additional image at each camera position). These advantages come at a price, e.g., the cameras need to be synchronized and the computational resource requirements climb. These make monocular setups and the related algorithms more attractive. In the following section we present the version of our algorithm, adapted for the monocular setup.

Given two sets of matching image points $\mathbf{x}_i, \mathbf{x}'_i$ from two subsequent frames I_1, I_2 respectively, we estimate the camera motion between these frames. Similar to the stereo algorithm in Section 2.3.1 the algorithm first partitions the points and then estimates the rotation followed by the estimation of the translation direction (it is well known that the magnitude of the camera translation is unavailable in the monocular setting).

Partitioning the points To estimate the rotation of the camera as in the Eq. (2.3), it is required to partition the set of the image points into the distant ones and the near-by ones. While in the stereo setting we may triangulate the points and threshold their depths, in the monocular setting this can not be done. This section proposes a method to perform the aforementioned partition in the monocular setting.

Consider the subsequent images I_1, I_2, I_3 taken by a moving camera at the locations O_1, O_2 and O_3 respectively. We assume the image points $\mathbf{x}_i$ in I_1 are known to be distant relative to the camera at O_1. The magnitude of the camera translation is small relative to the distant points depths, thus we assume that these points are distant w.r.t. the camera at O_2 and O_3 as well. Some of these points will be lost in I_3, thus it is desirable to known which of the points tracked from I_2 to I_3 (which are not part of $\mathbf{x}_i$'s) are distant (denote these by $\mathbf{y}_j$).

The real baselines $t_1 = \|O_1 - O_2\|$ and $t_2 = \|O_2 - O_3\|$ are unknown and thus we can not use them to obtain real depths of the points.

We use the following procedure to classify the newly tracked points in I_2 as distant:

1. Set $t_1 = 1$ and triangulate the distant points $\mathbf{x}_i$ to obtain the depths Z_i

2. Set $t_2 = 1$ and triangulate the points $\mathbf{y}_j$ to obtain the depths Z_j

3. Classify the point y_j to be distant if $Z_j > \min_i Z_i$.

While the assumption $t_1 \approx t_2$ is acceptable for the KITTI dataset, it may be improved on by computing the t_1/t_2 ratio (by minimizing the reprojection errors of $\mathbf{x}_i$ into I_3, similar to the translation estimation described in the Section 2.3.1).

To initialize the monocular algorithm we may further assume that the initial motion is a pure translation and thus the points with small disparity are the distant ones (disparity being the magnitude of the motion in the image plane).

Rotation Estimation is exactly as in the Section 2.3.1. Denote the estimated rotation by R and the corresponding homography by H.

Translation Estimation is as follows. Compensate the rotation by computing $\mathbf{y}_i = \mathrm{H}_\infty \mathbf{x}_i$. We optimize over the location of the epipole e and minimize the orthogonal distances of the points to their corresponding epipolar lines:

$$\underset{e}{\operatorname{argmin}} \sum_i d(l_i, \mathbf{y}_i) + d(l_i, \mathbf{x}'_i) \text{ s.t. } d(e, l_i) = 0 \text{ while } l_i = \underset{l}{\operatorname{argmin}}\ d(l, \mathbf{x}'_i) + d(l, \mathbf{y}_i) \tag{2.17}$$

We define the epipolar line l_i to be the line that passes through the epipole and its distance to $\mathbf{x}'_i$ and $\mathbf{y}_i$ is minimal. $d(l, \mathbf{x})$ denotes the distance from the line l to the point $\mathbf{x}$. The epipole provides us with the translation direction of the camera.

2.4 Experimental Results

2.4.1 The Choice of Features

We chose to evaluate our algorithm on the KITTI dataset [GLU12], which is a de-facto standard for the visual odometry research works.

Feature Detector/Descriptor: We use Harris [HS88] corner detector. It is fast, well localized and (most important) Harris corners are abundant in urban scenes we work with. We detect corners in each new image and then match them to obtain putative matches. We tune sensitivity threshold of the detector in such a way, that we are left with about five hundred putative matches after matching and pruning. We extract a square patch of 7×7 pixels centered at the corner point and use this vector as feature descriptor.

We would like to point out that our method may be used with any feature detector that would allow to match features across images. The choice of feature detector should be viewed as a parameter to the algorithm and mainly depends on the images at test.

Feature Matching: We use sum-of-square differences (SSD) of feature descriptors as a metric function when matching features. For each feature we choose a single best match w.r.t. the metric function in the other image. We employ a number of heuristics to prune outliers:

- Reciprocity: features a and b match only if a matches b and b matches a
- Epipolar constraint: we work with calibrated stereo pair. When we match features across images of stereo pair, the search is one-dimensional, i.e., along the horizontal epipolar line. This heuristic is not used when matching features across subsequent frames.

- Chierality (visibility): also used when matching features across stereo pair images. We triangulate the features to obtain the 3-D point and keep the match only if the 3-D point is visible in both cameras.

- Circular match: similar to [GZS11] we keep only those matches that form a circle.

2.4.2 Experimental Results

The Tables 2.1 and 2.2 present the results of the experiments for the KITTI dataset. The columns denote the number of the sequence. The rows denote the algorithm: SS is the baseline, HX is the stereo version of the algorithm, HG is the monocular version. The numbers are the mean error for the corresponding sequence with the last column is the mean error for the dataset. The error computation method is described in [GLU12].

Stereo In this set of experiments we ran our algorithm in its stereo mode as described in the Section 2.3.1. Table 2.1 and 2.2 present the rotation and the translation errors respectively in the row HX. The columns marked bold are those that our algorithm outperforms the baseline (on 9 of 11 sequences). The results show that our algorithm improves the rotation results over the benchmark algorithm and successfully competes with it in the translation estimation.

Mono In an additional set of experiments we ran our algorithm in a monocular mode. Monocular motion estimation lacks a scale parameter. In order to compare the results we set the scale of the translation to be that of the stereo algorithm. Feature point selection/partition was done without using any stereo information and the motion estimation was done as explained in the Section 2.3.2.

2.5 Conclusions and Discussion

This paper presents a novel visual odometry algorithm. The novelty of the algorithm is in its rotation estimation method. The rotation is estimated by means of the infinite homography. The algorithm may be used both in the stereo and in the monocular setting.

The strengths of the presented algorithm are in its ability to split the motion estimation problem into two smaller problems and to operate directly on the image points instead of on the computed 3-D quantities. Splitting the problem helps because each sub-problem is easier to solve. The ability to partition the points into the distant and the near-by ones is what allows us to separate the rotation and the translation estimation.

The stereo version of the algorithm shows better performance, but the monocular version has the advantage of being a more practical one. Indeed the authors in [GLU12]

Table 2.1: Rotation errors for the KITTI sequences [deg/m]

	00	01	02	03	04	05	06	07	08	09	10	mean
SS	3.95e-04	1.98e-04	4.11e-04	1.07e-03	8.03e-04	3.49e-04	4.72e-04	2.96e-04	3.69e-04	3.44e-04	4.82e-04	4.71e-04
HX	**2.70e-04**	**1.75e-04**	**4.10e-04**	**6.51e-04**	**6.04e-04**	3.95e-04	**3.77e-04**	**2.37e-04**	**3.23e-04**	**3.22e-04**	5.66e-04	**3.93e-04**
HG	8.72e-04	3.89e-04	6.28e-04	1.07e-03	5.99e-04	6.96e-04	3.31e-04	8.12e-04	8.13e-04	6.82e-04	5.23e-04	6.74e-04

Table 2.2: Translation errors for the KITTI sequences %

	00	01	02	03	04	05	06	07	08	09	10	mean
SS	4.40e+00	9.25e+00	4.03e+00	1.22e+01	5.06e+00	2.80e+00	4.37e+00	2.21e+00	4.12e+00	5.25e+00	5.60e+00	5.39e+00
HX	3.07e+00	1.08e+01	3.80e+00	7.94e+00	3.82e+00	4.06e+00	3.99e+00	1.67e+00	3.28e+00	3.77e+00	5.65e+00	4.72e+00
HG	1.21e+01	1.48e+01	8.72e+00	1.33e+01	8.62e+00	8.37e+00	4.46e+00	7.93e+00	9.76e+00	1.16e+01	8.36e+00	9.82e+00

report that they re-calibrate the cameras before each drive, which is hardly possible in real world installations.

Chapter 3

Monocular Scale

3.1 Introduction

Recovering camera 6-DOF ego-motion from images is a well studied problem. It arises in various practical contexts (e.g. virtual/augmented reality application, autonomous or aided navigation, etc.). The problem was studied in both stereo and the monocular setups. To recover the full 6-DOF motion, the previous works resorted to the stereo setup, used auxiliary sensors (e.g. IMU) or made assumptions about the camera pose and the scene. All of these have their drawbacks: stereo pairs are fragile and require careful calibration procedures, additional sensors are not always available and also require calibration, scene assumptions don't always hold. Motion estimation from images of a single moving camera is probably the hardest setup, as well as the most desirable one, because of its simplicity. It is well known that the translation scale parameter is not directly observable for a motion of a single camera.

We argue, that for natural scenes the scale information is present in the images and may be extracted. Our method learns a regressor, that is capable to prediction the motion scale.

3.2 Related work

Geometry based methods One category of works proceed by making assumptions on the camera motion, e.g., [SFPS09]. The authors assume that the vehicle adheres to the Ackerman steering principle and exploit the non-holomicity of the vehicle motion, to compute the exact scale of the camera motion. Other works make assumptions about the scene (e.g., a presense of the planar surface in front of the vehicle) and the height of the camera, e.g. [ZDL16], where the authors estimate the ground plane homography that relates subsequent images. The motion scale is estimated/refined based on the homography that relates the ground plane in subsequent images.

Learning methods We are not aware of works that learn the scale, rather there are a number of methods that use machine learning approaches to tackle the visual odometry. One such work is [**?**] that trains LSTM to predict the full 6-dof motion of the camera over sequences of images. In [**?**] the authors use optical flow images as CNN input to learn and infer camera motion. Authors in [**?**] also propose to train a CNN.

3.2.1 Our method

We assume that a single camera moves through space and takes images. We treat the initial camera pose (at time $t = 0$) as the world coordinate frame. We denote the pose of the camera at time t by $\hat{\mathbf{T}}_t$ described by the rotation matrix $\hat{\mathbf{R}}_t$ and the translation vector $\hat{\mathbf{t}}_t$ as seen in the world coordinate frame. We denote camera image taken at time t by I_t. To facilitate the discussion, we also introduce notation for camera pose $\mathbf{T}_t = [\mathbf{R}_t \mid \mathbf{t}_t]$ as seen from the coordinate frame associated with camera pose at time $t - 1$. Most of the time, we will omit the time index, since its clear from the context. By translation scale (or simply, scale) we refer to the norm of the translation vector $\mathbf{t}$ (e.g., $s = \|\mathbf{t}\|$)

We pose the scale estimation problem as a regression problem and search for a good regressor model.

3.3 Random forest

3.3.1 Decision trees

In this section we briefly describe tree-based methods for regression. These involve splitting the feature space into a number of small regions. The prediction for a sample is made by computing a mean or a mode of training samples that belong to the same region. Since the set of rules used to split the feature space into smaller regions may be described by a tree, these methods are referred to as *decision tree* methods.

Building a decision tree may be described by a two step procedure:

1. Split the feature space, e.g., the set of all possible values for $X_1, X_2, \ldots, X_n$ into J distinct regions $R_1, R_2, \ldots, R_J$.

2. For every sample that belongs to the regions R_i we make the same prediction, which is a mean of the responses of training samples that belong to this region.

The regions R_i are usually chosen to be multidimensional boxes (axis aligned). We would like to find such a partition of the feature space that minimizes

$$\sum_{j=1}^{J} \sum_{x \in R_j} (x - \bar{x}_j)^2 \tag{3.1}$$

Where $\bar{x}_j$ denotes the mean of the response values of the samples in region R_j. Unfortunately, solving the optimization problem 3.1 is computationally hard. Usually it is replaced with a greedy algorithm, called *recursive binary splitting.* This approach starts at the top of the tree and greedily chooses the best split at that point that minimizes the variance of its sub-trees. To be more precise, for each j and s we define the hyper-planes:

$$R_1(j,s) = \{X|X_j < s\} \quad \text{and} \quad R_2(j,s) = \{X|X_j \geq s\}, \tag{3.2}$$

We seek such j and s that minimize the equation:

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2, \tag{3.3}$$

Where y_{R_1}, y_{R_2} are the average responses of the samples in $R_1(j,s), R_2(j,s)$ respectively. Once we found the j and s we recursively split each sub-tree in a similar manner. The process is repeated until stopping criterion (e.g., number of nodes in the leaf) is reached.

3.3.2 Bagging

The decision trees tend to suffer from *high variance.* This means that if we split the training set into a number of random subsets and fit random tree into each sub-sample, we would likely to get much different answers from these trees when asked the same question. It is known that the variance of a mean of a set of independent random variables is $\frac{1}{n}$. Thus, in order to improve the variance of the estimator, it is possible to fit n estimators, each to its training-set and the average their predictions. Since, usually, we don't have n training sets, we would use *bootstrapping* (e.g., sample independently with replacement from the data set).

3.3.3 Random Forest

Random forest suggest additional improvement over bagging, by decorrelating the random trees. The issue they attempt to address is this: lets say there is a very dominant feature w.r.t. to task at hand for a given data-set. Bagging ensures that we use different training sets, but yet, most trees will tend to first split on this dominant feature. In this case the trees will resemble each other. In random forest, the trees are constructed by using only a subset of features (e.g., $m = \sqrt{p}$). This means that at calculating the splits, the algorithm is allowed to consider only a subset of features.

3.4 Convolutional Neural Networks

Convolutional neural networks (CNN) leverage the availability of the computational power and the abundance of data. CNN have become a method of choice in a number of computer vision areas (e.g., image classification [KSH12] [SZ14], [SLJ+15], object recognition [SEZ+13] [GDDM14] [HZRS14]. They were also shown capable of per-pixel tasks, such as semantic segmentation [NDL+05] [GGAM14], depth estimation from a single image [LSLR16], optical flow estimation [FDI+15].

Network architecture is a choice that needs to be made a-priori. To best of our knowlegde, there is no clear guideline on how to choose a network architecture for a new task. We experiment with two network architectures:

- the ZF [ZF13] object recognition network. This is a more traditional CNN architecture.
- the FlowNet [FDI+15] optical flow estimation network. This architecture adheres to a newer fully-convolutional family of networks. These are known to train better and have less parameters vs. their fully-connected counterpart networks.

The input to our network is a pair of subsequent images. A decision to make is how to input the images into the network. There are two common choices: either create two siameze branches, so that each gets its own input image, or to create a single multichannel image. [FDI+15] show that the latter works equally well while being simpler, so we choose to concatenate the images along the channel dimension to produce a single 6-channel (for color) or 2-channel (for grayscale) image.

3.4.1 ZF

The network architecture consists of five convolutional and three fully connected layers. Table 3.1 summarizes the architecture.

Layer	**Receptive Fied Size**	**Padding**	**Stride**	**Number of Channels**
conv1	7×7	2	3	96
conv2	5×5	2	2	256
conv3	3×3	1	1	384
conv4	3×3	1	1	384
conv5	3×3	1	1	256
fc6				4096
fc7				4096
fc8				1

Table 3.1: ZF network geometry

The flow of data through the network

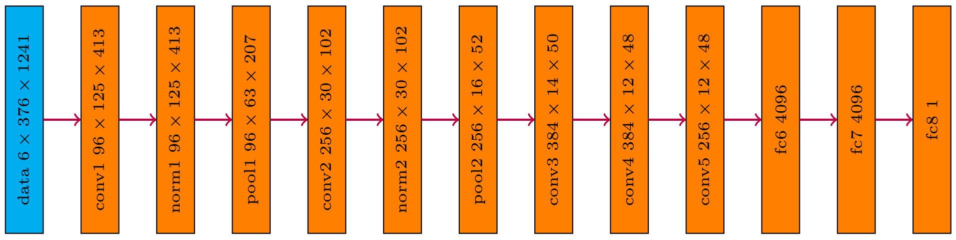


Figure 3.1: ZF network data flow. Each rectangle depicts a top blob for a corresponding layer.

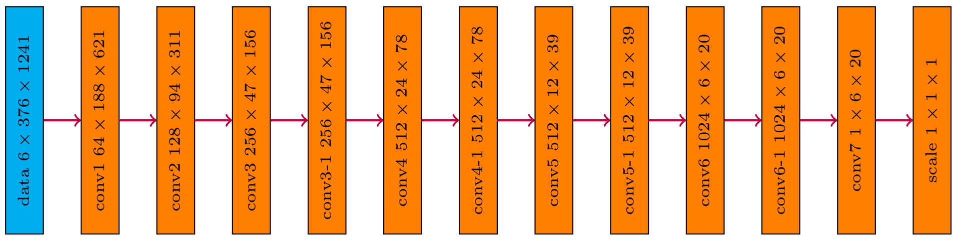
There is a last fully connected layer that reduces a network to a single output, which we are interested to learn. We use euclidean loss to train the network. Total number of parameters is about 624 million.

3.4.2 FlowNet

Fully-convolutional network is a recent trend in the field. They are known to train better and have less parameters vs. their fully connected counterparts [?]. The geometry of the network is described in Table 3.2. Each convolutional layer is followed by the non-linearity. We chose this architecture since it proved successfull for optical flow estimation task and has pre-trained model readily available.

Layer	**Receptive Fied Size**	**Padding**	**Stride**	**Number of Channels**
conv1	7×7	3	2	64
conv2	5×5	2	2	128
conv3	5×5	2	2	256
conv3-1	3×3	1	1	256
conv4	3×3	1	2	512
conv4-1	3×3	1	1	512
conv5	3×3	1	2	512
conv5-1	3×3	1	1	512
conv6	3×3	1	2	1024
conv6-1	3×3	1	1	1024
conv7	3×3	1	2	1

Table 3.2: Geometry of the flownet based network



We use Eucledian loss to train the network. The total number of parameters is 24 million.

3.5 Recurrent Neural Networks

Many problems require passing information through time. Since the traditional neural networks are stateless, they are poorly suited for this purpose. Recurrent Neural Networks have loops in them, so that the information can persist.

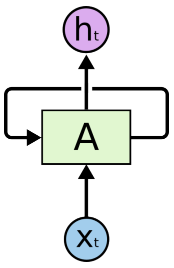


Figure 3.2: RNN

In the Figure 3.2 network A accepts input x_t and outputs h_t. The loop allows the network to pass information from one step of to another.

The RNN may be thought of as a chain of multiple copies of the same neural network where each node passes a message to its successor. This is called *network unrolling*, depicted in Figure 3.3. Unrolled network stresses the relation of the RNN to the sequential data streams. This is an architecture of choice when modeling such data.

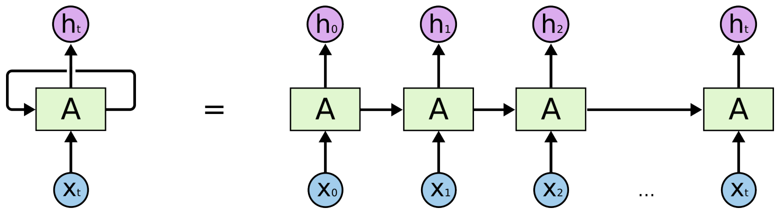


Figure 3.3: An unrolled RNN

It turns out that the vanilla RNN are hard to train due to the exploding/imploding gradients. Fortunately, Long Short Term Memory Networks (LSTM), which is a special kind of RNN addresses these issues.

LSTM were introduced by [HS97], they are specifically designed to model long term dependencies. Similarly to RNN, LSTM posses a chain-like structure. Each repeating node has an internal structure depicted in Figure 3.4.

To use the LSTM for the image data we combine the convolutional network with the LSTM. X_t vectors are produced by the ZF convnet. This is a fairly common representation used previously. The convnet and the LSTM are trained jointly.

The geometry of the network is as follows:

- conv1: (96, 6, 7, 7)
- conv2: (384, 48, 5, 5)
- conv3: (512, 384, 3, 3)

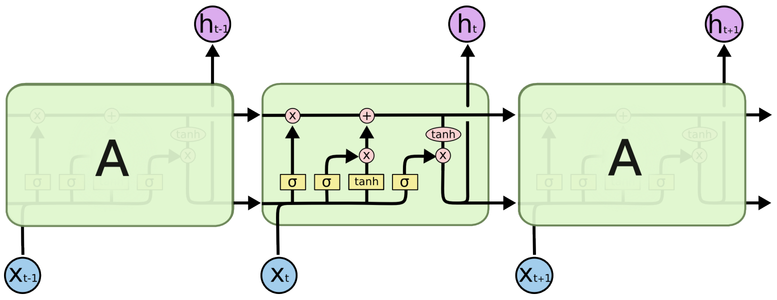


Figure 3.4: Long Short Term Memory

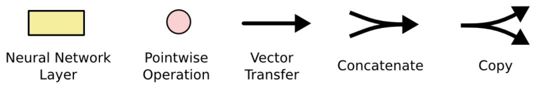


Figure 3.5: A notation used in Figure 3.4

- conv4: (512, 256, 3, 3)
- conv5: (384, 256, 3, 3)
- fc6: (4096, 34560)
- lstm1: (1024, 4096)
- fc-final: (1, 256)

Total number of trainable parameters is about 150 millions.

3.6 Experiments

3.6.1 Data-set

We train and test on the KITTI data-set [GLSU13]. The data-set consists of 11 sequences with ground truth data. We arbitrarily use sequence 00 for testing and sequences 01-10 for training. There are 4540 and 18650 images in the test and the train sets respectively. We only use the data from the left color camera. Figure 3.7 depicts the distribution of the scale values over the train and the test sets.

3.6.2 Random Forest

3.6.3 Feature Extraction

In order to train the random forest we need to represent subsequent image pairs as feature vectors, which are created as follows: we extract and match sparse salient points in both input images. The output of this stage is a set of a corresponding pixel locations. Then, we compute the point displacement magnitudes. Finally, we bin all the interest pundits according to a grid and create histogram of displacement magnitudes for each

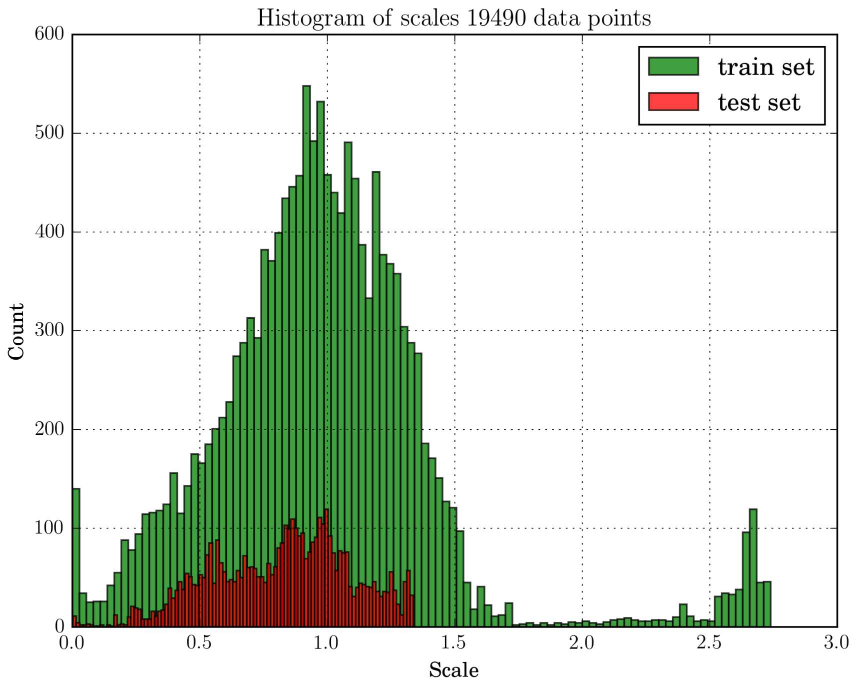


Figure 3.6: The train set

Figure 3.7: Scale distributions

bin. Concatenating all histograms together produces a feature vector. We use Harris corners and square 11×11 patches as corner descriptors. Sum of square differences is used as a distance measure with the winning pair declared a match. To prune the outliers we fit the fundamental matrix into the matched corner sets and remove the corners that do not agree with the model. Figure 3.8 shows a typical example of extracted and matched corners.

Some statistics of the features is presented in the Figure 3.9. We expect the peaks, that correspond to a closer image regions have a distributions shifted to the right (i.e., larger displacements) and the peaks that correspond to a regions farther away should be closer to zero. This behavior can be observed especially well for the feature vectors that correspond to larger camera displacements (e.g. Figure 3.9c).

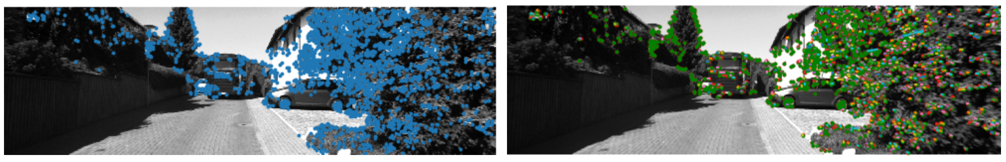
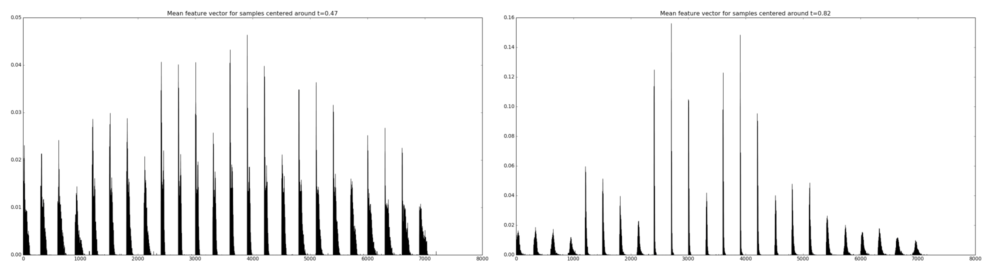
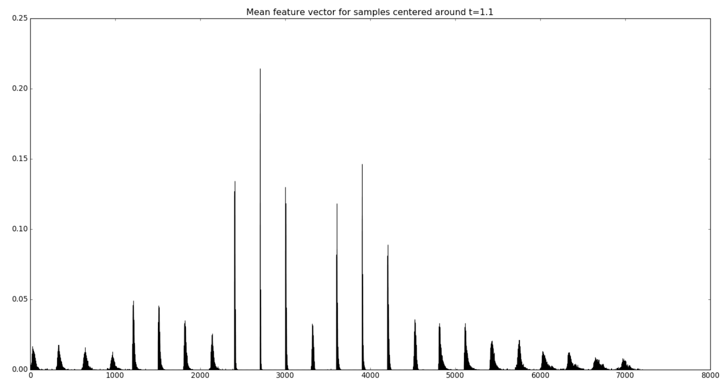

(a) (b)

Figure 3.8: Typical corner extraction and matching. Figure a shows the raw extracted corners, while Figure b shows pruned and matched corners.

We bin each image into 6×4 grid. For each bin in the image we compute the



(a) (b)



(c)

Figure 3.9: Average feature vectors for samples centered around the specific camera translation magnitude. Each peak corresponds to a grid cell (e.g. here the grid is 4 rows by 6 columns by 300 bins, so the feature vector is of the dimension 6*4*300=7200). The grid is sampled in a column-major mode. So the first four peaks correspond to the leftmost column of the image grid.

histogram of corner disparities. By disparity we denote the displacement of the corner in the image. We use 300-bin histogram for disparities (e.g, feature vector length is 7200).

We use the extracted features to fit a random forest, by means of recursive note splitting with sub-node variance minimization. The mean absolute error is 0.174 meters with a standard deviation of 0.148 meters.

3.7 Neural networks

We use Caffe [JSD$^+$14] framework to train and test our models.

ZF We train using vanilla SGD for 100 epochos and decrease learning rate by factor of .1 every 10 epochs. We train the network from scratch.

Flownet We also train using vanilla SGD for 100 epochs decreasing learning rate by .1 every 10 epochs. We start from pre-trained weigts of [FDI$^+$15].

LSTM TBD

3.8 Results

Let the ground truth sequence of camera motion scales by $\{y_i\}_{i=0}^N$. Let the predicted scales for the same sequence be $\{\hat{y}_i\}_{i=0}^N$. We report the mean absolute error:

$$\mu = \frac{1}{N}\sum_0^N |y_i - \hat{y}_i|$$

		Random Forest	ZF	FlowNet	LSTM ZF
μ	$[meters]$	.237	.200	**.103**	.198
σ	$[meters]$	.229	.161	**.08**	.147

Table 3.3: Experimental Results

Chapter 4

Conclusion and open questions

This kind of chapter can include may different things (or only some of them):

- Discussion of results
- Conclusions from the results or from the process in general
- Open questions for future research, resulting from the research performed or from the results obtained

But not things like the bibliography or other back matter which is generated outside of this chapter.

4.1 Some conclusion

Here is what I conclude.

4.2 Some open questions

A question in brief. In **??** we explored a certain subject, but what about this-or-that idea? Perhaps it is worth exploring. Can one produce interesting results?

A second question in brief. A broader exposition of the question and indications of directions or ideas regarding its resolution.

Bibliography

[BTV06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3951 LNCS:404–417, 2006.

[BYK13] Hernán Badino, Akihiro Yamamoto, and Takeo Kanade. Visual odometry by multi-frame feature integration. *Proceedings of the IEEE International Conference on Computer Vision*, (Cvad 13):222–229, 2013.

[FDI$^+$15] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[GGAM14] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014.

[GLSU13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[Gov09] Natasha Govender. Evaluation of Feature Detection Algorithms for Structure from Motion. *Csir*, 2009.

[GZS11] Andreas Geiger, Julius Ziegler, and Christoph Stiller. StereoScan: Dense 3d reconstruction in real-time. *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 963–968, 2011.

[HFF09] Johan Hedborg, Pe Forssén, and Michael Felsberg. Fast and accurate structure and motion estimation. *5th International Symposium on Advances in Visual Computing: Part I*, pages 211–222, 2009.

[Hor87] Berthold K P Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629, 1987.

[HP87] C G Harris and J M Pike. 3D Positional Integration from Image Sequences. *Procedings of the Alvey Vision Conference 1987*, pages 233–236, 1987.

[HS88] Chris Harris and Mike Stephens. A Combined Corner and Edge Detector. *Procedings of the Alvey Vision Conference 1988*, pages 147–151, 1988.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[HZRS14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer, 2014.

[JSD$^+$14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[KAS10] Kurt Konolige, Motilal Agrawal, and Joan Solà. Large-scale visual odometry for rough terrain. *Springer Tracts in Advanced Robotics*, 66(STAR):201–212, 2010.

[Kle07] Parallel tracking and mapping for small AR workspaces. *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR*, 2007.

[KRD08] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. iSAM: Incremental smoothing and mapping. 24(6):1365–1378, 2008.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[KSS11] Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2969–2976, 2011.

[Low04] David G Lowe. Distinctive image features from scale invariant keypoints. *Int'l Journal of Computer Vision*, 60:91–11020042, 2004.

[LSLR16] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2024–2039, 2016.

[Mor80] Hans Peter Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. *tech. report CMU-RI-TR-80-03*, page 175, 1980.

[MS06] Annalisa Milella and Roland Siegwart. Stereo-based ego-motion estimation using pixel tracking and iterative closest point. *Proceedings of the Fourth IEEE International Conference on Computer Vision Systems, ICVS'06*, 2006(Icvs):21, 2006.

[NDL$^+$05] Feng Ning, Damien Delhomme, Yann LeCun, Fabio Piano, Léon Bottou, and Paolo Emilio Barbano. Toward automatic phenotyping of developing embryos from videos. *IEEE Transactions on Image Processing*, 14(9):1360–1371, 2005.

[Nis]

[NLD11] R A Newcombe, S J Lovegrove, and A J Davison. {DTAM}: Dense Tracking and Mapping in Real-Time. *Int. Conf. on Computer Vision (ICCV)*, pages 2320–2327, 2011.

[OMSM03] Clark F. Olson, Larry H. Matthies, Marcel Schoppers, and Mark W. Maimone. Rover navigation using stereo ego-motion. *Robotics and Autonomous Systems*, 43(4):215–229, 2003.

[PPFM15] Mikael Persson, Tommaso Piccini, Michael Felsberg, and Rudolf Mester. Robust stereo visual odometry from monocular techniques. *IEEE Intelligent Vehicles Symposium, Proceedings*, 2015-Augus:686–691, 2015.

[RD06] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3951 LNCS:430–443, 2006.

[SEZ$^+$13] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[SF11] Davide Scaramuzza and Friedrich Fraundorfer. Visual Odometry. (June), 2011.

[SFPS09] Davide Scaramuzza, Friedrich Fraundorfer, Marc Pollefeys, and Roland Siegwart. Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1413–1419. IEEE, 2009.

[SG] Shiyu Song and Clark C Guest. Parallel , Real-Time Monocular Visual Odometry.

[SLJ$^+$15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[ST94] Jianbo Shi Jianbo Shi and Carlo Tomasi. Good features to track. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, (December):593–600, 1994.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[TMHF00] Bill Triggs, Philip F. Mclauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle Adjustment - a Modern Synthesis. *Vision algorithms: theory and practice. S*, 34099:298–372, 2000.

[ZDL16] Dingfu Zhou, Yuchao Dai, and Hongdong Li. Reliable scale estimation and correction for monocular visual odometry. In *Intelligent Vehicles Symposium (IV), 2016 IEEE*, pages 490–495. IEEE, 2016.

[ZF13] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

... ואילו עמוד זה של התקציר המורחב בעברית, שהינו העמוד השני בתקציר, ממוספר ב-יי ונמצא ראשון בקובץ ה-PDF --- לפני עמוד י. המטרה היא, שבעת ההדפסה לא יהיה צורך להפוך דפים, לשנות את סדרם וכולי - רק להדפיס ולכרוך.

תקציר

כאן יבוא תקציר מורחב בעברית (כאשר שפת החיבור העיקרית היא אנגלית). היקף התקציר יהיה 1000-2000 מילים. התקציר יהווה שלמות בפני עצמו ויהיה מובן לקורא בעל ידיעות כלליות בנושא. התקציר ייכתב במשפטים מקושרים שלמים, בדרך-כלל אין לציין בו מקורות ספרותיים וציטוטים. אין להתייחס למספר של פרק, סעיף, נוסחה, ציור או טבלה שבגוף החיבור, ואין להשתמש בקיצורים, סמלים ומונחים לא מקובלים, אלא אם יש בתקציר די מקום לזיהויים.

אם צריך לשים קישור בתקציר העברי, (למשל לצורך ציטוט מקור, כאשר משתמשים גם ב-hyperref שיוצר קישורים בתוך ה- עבור ציטוטי מקורות), אפשר לעטוף אותו בתוך {}L\ , כך: [?].

תת-חלק בתקציר המורחב

תוכן מקוצר לגבי נושא מסוים.

תת-חלק נוסף בתקציר המורחב

תוכן מקוצר לגבי נושא שני. התייחסות למושג מסוים שהחיבור בוחן. וכולי וכולי.

נקודה מעניינת לגבי העמודים בעברית

שימו לב כי העמודים בעברית אמורים להיות מיוצרים בסדר ה"הפוך", הווה אומר העמוד האחרון בקובץ ה-PDF הוא הכריכה העברית, לפניו השער העברי, ודפי התקציר צריכים להו-פיע בסדר הפוך (וכן במספור רומי, לפי נהלי הטכניון). כך עמוד זה, העמוד הראשון בתקציר יש לצפות, ממוספר ב-i ונמצא אחרון בקובץ ה-PDF...

$^{2\circ}$§©© ÷ ^{1}¬´¸´± ¸ ©^{1}§^{12}¶Ⓡ°⏀,´Δ¬¨¬®£^{2}⏀Ⓡ§^{1}.¶ ¸·§®

ציון אם העבודה פורסמה בכתבי עת או הוצגה בכנסים. הרשימה תכתב בהתאם לכללי הציטוט (כולל הכותר ושמות השותפים). במקרה של שיתוף פעולה, פירוט תרומת המגיש.

אני מודה לטכניון על התמיכה הכספית הנדיבה בהשתלמותי.

כותרת החיבור

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
שם התחום המדעי

שם המחבר

הוגש לסנט הטכניון --- מכון טכנולוגי לישראל
2017 ÷ ÷ ^{12}” *gitd* ©° ¸¨

כותרת החיבור

שם המחבר