

195?

# Recovering Patch Parameters from the Optic Flow with Auto Associative Neural Networks

Anuj Dev, Ben J.A. Kröse, Leo Dorst, Frans C.A. Groen

**Keywords:** Neural Networks, Image Understanding, Structure from Motion

## Introduction

The optic flow is the vector field formed by the projection of the 3D-motion in the environment on the image plane of the observer. We are interested in a well known, but only partially solved, problem in computer vision: the extraction of scene structure and 3D-motion from the optic flow which is called *Structure from Motion*.

In our approach the scene is modelled locally as a planar patch, to introduce redundancy in the optic flow, so that the local Taylor expansion of the optic flow only contains terms up to the second order. The problem is thus to find the mapping  $\mathcal{W}$  from the Taylor coefficients  $(\alpha_1, \dots, \alpha_8)$  to the unknown parameters of the planar patch: the orientation  $\phi$ , its translational velocity  $\mathbf{t}$  and its rotational velocity  $\Omega$ . Since our camera mapping is unknown we want to estimate the mapping  $\mathcal{W}$  from a set of example data points  $\mathbf{v}^i$  which form the training set. However, from the literature it is known that the mapping  $\mathcal{W}$  is not a function in the sense that it is a  $m \rightarrow 1$  mapping. This means that standard function approximators like Multi Layer Perceptrons (MLP) are unable to approximate  $\mathcal{W}$ . We will show how such mappings can be learned by auto associative MLP and demonstrate this method on the structure from motion problem.

## Learning a manifold's parameterization

First we generalise the mapping by defining an input vector  $\mathbf{x} = (\alpha_1, \dots, \alpha_8)$  in the input space  $\mathcal{X}$  and an output vector  $\mathbf{y} = (\phi, \mathbf{t}, \boldsymbol{\Omega})$  in the output space  $\mathcal{Y}$ . With this notation a data point  $\mathbf{v}^i$  is a concatenation of an input vector  $\mathbf{x}^i \in \mathcal{X}$  and output vector  $\mathbf{y}^i \in \mathcal{Y}$ . Geometrically, a mapping  $\mathbf{x} \rightarrow \mathbf{y}, \mathcal{X} \rightarrow \mathcal{Y}$ , can now be viewed as a manifold  $\mathcal{M}$  which resides in the product space  $\mathcal{V} = \mathcal{X} \times \mathcal{Y}$ . The manifold will be denoted by its parameterization  $\mathcal{M} : (\mathbf{F}(\mathbf{s})) = (\mathbf{F}_{\mathbf{x}}(\mathbf{s}), \mathbf{F}_{\mathbf{y}}(\mathbf{s}))$  where  $\mathbf{F}(\mathbf{s}) \in \mathcal{V}$ ,  $\mathbf{F}_{\mathbf{x}}(\mathbf{s}) \in \mathcal{X}$  and  $\mathbf{F}_{\mathbf{y}}(\mathbf{s}) \in \mathcal{Y}$  are the vector valued parameter functions and  $\mathbf{s}$  the parameter vector which resides in the parameter space  $\mathcal{S}$  with  $\dim \mathcal{S} \leq \dim \mathcal{X}$ .

## Auto Associative Multi Layer Perceptrons

Since multi layer perceptrons MLP are general function approximators, we could learn the vector valued function  $\mathbf{F}(\mathbf{s})$ . The problem is that there are no learning examples available in the form of  $(s^i, v^i)$  pairs. But if we assume that there is a vector valued function <sup>1</sup>  $\mathbf{F}(\mathbf{s})$  which is bijective onto  $\mathcal{M}$ , <sup>2</sup> then there exists a vector valued function  $\mathbf{G} = \mathbf{F}^{-1}$  which is also bijective so that

$$\mathbf{F}(\mathbf{G}(\mathbf{v})) = \mathbf{v} \quad \forall \mathbf{v} \in \mathcal{M}$$

The trick is now to learn both functions  $\mathbf{F}$  and  $\mathbf{G}$  within one MLP. We train the MLP to reconstruct its input vector  $\mathbf{v}$  on the output layer and create a *bottleneck* by taking the number of hidden nodes the same as the dimension of the manifold  $\mathcal{M}$ . Note that this dimension is lower than  $\dim \mathcal{V}$ , hence the *bottleneck*. In this way the hidden nodes span the parameter space  $\mathcal{S}$ , the mapping from the input layer to the hidden layer represents  $\mathbf{G}$  and the mapping from the hidden layer to the output layer represents  $\mathbf{F}$ . These networks are known as auto associative MLP [2] and shown in figure 1.

---

<sup>1</sup>Implicitly we say that there is no unique parameterization of a manifold.

<sup>2</sup>Geometrically, this means that the manifold  $\mathcal{M}$  cannot have any self intersections.

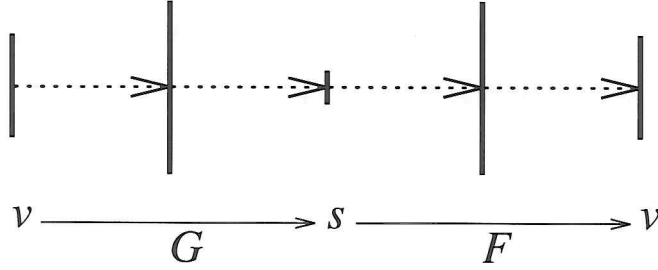


Figure 1: MLP architecture which reconstructs its input. The dimensionality of the middle layer,  $dim\mathcal{S}$ , is lower than the dimension of the input and output layer,  $dim\mathcal{V}$ , and represents the parameter space. Note that we have drawn a five layer MLP which means that every vector valued function  $\mathbf{F}, \mathbf{G}$  can be learned.

### Approximating mappings from parameterized manifolds

Given a parameterization of a manifold we come to the second objective which is to give an approximation  $\tilde{\mathbf{y}}$  of the mapping for a given  $\tilde{\mathbf{x}}$ . Since the mapping need not to be  $m \rightarrow 1$ , a function, there are several possible values of  $\tilde{\mathbf{y}}$  "correct". These values  $\mathbf{y}^i$  correspond to:

$$\tilde{\mathbf{y}}^i = \mathbf{F}_{\mathbf{y}}(\mathbf{s}^i) \quad \forall \mathbf{s}^i | \mathbf{F}_{\mathbf{x}}(\mathbf{s}^i) = \tilde{\mathbf{x}}$$

This means that there has to be an additional constraint on the desired mapping: we will choose the value which is in some metric closest to a given point  $\mathbf{v}^p = \mathbf{F}(\mathbf{s}^p) \in \mathcal{M}$ <sup>3</sup>. For the gradient based method described next we will take the approximation which is closest, in euclidian distance, in the input space:  $\mathbf{s}^i = \min_i (\|\mathbf{F}_{\mathbf{x}}(\mathbf{s}^i) - \mathbf{F}_{\mathbf{x}}(\mathbf{s}^p)\|)$

### Gradient descend on the input distance

The basic idea is to walk over the manifold from the starting point  $\mathbf{s}^p$  toward the point where  $\mathbf{F}_{\mathbf{x}}(\mathbf{s}) = \tilde{\mathbf{x}}$ . This is done by changing the parameter vector  $\mathbf{s}$  so that the walking is in the direction of  $\tilde{\mathbf{x}}$ . This means that we want to minimize the function  $E$ :  $E = \|\tilde{\mathbf{x}} - \mathbf{F}_{\mathbf{x}}(\mathbf{s})\|^2 = \delta^2$  and boils

<sup>3</sup>Actually  $\mathbf{v}^p$  can be any point on the manifold but may be thought of as the previous *state* of the system that is being approximated by the AAMLPL.

down to a well known iterative scheme:  $\mathbf{s}_{n+1} = \mathbf{s}_n + \mu \delta \frac{\partial \mathbf{F}_x}{\partial \mathbf{s}}$  where  $\mu$  is a iterating constant. Note that  $\partial \mathbf{F}_x / \partial \mathbf{s}$  is the Jacobian of the last layers of the MLP and can be calculated from them.

## Simulations

We first give a definition of the reconstruction error

$$E_y = \frac{\|\mathbf{y} - \mathbf{F}_y(\mathbf{G}(\mathbf{v}))\|}{\dim \mathcal{Y}} \quad \forall \mathbf{v} \in \mathcal{M}$$

$E_y$  is a measure how well the manifold is approximated by the MLP. We have tested the approach with a perspective projection and unit focal length. The relation of the Taylor coefficients with the parameters of the planar patch is then known and given by [1]:

$$\alpha_1 = -\omega_y - t_x; \alpha_2 = t_z - \phi_x t_x; \alpha_3 = -\omega_z - \phi_y t_x; \alpha_4 = -\omega_y \phi_x t_z$$

$$\alpha_5 = -\omega_x - t_y; \alpha_6 = -\omega_z - \phi_x t_y; \alpha_7 = t_z - \phi_y t_y; \alpha_8 = -\omega_x \phi_y t_z$$

From this relation learning examples were generated by choosing random patch parameters (from the interval  $[-1, 1]$ ) and computing the Taylor coefficients  $\alpha_1, \dots, \alpha_8$ . We used a 10-16-8-16-10 network and trained it to reconstruct the 10 dimensional input consisting of  $(\alpha_1, \dots, \alpha_8, t_x/t_z, t_y/t_z)$ . An mean error  $E_y = 0.03$  was found, which indicates that a good parameterization of the manifold is found.

## References

- [1] Waxman A.M., Ullman S. 1985 *Surface Structure and Three Dimensional Motion from Image Flow Kinematics* The International Journal of Robotics Research. 4-3.
- [2] Demers D. 1993. *Non-linear dimension reduction*. in proceedings NIPS-5.