A REPORT

ON

ACCURATE STEREO BASED DEPTH ESTIMATION

By

Name of the student:	ID No.	Discipline(s)
Veerav Chebrolu	2013B4AA886H	ECE, Mathematics

Prepared in the partial fulfillment of the
Practice School II Course

AT

SAMSUNG R&D INSTITUTE INDIA, BANGALORE
A Practice School II Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

Abstract Sheet

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)

Practice School Division

Station: Samsung R&D Institute India, Bangalore **Centre:** Bangalore
Duration: 5 ½ months **Date of Start:** 6 - Jul - 2017
Date of Submission: 08 - Dec - 2017

Title of the Project: Accurate Stereo Based Depth Estimation

ID No./Name(s)/: 2013B4AA886H Veerav Chebrolu
Discipline(s) of the Student(s) **B.E** Electronics and Communications, **MSc**. Mathematics.

Name(s) and: Designation(s) of the expert(s) Dr. Rituparna Sarkar, Chief Engineer

Name(s) of the: PS Faculty Ms. Anita Ramachandran

Key Words: disparity, epipolar geometry, image rectification, matching cost, cost aggregation, convolutional neural network, active depth

Project Areas: Computer Vision

Abstract:

The report is based on stereo algorithms which are used to obtain a depth map. It covers basic concepts and mainly focuses on four main stages of processing a stereo algorithm namely: matching cost, cost aggregation, optimization and disparity refinement. The report then looks upon stereo matching based on convolutional neural networks. It looks upon two methods namely mc-cnn which computes matching cost using convolutional neural networks and cnn-crf which is an end to end method which exploits the benefits of convolutional neural network and conditional random fields where the CNN's compute expressive features for matching and distinctive color edges which is used to compute unary and pairwise cost for the CRF.

Signature(s) of Student(s) Signature of PS Faculty

Date: 8th December 2017

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)

Acknowledgement

I would like to thank my mentor Dr. Rituparna Sarkar for guiding and helping me through the project. My heartfelt gratitude to Samsung R&D Institute India, Bangalore that provided the opportunity to intern and pursue this project. I would like to thank my manager Dr. Narasimha Pai for providing the required support. I would like to thank my Practice School - II instructor Ms. Anita Ramachandran for conducting the evaluations and ensuring that we are up to date with the work. I would also like to thank BITS Pilani University for providing me the opportunity to attend Practice School and gain ample experience from the same. I would like to thank my colleagues and friends who have been supportive in this project.

Table of Contents

1. INTRODUCTION

Stereo Vision is the extraction of depth information from a pair of images capturing the same scene from two different angles. By means of triangulation, we can find the depth information, if we are able to find the corresponding (homologous) points on the two images. The search for the corresponding points on the two images is constrained by **epipolar geometry**. This essentially ensures that the corresponding points of the two images lie on a straight line known as **epipolar line**. Thus a 2D search is reduced to to a 1D search.

In order to make the search for corresponding points easier, the two images are transformed in such a way such that the epipolar lines become the corresponding horizontal lines. This means that any corresponding points between the two images lie on the same row. This is known as image **Rectification**.

The shift in the number of pixels of a point from left image to right image is known as **disparity** of that point. The depth of a point can be computed from the disparity of a point given by the relation $\boldsymbol{z = \frac{B.f}{d}}$ where 'B' is the baseline, 'f' is the focal length and 'd' is the disparity of the point. Baseline is defined as the distance between two camera centres.

Thus the goal of a stereo problem is to find the disparity map, through which depth map can be computed.

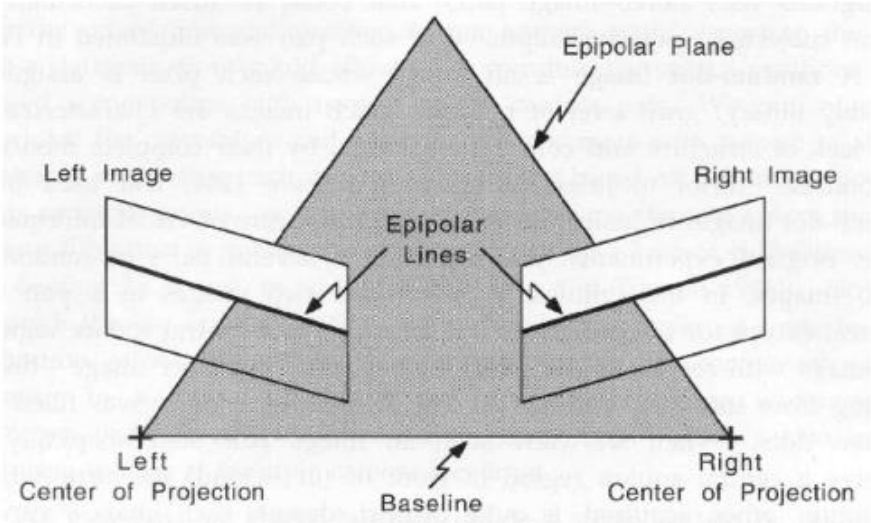
A typical stereo matching problem essentially consists of four steps: Matching cost computation, Cost Aggregation, Optimisation, Refinement. The rest of the report expands on these four steps.

2. BASIC CONCEPTS

Epipolar (Stereo) Geometry

- **Epipole**: The image in one camera of the projection centre of the other camera is called epipole.

- **Left Epipole**: The projection of O_r on the left image plane.
- **Right Epipole**: The projection of O_l on the right image plane.
- **Epipolar plane**: The plane defined by P, O_l , O_r.
- **Epipolar line**: The intersection of epipolar plane with image plane.



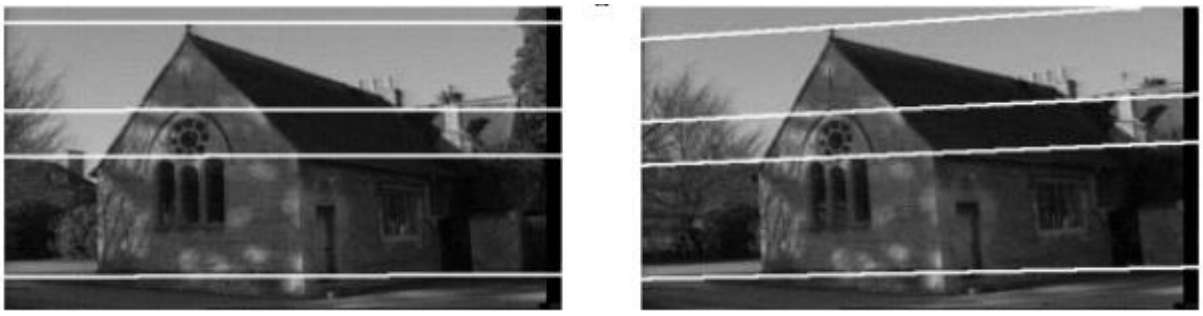
Stereo Basics

- The camera frames are related by translation vector $T = O_r - O_l$
- The relation between P_l and P_r (projection of P in left and right frames) is given by $P_r = R(P_l - T)$
- The usual equations of perspective projection define the relation between 3D points and their projections.

$$p_l = \frac{f_l}{z_l} P_l \qquad p_r = \frac{f_r}{z_r} P_r$$

Epipolar constraint

- Given p_l , P can lie anywhere on the ray from O_l through P_l.
- The image of this ray in the right image is the epipolar line through the corresponding point p_r.
- The correct match must lie on the epipolar line. This establishes a mapping between the points in the left image and the right image. The search of the correspondences is reduced to a 1D problem.



STEREO ALGORITHMS

3. PREVIOUS WORK

Matching Costs

The first component of any dense stereo matching algorithm is a similarity measure that compares pixel values in order to determine how likely they are to be in correspondence.
The most common pixel-based matching costs include squared intensity differences (**SSD**) (Hannah 1974) and absolute intensity differences (**SAD**) (Kanade 1994).
More robust measures such as truncated quadratics and contaminated Gaussians have been proposed (Black and Anandan 1996, Black and Rangarajan 1996, Scharstein and Szeliski 1998). These measures are useful because they limit the influence of mismatches during aggregation.
Other traditional matching costs include normalized **cross-correlation** (Hannah 1974,Bolles et al. 1993), which behaves similarly to sum-of-squared-differences (SSD), and binary matching costs (i.e., match / no match) (Marr and Poggio 1976), based on binary features such as edges (Baker and Binford 1981, Grimson 1985) or the sign of the **Laplacian** (Nishihara 1984). Because of their poor discriminability, simple binary matching costs are no longer used in dense stereo matching.
Some costs are insensitive to differences in camera gain or bias, for example gradient-based measures (Seitz 1989, Scharstein 1994), phase and filter-bank responses, filters that remove a regular or robust (bi-laterally filtered) means (Ansaret al.2004, Hirschmuller and Scharstein 2009), and non-parametric measures such as **rank** and **census transforms** (Zabih and Woodfill 1994) or entropy (Zitnick et al. 2004, Zitnick and Kang 2007).
The **census transform**, which converts each pixel inside a moving window into a bit vector representing which neighbors are above or below the central pixel, is quite robust against large scale non-stationary exposure and illumination changes. It is also possible to correct for differing global camera characteristics by performing a pre-processing or iterative refinement step that estimates inter-image bias-gain variations using global regression (Gennert 1988), histogram equalization (Cox et al.1995), or mutual information (Kimet al.2003,Hirschmuller).
In order to compensate for sampling issues, i.e., dramatically different pixel values in high-frequency areas, Birchfield and Tomasi (1998) proposed a matching cost that is less sensitive to shifts in image sampling. Rather than just comparing pixel values shifted by integral amounts

(which may miss a valid match), they compare each pixel in the reference image against a linearly interpolated function of the other image. More detailed studies of these and additional matching costs are explored in (Szeliski and Scharstein 2004,Hirschmuller and Scharstein 2009). In particular, if there is a significant exposure or appearance variation between images that we are matching, some of the more robust measures that performe, such as the census transform (Zabih and Woodfill 1994), bilateral subtraction (Ansar et al 2004), or hierarchical mutual inforamtion (Hirschmuller 2008), should be used.

Aggregation of cost

Local and window-based methods aggregate the matching cost by summing or averaging over a support region. A support region can be either two dimensional at a fixed disparity (fronto – parallel surface) or three dimensional x-y-d space (supporting slanted surfaces). Two-dimensional aggregation has been implemented using square windows or Gaussian convolution (traditional), multiple windows anchored at different points, i.e., shiftable windows (Arnold 1983, Bobick and Intille 1999), windows with adaptive sizes (Okutomi and Kanade 1992, Kanade and Okutomi 1994, Veksler 2001, Kang et al. 2001), and windows based on connected components of constant disparity (Boykov et al. 1998).
Three-dimensional support functions that have been proposed include limited disparity difference (Grimson 1985), limited disparity gradient (Pollard et al. 1985), and Prazdny's coherence principle (Prazdny 1985). Aggregation with a fixed support region can be performed using 2D or 3D convolution.

Disparity Computation and optimization

Local Methods
In local methods, the emphasis is on the matching cost computation and on the cost aggregation steps. Computing the final disparities is trivial: simply choose at each pixel the disparity associated with the minimum cost value. Thus, these methods perform a local "winner- take-all" (WTA) optimization at each pixel. A limitation of this approach (and many other correspondence algorithms) is that uniqueness of matches is only enforced for one image (the reference image), while points in the other image might get matched to multiple points.

Global Optimization
In contrast, global methods perform almost all of their work during the disparity computation phase, and often skip the aggregation step. Many global methods are formulated in an energy-minimization framework. The objective is to find a disparity function'd' that minimizes a global energy.

$$E(d) = E_{data}(d) + \lambda E_{smooth}(d)$$

The data term measures how well the disparity function agrees with the input image pair. The smoothness term encodes the smoothness assumptions made by the algorithm. The smoothness term is often restricted to measuring the differences between neighboring pixel's disparity.

Refinement of disparity

Occluded areas can be detected using cross-checking (comparing left-to-right and right- to-left disparity maps). A median filter can be applied to "clean up" spurious mismatches, and holes due to occlusion can be filled by surface fitting or by distributing neighboring disparity estimates.

4. Benchmarking the Implemented Method

Benchmarking of the stereo algorithm based on the **census transform** has been done. The benchmarking is done on Middlebury training dataset. The stereo algorithm that has been benchmarked consists of four steps mentioned as above. The first step that is **matching computation** is computed by calculating **absolute difference** and **census transform**.
The second step that is **cost aggregation** is computed by calculating **cross based support arm region.** The third step that is **optimization** is done by implementing **semi global matching.** The fourth step that is **disparity refinement** is done by performing **left right consistency**. Outliers are further classified into **mismatch** points and **occluded** points. Occluded points are filled by region voting and mismatch points are filled by interpolation strategy.

5. Stereo Matching Using Convolutional Neural Network for initializing Matching Cost

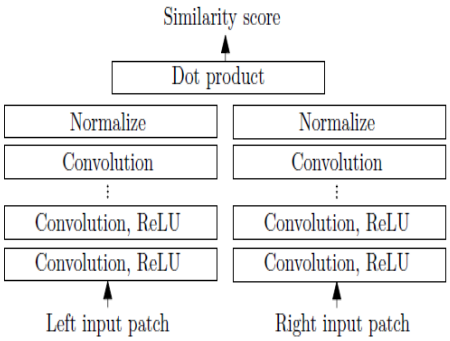
This method is based on the paper "Stereo Matching by Training Convolutional Neural Network to Compare Image Patches". The method focuses on the first step of stereo i.e**. matching cost**.
The problem is approached by learning a similarity measure on small image patches using a convolutional neural network. Training is carried out in a supervised manner by constructing a binary classification data set with examples of similar and dissimilar pairs of patches.
The output of the convolutional neural network is used to initialize the stereo matching cost. A series of post-processing steps follow: cross-based cost aggregation, semi global matching, a left-right consistency check, sub pixel enhancement, a median filter, and a bilateral filter.

Dataset:
Ground truth disparity maps from either the KITTI or Middlebury stereo data sets are used to construct a binary classification data set. At each image position where the true disparity is known, one negative and one positive training example is extracted ensuring that the data set contains an equal number of positive and negative examples. A positive example is a pair of patches, one from the left and one from the right image, whose center pixels are the images of the same 3D point, while a negative example is a pair of patches where this is not the case.

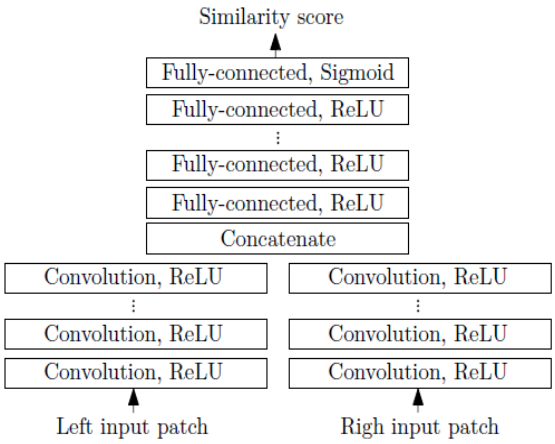
Network Architectures

Two types of network architectures are used for computing the similarity measure namely the **fast architecture** and the **accurate architecture**. In both the cases, the input is left and right image patches and the output is similarity measure. Both the architectures contain a trainable feature extractor that represents each image with a **feature vector**. In both the methods, similarity is measured on the feature vectors instead of the raw intensity values. The fast architecture uses a fixed similarity measure whereas the accurate architecture tries to learn the similarity measure.
Both the architectures are **Siamese** network. The two sub networks consist of a number of convolutional layers followed by rectified linear units. In the fast architecture the similarity is measured by **cosine** similarity which is basically normalization and dot product of the feature vector. The fast architecture is trained by minimizing **hinge loss**. The loss is computed by considering a pair of examples, one belonging to positive class and the other belonging to negative class. The loss is zero if the similarity of the positive example is greater than the similarity of the negative example by a margin m. In the paper, it is set as 0.2.
In the accurate architecture, the similarity is measured by considering a **fully connected layer** replacing the cosine similarity in the fast architecture. The resulting two vectors from the sub network is concatenated and forward propagated through the fully connected layer followed by a rectified linear unit. The last fully-connected layer produces a single number which, after being transformed with the sigmoid nonlinearity, is interpreted as the similarity score between the input

patches. The network was trained using a **binary cross entropy** function instead of using a hinge loss. The paper mentions the decision to use different loss functions was based on empirical evidence, since experiments showed that that binary cross entropy performed better on accurate architecture.



Fast architecture



Accurate architecture

Stereo Method

The matching cost is obtained by $-s(< P^L$ **(P),** P^R**(p-d)** $>)$ where $s(< P^L$ **(P),** P^R**(p-d)** $>)$ is the output of the network run on input patches P^L **(P),** P^R**(p-d)**. Then it is post processed by considering the steps: **Cost aggregation** based on cross based arms which averages the matching cost over a neighborhood that is adaptively selected. Then **semi global matching** is performed which introduces a smoothing constraint. Then **sub pixel enhancement** is applied to obtain a higher resolution disparity map by considering a quadratic curve for fitting the neighboring cost. The final steps involve refinement by using a 5x5 **median filter** following with a **bilateral filter**.

6. End-To End Training Using Hybrid CNN-CRF Model

This method is an end to end stereo method which uses hybrid approach combining CNN and CRF. The CNNs compute expressive features for matching and distinctive color edges, which in turn are used to compute the unary and pairwise costs of the CRF.
Unary-CNN computes local features of both images which are then compared in a fixed correlation metric. Pairwise-CNN can additionally estimate contrast-sensitive pairwise costs in order to encourage or discourage label jumps. Using the learned unary and pairwise costs, the CRF tries to find a joint solution optimizing the total sum of all unary and pairwise costs in a 4- connected graph.
The method does not to use any post processing in order to show that most of the performance gain through post-processing can be covered by a well-trained CRF model.
The Unary-CNN straightforwardly generalizes manually designed matching costs such as those based on differences of colors, local binary patterns, Census transform , etc. The Pairwise-CNN generalizes a contrast-sensitive regularizer

CNN-CRF Model

Unary CNN

The Unary CNN computes dense image features for left and right images, where both the instances of unary CNN share the same parameters. For each pixel these extracted feature vectors are then correlated with all the possible disparities to form a correlation volume.
3 layers and **7** layers are used in unary cnn with **100** filters in each layer. The filter size of first layer is **3x3** and the others is **2x2**. **Tanh** activation function is use.

Correlation

The cross-correlation of features ϕ^0 and ϕ^1 extracted from the left and right image, respectively, is computed as:

$$p_i(k) = \frac{e^{\langle \phi_i^0, \phi_{i+k}^1 \rangle}}{\sum_{j \in \mathcal{L}} e^{\langle \phi_i^0, \phi_{i+j}^1 \rangle}} \quad \forall i \in \Omega, \forall k \in \mathcal{L}.$$

CRF- Model

The CRF model optimizes the total cost of the complete disparity labels:

$$\min_{x \in \mathcal{X}} \left(f(x) := \sum_{i \in \mathcal{V}} f_i(x_i) + \sum_{ij \in \mathcal{E}} f_{ij}(x_i, x_j) \right)$$

Where $f_i(x_i)$ is the unary matching cost and $fij(x_i, x_j)$ is the pairwise matching cost. We can define the pairwise matching cost using the model:

$$f_{ij}(x_i, x_j) = w_{ij} \rho(|x_i - x_j|; P_1, P_2).$$

Pairwise CNN

We use a CNN for estimating the edge weights for computing pairwise matching cost. We use a 3 layer CNN with 64 filters of size 3x3 with Tanh as activation function.

7. Quantitative Analysis

Error and Time Analysis on Middlebury (Test) dataset

Method	Bad 2	Bad4	AvgErr	Time
Middlebury-Test(nocc)				
MC-cnn-acrt	**8.08**	**4.91**	3.82	150s
MC-cnn-fst	9.47	6.7	4.37	1.69s
CNN-CRF	12.5	6.61	**3.02**	4.46s
SGBM	23.8	18.5	28.8	**0.23s**

Error and Time Analysis on Kitti dataset

Method	bad3	Time
Kitti Test		
MC-cnn-acrt	**3.33**	67s
MC-cnn-fst	4	1s
CNN-CRF	4.84	1.3s
SGBM	7.62	1.1s

Cross Transfer Learning

MC-CNN

Training	Testing	MC-cnn-act	MC-cnn-fst
Kitti	Kitti (bad3)	4	3.33
Middlebury	Kitti (bad3)	4.49	4.48
Middlebury	Middlebury(bad2)	8.08	9.47
Kitti	Middlebury(bad2)	14.19	13.7

CNN-CRF

Training	Testing	CNN-CRF
Kitti	Kitti (bad3)	4.84
Middlebury	Kitti (bad3)	5.35
Middlebury	Middlebury(bad 4)	6.61
Kitti	Middlebury(bad 4)	15.22

MC-CNN Error Analysis if the corresponding post-processing step is removed.

postprocessing	MC-cnn-acrt (Middlebury)	MC-cnn-fst (Middlebury)
Cross-based cost aggregation	10.63	9.87
Semi-global matching	**11.99**	**25.5**
interpolation	7.91	9.87
Subpixel enhancement	8.44	10.29
Median filter	7.91	10.16
Bilateral filter	7.96	10.39
No stereo	28.33	30.84
Full stereo	7.91	9.87

CNN-CRF Error Analysis breakup for corresponding step

Benchmark	**Method**	**CNN**	**+CRF**	**+Joint**	**PW**
Middlebury	CNN3	23.89	**11.18**	9.48	9.45
(bad4)	CNN7	18.58	**9.35**	8.05	7.88
Kitti	CNN3	28.38	**6.33**	6.11	4.75
(bad3)	CNN7	13.08	**4.79**	4.60	4.04

8. Qualitative Results

Image	mc-cnn-acrt	mc-cnn-fst	cnn-crf	SGBM
Adirondack	Cup handle is missing.	Cup handle is missing.	Cup handle is present.	Fattening at cup handle. Not smooth at chair handles.
ArtL	Thin structure like paint brush is missing at many places. Edges of the face is oversmoothend.	Thin structure like paint brush is missing at many places. Edges of the face is oversmoothend.	Performs comparatively better at the edges of the face and retains more thin structures like paintbrush.	Thin structure like paint brush is missing at many places. Missing disparities at top of the face.
JadePlant	Structure in front of the pot and plant stems are fattened. Shapes of the flowers are lost.	Structure in front of the pot and plant stems are fattened. Shapes of the flowers are lost.	Structure in front of the pot and plant stems are p properly obtained. Shapes of the flowers have been obtained at some places.	Structure in front of the pot and plant stems are missing and fattened. Shapes of the flowers are lost.
Motorcycle	Wheels are smooth. Spokes of the wheels are oversmoothend and cannot be differentiated. Some places have erroneous disparity around wheels.	Wheels are smooth. Spokes of the wheels are oversmoothend and cannot be differentiated. Some places have erroneous disparity around wheels.	Wheels are smooth. Spokes of the wheels are oversmoothend and cannot be differentiated. Cross structure at the back of the chair is properly obtained.	Tyres are not smooth at the edges. Spokes of the wheels are not clearly distinguishable.
MotorcycleE	There is not much difference between disparity map between	There is not much difference between disparity map between	There is not much difference between disparity map between	Disparity map is full of artifacts.

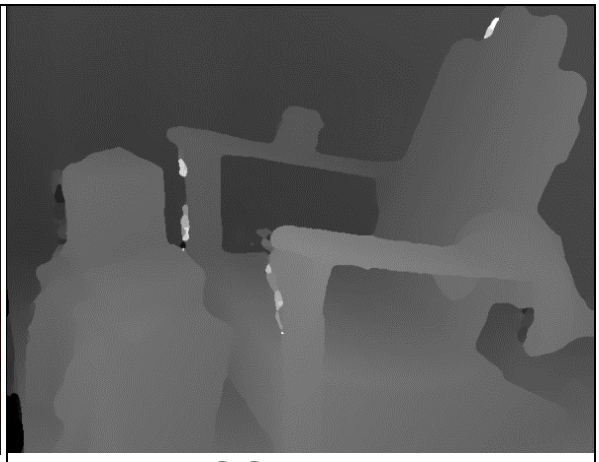
	motorcycle and motorcycleE.	motorcycle and motorcycleE	motorcycle and motorcycleE.	
Piano	Top of the wall is completely smooth.	Erroneous values of disparity at the edges around the legs of the stool. Not smooth on the wall of the room.	Error at the legs of the stool beneath piano and slightly more error compared to mccnn_acrt on the wall.	Lot of artifacts on the walls
PianoL	Error has slightly increased on the wall behind the lamp compared to piano.	Error has increased more on the wall behind the lamp.	Slight error in disparity on wall behind the lamp since in left image light is on and on right image light is off.	High error in disparity map. It completely fail especially near the lamp.
Pipes	Wall on the top which is a rough texture is nicely obtained. Small patches of errors in disparity at few places.	Similar to mccnn-acrt.	Wall on the top which is a rough texture is nicely obtained. Pipe structures are comparatively more rigid and not over smoothened at the edges.	Wall on the top which is a rough texture is nicely obtained. Not smooth at the edges of the pipes.
Playroom	Wall is smoothly obtained. Chair back is over smoothened.	Some missing pixels on the chair and on the wall.	Chair back is better.	Missing disparities on the wall and chair back. Shelf is not smooth.
Computer	Textureless wall is more accurately obtained. Marker is fattened Disparity is	Similar to act architecture but speaker is smoothly obtained without any	Error on the wall is more. Speaker is smoothly obtained and marker is not fattened.	Error on the cpu and on the red bag behind the cpu

	missing on speaker	missing pixels. Error on the red bag behind the cpu		
Crusade	Wall and table is comparatively better	Similar to full architecture	A portion of table is missing. Error in disparity on wall and the front chair	Error on the edge of the table and the back chair.

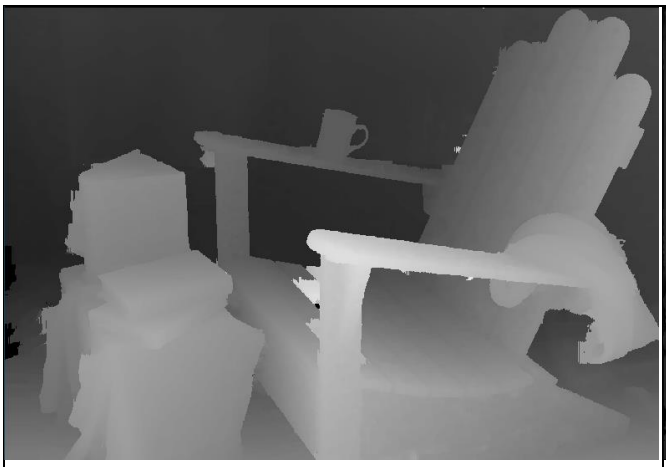
Some Observations: Thin structures are nicely obtained in cnn-crf whereas they are fattened/missing in mc-cnn. (motorcycle back bench, Adirondack cup, playroom,)
cnn-crf is failing more on textureless regions like on walls (Crusade: a portion of table and wall, Computer, Livingroom:shadow on wall, computer)

Adirondack
Cup is nicely obtained in cnn-crf. The edge of the arms of the chair is smooth on mccnn-acrt and cnn-crf

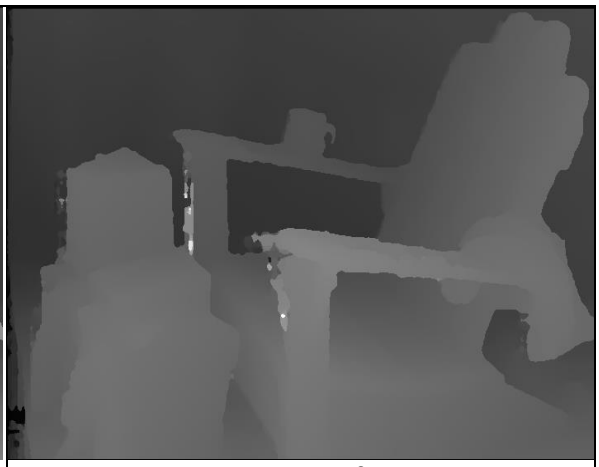
Input(Left



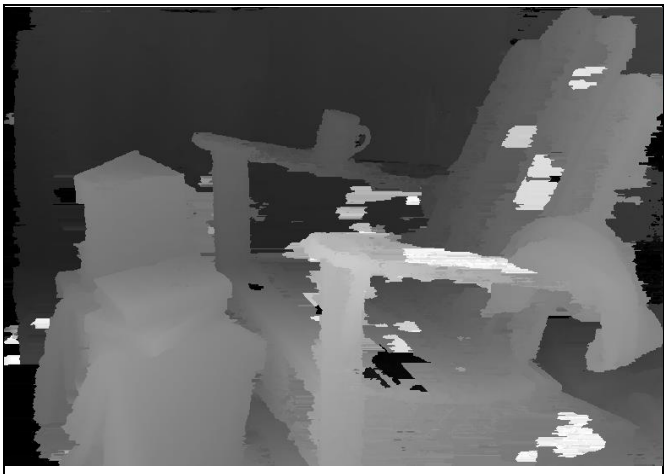
MC-CNN acrt



CNN-CRF



MC-CNN fst

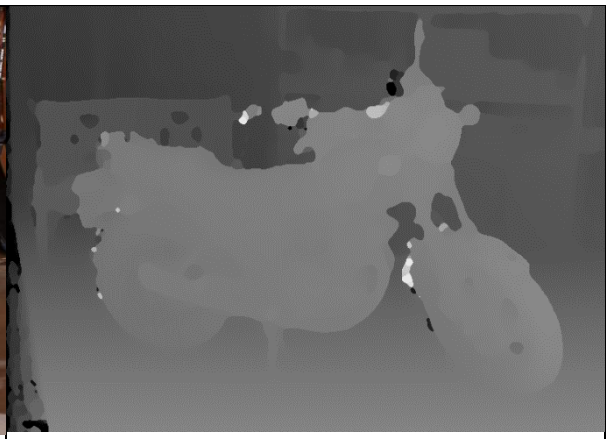


SGBM

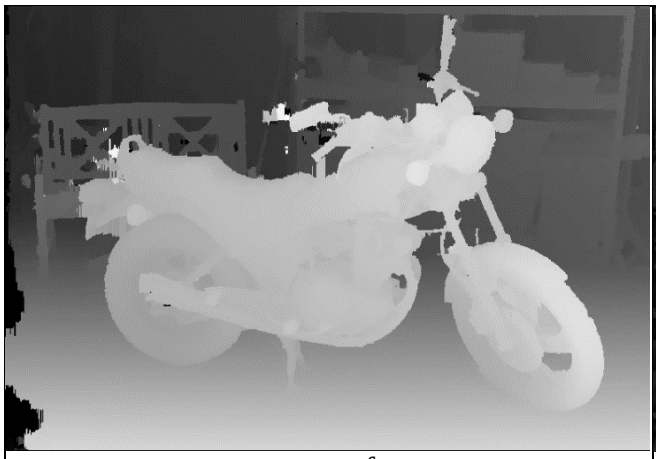
Motorcycle

The structures on the bench at the back is fattened in mc-cnn. The spokes of the wheels are not distinct in mc-cnn, where as in cnn-crf it is more accurate.

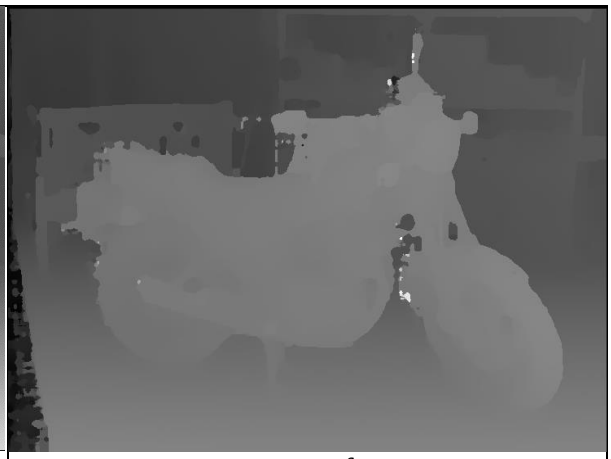
Input



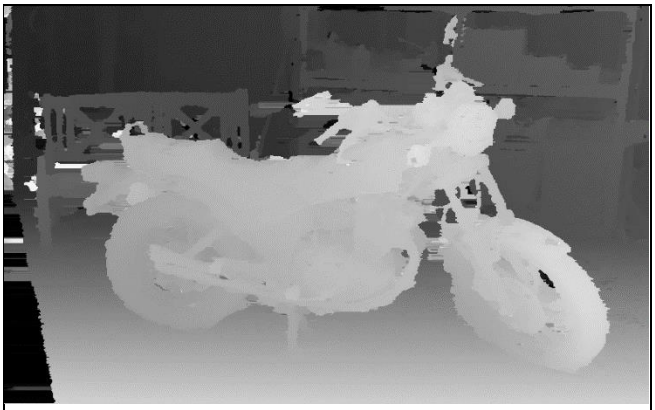
mc-cnn_act



cnn-crf

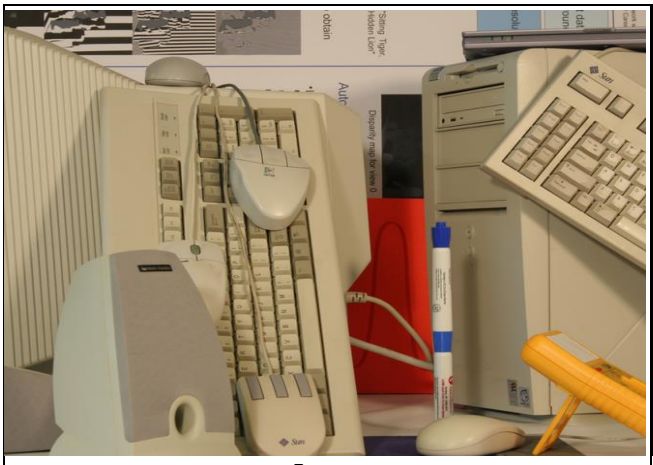


mc-cnn fst

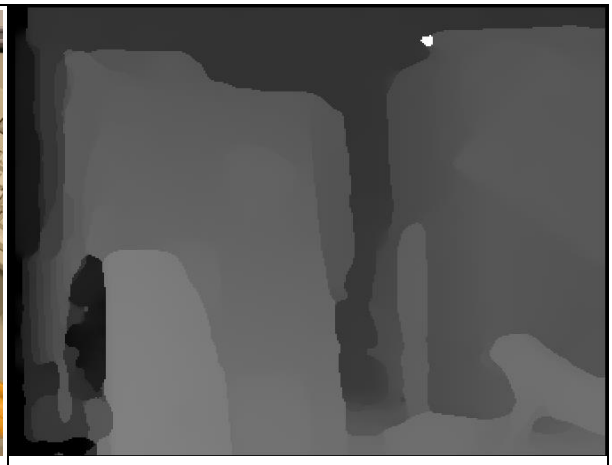


SGBM

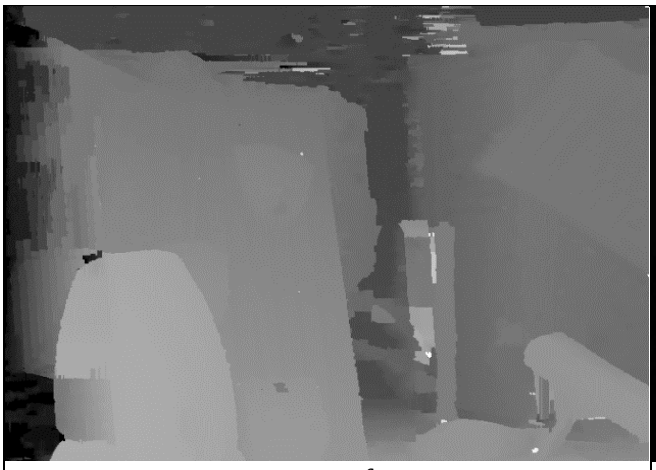
Computer:
Error in disparity on the wall by cnn-crf. Marker is fattened in mc-cnn. Disparity is missing on the speaker in mc-cnn acrt but it is smoothly obtained in mc-cnn fst.



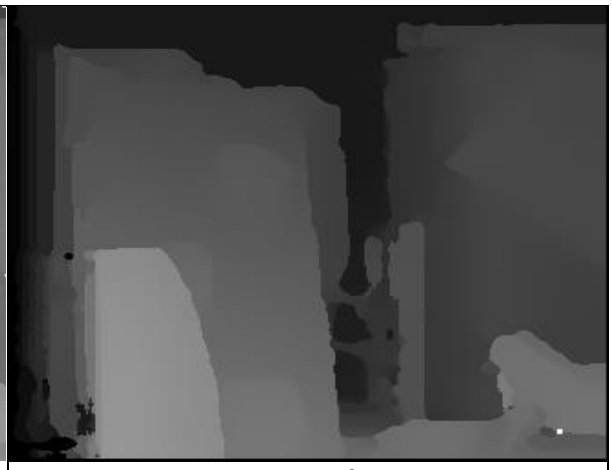
Input



mc-cnn act



cnn-crf



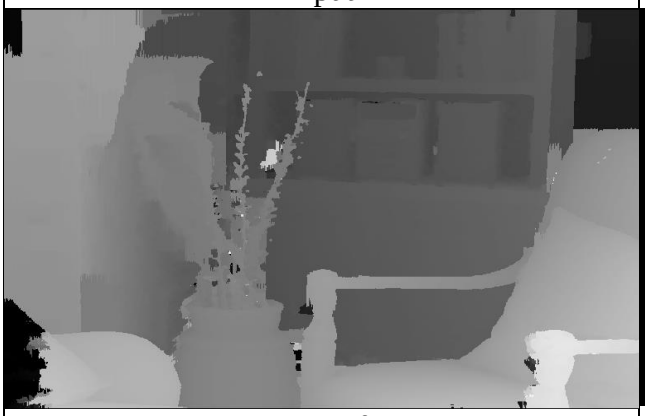
mc-cnn fst

Livingroom

Cnn-crf is comparatively failing more on the wall where shadow of the plant is present.

Input

mc-cnn acrt

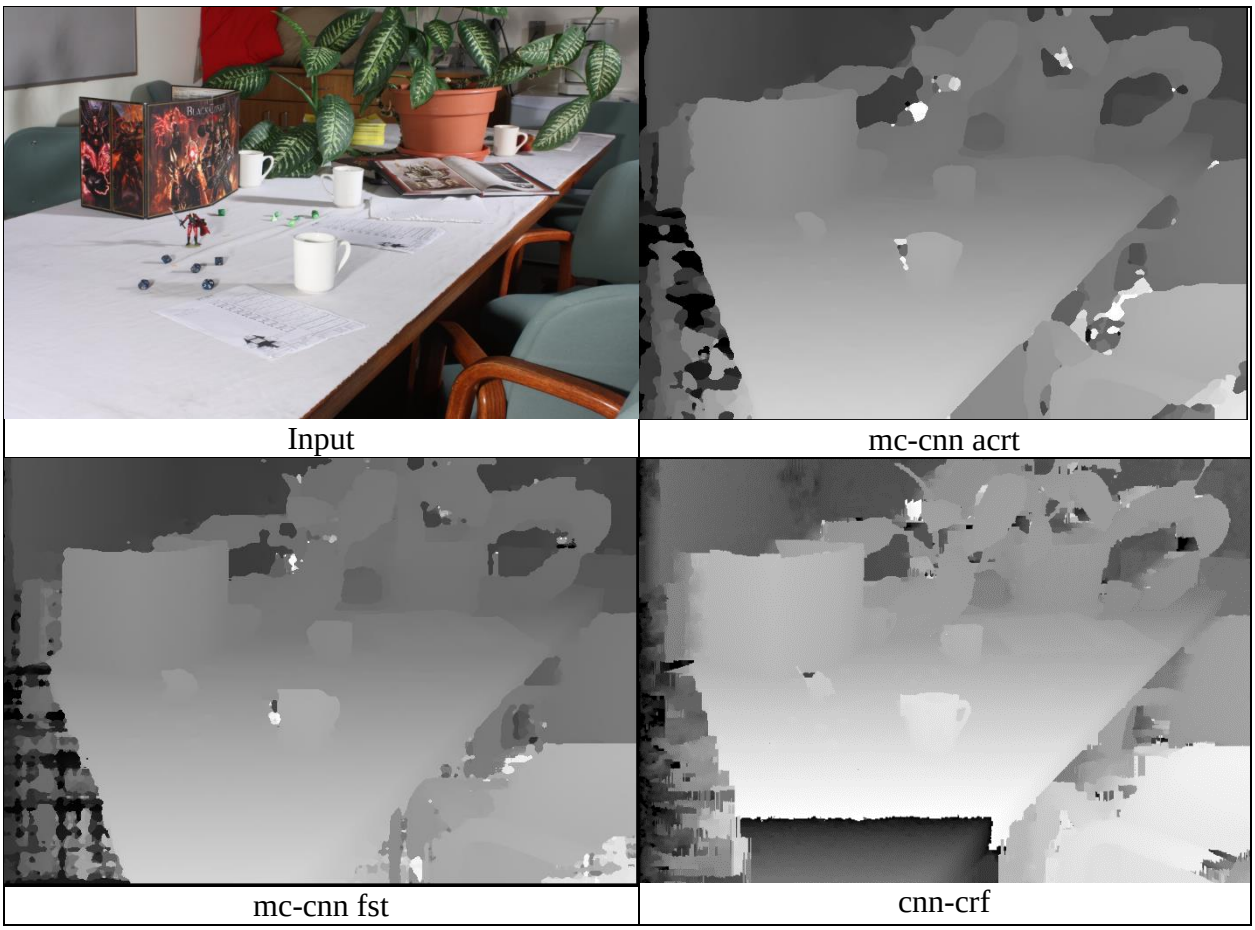


cnn-crf

mc-cnn fst

Crusade

Error in cnn-crf on the wall and a chunk of table is missing(white textureless region).Table is smoothly obtained by mc-cnn.



Input	mc-cnn acrt
mc-cnn fst	cnn-crf

Kitti2015: Some observations: In cnn-crf, object boundaries are comparatively better like speed-limit sign board, edges and textures of tress are more distinct.
Mc-cnn is smoother on the road and the sky. Object boundaries are fattened/not sharp

Kitti-2015(img12)

Input

mc-cnn fst

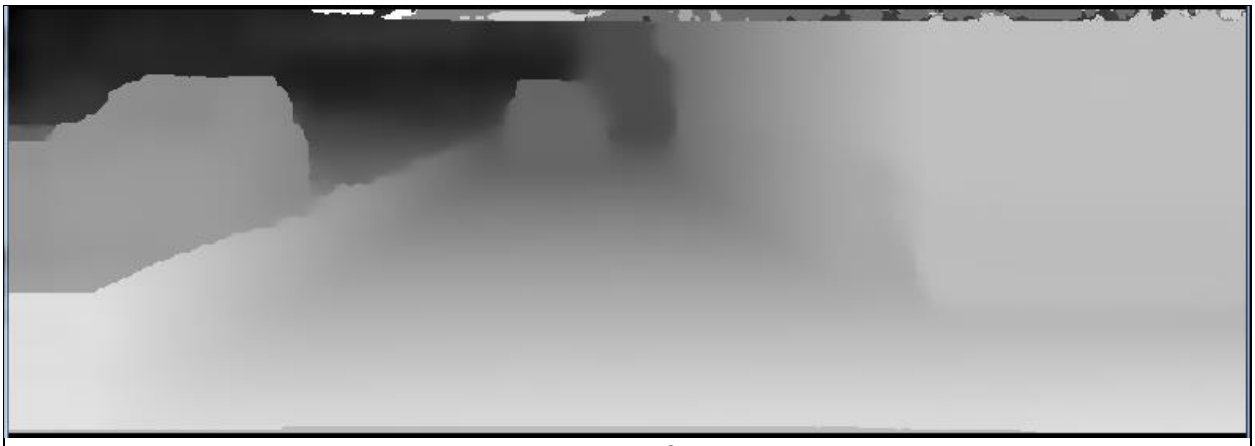
cnn-crf

mc-cnn acrt

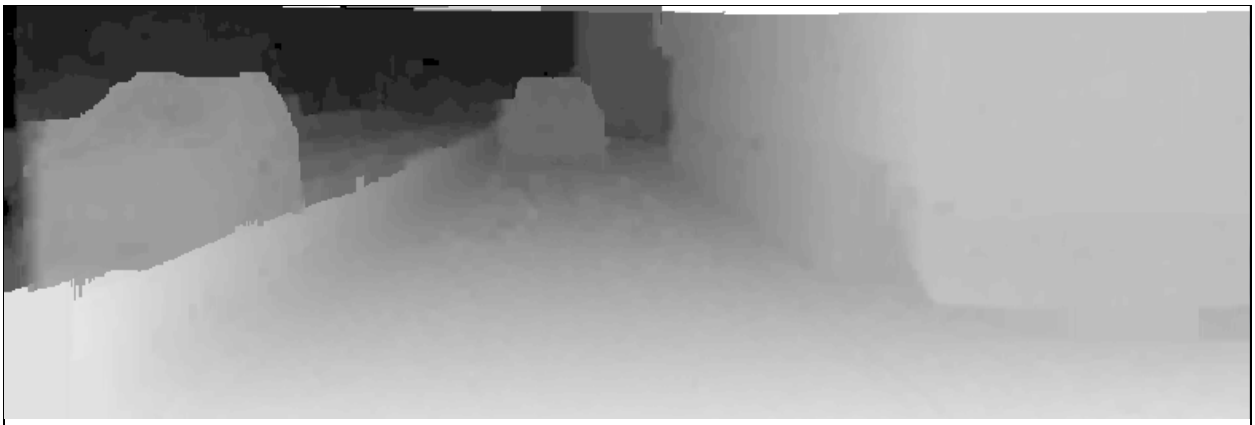
Good weather dataset (img100**)**

Road is smoothly obtained in mc-cnn. Tire boundaries are more distinct in cnn-crf

Input



mc-cnn fst



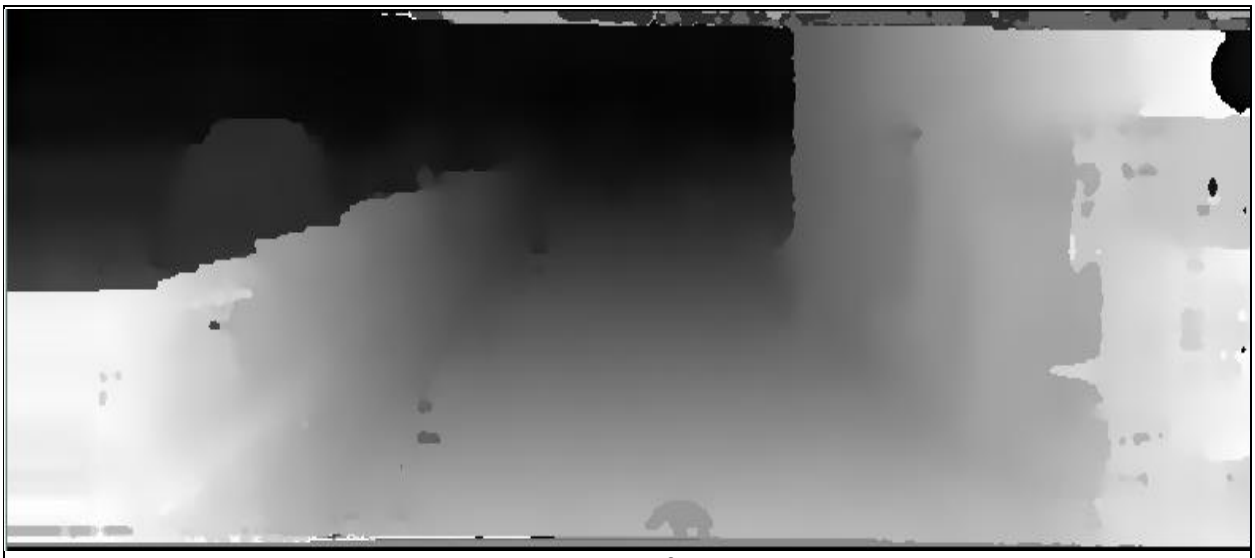
cnn-crf

mc-cnn _acrt

Bad weather (Img14)

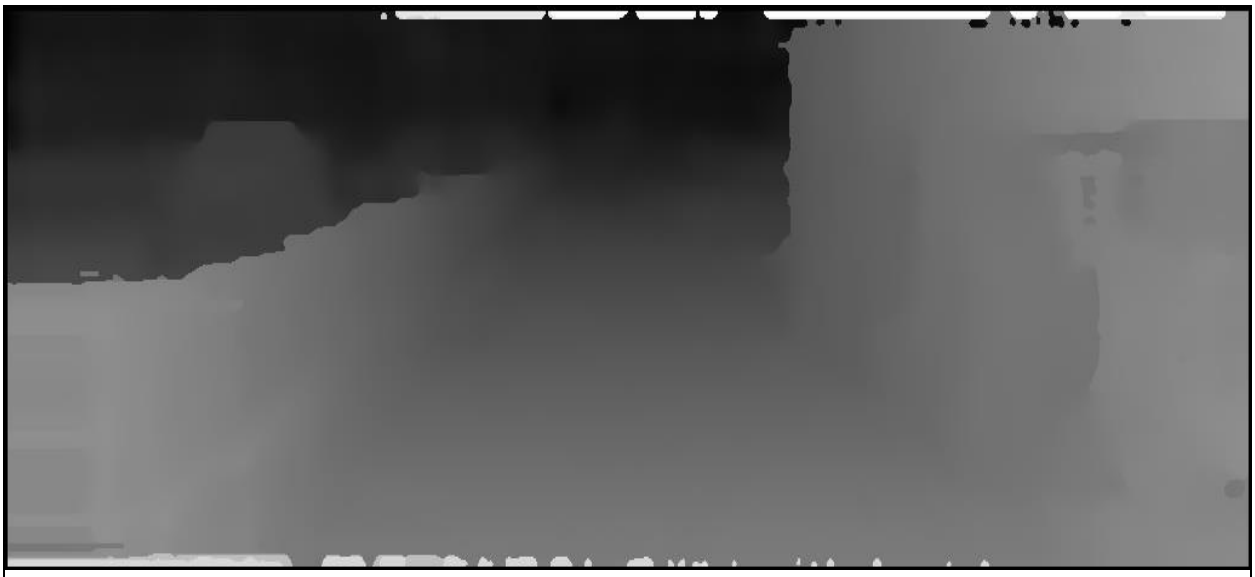
Road is smoothly obtained in mc-cnn.

Input



mc-cnn fst

cnn-crf



mc-cnn acrt

9. ACTIVE DEPTH

Two major types of active 3D sensors exist: **Time-of- Flight** (ToF) and **Structured Light** (SL)
ToF sensors emit infrared-light (IR) and capture its reflection. The distance assigned to a pixel is inferred from the time delay between emission and reception of the IR-signal.
ToF sensors run in real time and provide good results even on textureless surfaces. Yet, the sensors suffer from limited resolution as well as various error sources such as noise, multi path, "flying pixels" and are susceptible to background illumination.
Due to size and power limitations it is difficult to deploy ToF on smartphones.
SL sensors work by projecting a light pattern onto the scene and then capturing it with a camera. The distortion of the pattern is used to infer the depth.

Active depth sensing has a limited range of few metres. It fails on glossy surfaces, edges, fine structures and elements that are illuminated by bright light owing to the infrared-based depth perception. This also severely degrades the usability of active sensors outdoors.
Contrary to active sensing, Stereo methods benefit from bright illumination and sharp object boundaries. However, local stereo matching requires well-textured surfaces, something active-based 3D scanning does not depend on. Also, global stereo reconstruction algorithms are often slow.
Due to this complimentary nature of depth from stereo and active sensing, active depth estimation can be combined with stereo methods to obtain a more accurate and dense depth map.
The information can be combined by relying on passive stereo in highly textured regions while using data from active depth sensors in featureless region. A confidence map based on active sensor is created. Another confidence map is created based on stereo algorithm. Both the

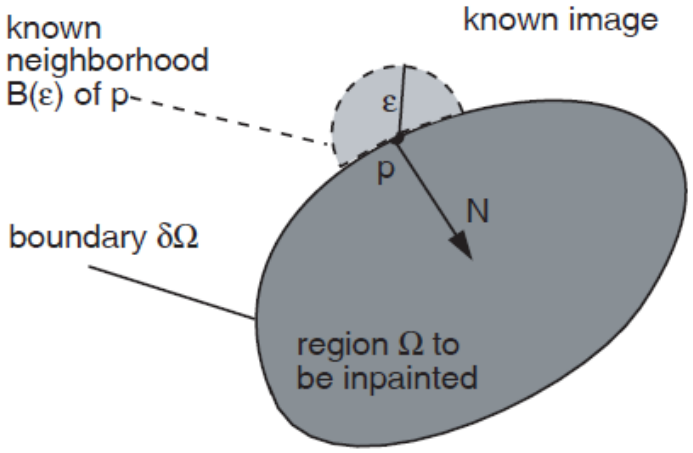
confidence maps are then incorporated into the cost function to obtain a more accurate and dense depth map.
The challenge faced is that data from the active sensors are of a much lower resolution than those from typical RGB cameras used in the passive subsystem. Thus there is a need for up-sampling at real time.
Disparity estimation can be initialized using low resolution active depth prior. For, pixels that exhibit prior knowledge from active depth, a reduction of the disparity search range to the uncertainty interval of the prior allows for a significant reduction of ambiguities and will in turn reduce the run time of algorithm.

KINECT RGB-D

Dataset used: Kinect(v1), http://rgbd-dataset.cs.washington.edu/dataset/rgbd-scenes/

Depth maps captured by Kinect-like cameras are lack of depth data in some areas and suffer from heavy noise. These defects have negative impacts on practical applications. In order to enhance the depth maps, we can use inpainting algorithms to fill in the holes. We will look into an inpainting algorithm based on Fast Marching method. Then we will look into an extended version of the original fast marching method (FMM) to reconstruct unknown regions.
The extended FMM incorporates an aligned color image as the guidance for inpainting. An edge-preserving guided filter is further applied for noise reduction.

Fast Marching Method



Point ‘p’ is located on the boundary δΩ of the region Ω to be inpainted.
The inpainting model is given by the following equation.

$$D(p) = \frac{\sum_{q \in \mathcal{B}(p)} \omega(p,q)[D(q) + \nabla D(q)(p-q)]}{\sum_{q \in \mathcal{B}(p)} \omega(p,q)}$$

where *w*(p, q) is the weighing function designed as the product of the following three terms: w_{dst}(p,q),w_{dir}(p,q), w_{lev}(p,q), ‘q’ belongs to neighborhood of p, D(q) is depth at q and D(p) is the depth that is to be filled at p.
The above equation shows how to inpaint a point on the unknown region’s boundary δΩ as a function of known image pixels only.
To inpaint the whole Ω, we iteratively apply the above equation to all the discrete pixels of δΩ, in increasing distance from δΩ’s initial position δΩi, and advance the boundary inside Ω until the whole region has been inpainted as shown in the pseudo code below.

δΩi = *boundary of region to inpaint*
δΩ = *δΩi*
while *(δΩ not empty)*
{
P = *pixel of δΩ closest to δΩi*
Inpaint p using equation
Advance δΩ into Ω
}
Inpainting points in increasing distance order from *δ*Ωi ensures that areas closest to known image points are filled in first, thus mimicking manual inpainting techniques.
Implementing the above requires a method that propagates *δ*Ω into Ω by advancing the pixels of *δ*Ω in order of their distance to the initial boundary *δ*Ωi. For this, we use the **fast marching method**.
In brief, the FMM is an algorithm that solves the Eikonal equation:
|∇T| = 1 on Ω, with T = 0 on *δ*Ω.
The solution T of the above equation is the distance map of the Ω pixels to the boundary *δ*Ω. The level sets, or isolines, of T are exactly the successive boundaries *δ*Ω of the shrinking Ω that we need for inpainting. The FMM guarantees that pixels of *δ*Ω are always processed in increasing order of their distance to boundary T, i.e., that we always inpaint the closest pixels to the known image area first.

Guided Inpainting and Filtering For Depth Maps

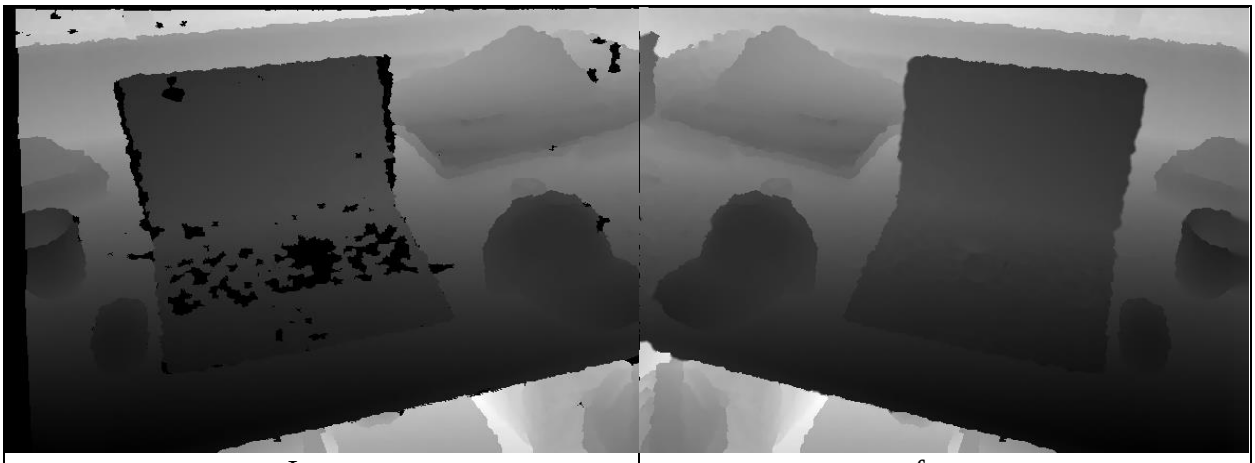
This method extends the fast marching method by incorporating an aligned color image as a guidance for inpainting. Then an edge preserving guided filter is applied for noise reduction.
The inpainting algorithm is similar to the above method. A new term $w_{col}(p,q)$ is introduced.
$w_{col}(p,q)$ makes pixels having similar color to D(p) contribute more than others. This term is designed based on an assumption that neighboring pixels similar in color are likely to have similar depth values.
In the above algorithm, the order of inpainting is only determined by the distance to the boundary, which may break edges. In order to use an aligned color image as the guidance, a color-similarity term around the pixel is calculated by:

$$C(p) = \frac{1}{|\mathcal{N}(p)|} \sum_{q \in \mathcal{N}(p)} \exp(-\frac{||I(q) - I(p)||^2}{2\sigma_c^2})$$

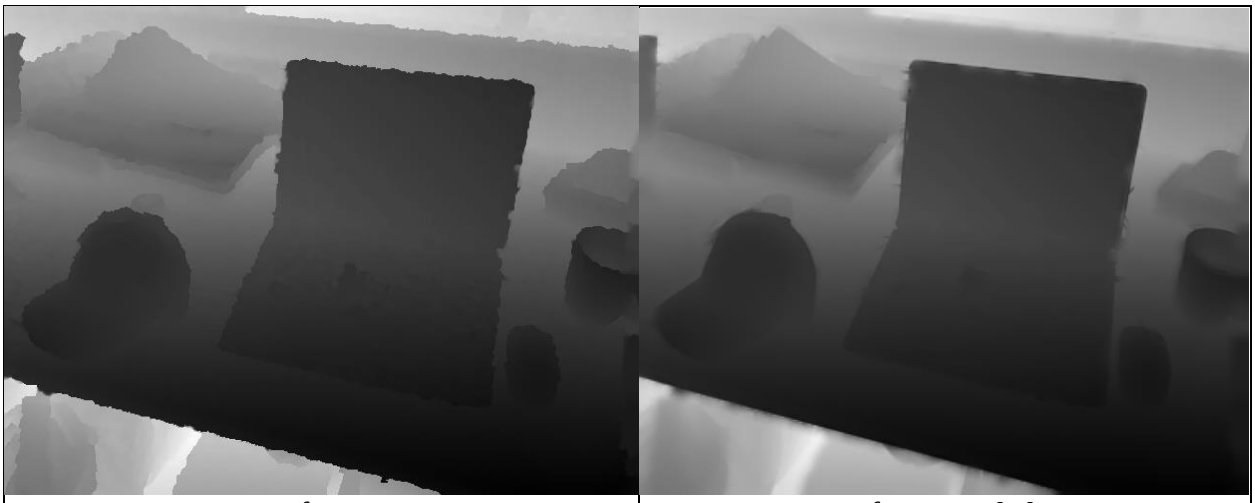
where N(p) is the neighboring window of p, including the pixels with both known and unknown depth values,and |N(p)| is the number of pixels. The color-similarity term assigns a high priority to a pixel having similar color around it. In other words, this term makes the pixels near edges estimated later, thus trying to achieve fine edges. With this modification term, we decide the proceeding order by minimizing E.

$$E = (1 - \lambda)T + \lambda(1 - C)$$

Guided Filtering: The guided inpainting process makes every pixel in unknown regions get a proper depth value, however, the inpainted depth map still suffers from noise and unstable boundaries. A guided filter is used with the aligned color image as guidance



Input	fmm



gfmm	gfmm+guided

Installation instructions:

1. **CNN-CRF stereo method**

- Github link: https://github.com/VLOGroup/cnn-crf-stereo
- For downloading the sourcecode, if connected through SSH (eg. Through mobaxerm), set environment variable GIT_SSL_NO_VERIFY as true
  export GIT_SSL_NO_VERIFY = true
  git clone https://github.com/VLOGroup/cnn-crf-stereo.git --recursive

- Dependencies: gcc >4.9 (preferably try any 4.9 version) : To locally install gcc without sudo permission refer:
- http://luiarthur.github.io/gccinstall
- http://bakeronit.com/2015/11/04/install_gcc/
- Note: ./contrib/download_prerequisites will download the support libraries and create symlinks. If wget is not able to connect to the server for downloading the support libraries, manually download the support libraries and then create symlink.
- After installation, set GCC path.
  Example: export PATH="/home/veerav/gcc-4.9/bin:$PATH"
  Create environment variable CC and CXX

- Cmake >3.2

- Create environment variable CC and CXX in order for cmake to find the updated version of gcc/g++
    - CC =/path-to-gcc-installation-directory/bin/gcc
    - CXX= /path-to-gcc-installation-directory/bin/g++
    - OpenCV path should be set.:
    - ImageUtilities: while installing it, use:
    - cmake –DCUDA_USE_STATIC_CUDA_RUNTIME=OFF if you get an error with ndarray.
    - Note: clear cmakecache after build errors, before retrying to build again
    - cnpy: Use cmake –DCMAKE_INSTALL_PREFIX=/ your-installation-directory-path/ to install in a specific directory. (if you do not have sudo permission to install in the default path)
    - Stereo-Net: set cnpy include directory and lib directory path in cmakelists.txt present inside stereo folder:(line 70)
    - SET(CNPY_INCLUDE_DIR "/data/veerav/soft/include" CACHE FILEPATH "Path to cnpy include")(line 80)
    - SET(CNPY_LIB_DIR "/data/veerav/soft/lib" CACHE FILEPATH "Path to cnpy lib")

2. **mc -cnn**

- github link: https://github.com/jzbontar/mc-cnn
- For torch installation refer: http://torch.ch/docs/getting-started.html
- Use older sourcecode version of torch (before july 2017) to avoid error in mc-cnn code
- Error: Trying to resize storage that is not resizable or follow instructions as mentioned by some users in issues thread.
- CUDA path should be set
- export PATH="/usr/local/cuda-8.0/bin:$PATH"
- export LD_ LIBRARY_PATH="/usr/local/cuda-8.0/targets/x86_64 linux/lib:$LD_LIBRARY_PATH"
- set environment variable CUDA_VISIBLE_DEVICES to target a specific gpu
  Example: export CUDA_VISIBLE_DEVICES=0

- Note: Update the environment variable in .~profile in the home directory
  To set a PATH variable: export PATH="/path –to-directory:$PATH"
  To set Library Path:
  export LD_LIBRARY_PATH="/path-to-lib-directory:$LD_LIBRARY_PATH"
- While installing something if you run out of memory, create a tmp folder in /data/user
  Export TMPDIR=/path-to-temp-directory

REFERENCES

1. **“**Scharstein, Daniel, and Richard Szeliski. "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms." *International journal of computer vision* 47.1-3 (2002).

2. “Multiview geometry in computer vision” Richard Hartley and Andrew Zisserman. Cambridge University Press

3. Mei, Xing, et al. "On building an accurate stereo matching system on graphics hardware." *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. IEEE, 2011.

4. “The OpenCV Reference Manual”, Itseez, opencv.org.

5. “Stereo Vision: Algorithms and Applications” Stefano Mattoccia, University of Bologna.

6. Zbontar, Jure, and Yann LeCun. "Stereo matching by training a convolutional neural network to compare image patches." *Journal of Machine Learning Research* 17.1-32 (2016)

7. Knöbelreiter, Patrick, et al. "End-to-End Training of Hybrid CNN-CRF Models for Stereo." *arXiv preprint arXiv:1611.10229* (2016).

8. Sethian, James A. "Fast marching methods." *SIAM review* 41.2 (1999): 199-235.

9. Gong, Xiaojin, et al. "Guided depth enhancement via a fast marching method." *Image and Vision Computing* 31.10 (2013): 695-703