Master's Thesis Draft

by

Victor Gan

BSc, University of Waterloo, 2013

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF SCIENCE
(Computer Science)

The University of British Columbia
(Vancouver)

April 2015

Abstract

This document provides brief instructions for using the `ubcdiss` class to write a **UBC!**-conformant dissertation in LaTeX. This document is itself written using the `ubcdiss` class and is intended to serve as an example of writing a dissertation in LaTeX. This document has embedded **URL!**s (**URL!**S) and is intended to be viewed using a computer-based **PDF!** (**PDF!**) reader.

Note: Abstracts should generally try to avoid using acronyms.

Note: at **UBC!** (**UBC!**), both the **GPS!** (**GPS!**) Ph.D. defence programme and the Library's online submission system restricts abstracts to 350 words.

Revision: June 20, 2015

Preface

At **UBC!**, a preface may be required. Be sure to check the **GPS!** guidelines as they may have specific content to be included.

d

Table of Contents

List of Tables

List of Figures

Glossary

This glossary uses the handy `acroynym` package to automatically maintain the glossary. It uses the package's `printonlyused` option to include only those acronyms explicitly referenced in the LaTeX source.

GPS Graduate and Postdoctoral Studies

PDF Portable Document Format

URL Unique Resource Locator, used to describe a means for obtaining some resource on the world wide web

Acknowledgments

Thank those people who helped you.

Don’t forget your parents or loved ones.

You may wish to acknowledge your funding sources.

NSERC CGS-M scholarship.

Julieta Martinez for the discussions.

Chapter 1

Introduction

(Contributions) In this paper we put this point on a further basis:

- contribution 1

Chapter 2

The Problem

2.1 Problem Goals

- Advances scientific understanding of the world
- Be general enough to be widely useful
- Novel: don't work on things everyone else is.

2.2 Specific Problem

2.2.1 Tracking

1. Is RGB+D Tracking significantly better than RGB tracking?
2. Is just D tracking better than RGB tracking?
3. Unsupervised tracklets, which also beats state-of-art tracking. T-by-D: input patch, output detection. Struck: input patch, output displacement. New: input video, output tracks, and formulate objective function to minimize error on this directly (through structured SVM?).
4. Combine state-of-art pedestrian detection with state-of-art tracking (Combine Kernelized Correlation Filters with Ptior's Integral Images)

5. Category recognition (HOG) vs Instance recognition (SIFT). We care about instance recognition in tracking, because the same object appears throughout time. There's no reason to use HOG. (idea from [**?**])

6. Integral Videos for fast feature computation, combined with edge boxes

7. Henriques' KCFs with 3d data.

2.2.2 Object Proposals

1. Problem: Given 2 class labelled data. Find a basis transformation that maximally separates the two clusters, regardless of background

2. New OBject Propoal technique: We want to find local features of objects that are maximally discriminative from the background and other surrounding/-moving objects. SIFT tries to do this by making 'invariant' features. Perhaps we can learn (eg. OMP/sparsity?) the best space to cluster background pixels to target pixels. A lean neural network/autoencoder?

2.2.3 Misc

List of specific problems I might tackle:

1. motion blur. HOG doesn't take into account motion blur. Color does? The background probably has less motion. Track the background instead?

2. Review of NON-online and NON-model-free tracking.

3. Review natural images. Paper on generating synthetic natural images, trained on Flickr10M.

4. Use prime numbers as a multiplicative basis for dictionary learning?

2.3 Questions to Answer

1. Where does deep learning fit in?

2. How should I treat the initial labelled bounding box differently?

People appeal to deep learning as if it learns generalizable features of natural image statistics.

2.4 Constraints

1. Start and stop of tracks
   (a) Occlusions occur when object goes in front of tracked object
   (b) tracks can start/stop at edges of frame and when appearing behind a foreground object
2. objects move at a certain velocity,
3. objects can change scale and rotation
4. objects are rigid
5. objects are at the same distance from the camera. Objects aren't necessarily the same colour/texture, but they move in the same tracks
6. video is not like static images: motion blur is inherent in fast moving objects, making it look different than normal
7. realtime/anytime (improves, but can be stopped at any time)

2.4.1 Physical Constraints

1. moving object is connected
2. moving object moves smoothly (minimizes jerk?)
3. moving object conserves energy/momentum?
4. moving object cannot just disappear: it must be occluded

1. how do you make an view-independent feature set for moving objects?
2. optimization: define a quickly converging optimization function

2.5 Potentially Useful Ideas

1. markov chain
   (a) viterbi algorithm
   (b) forward-backward algorithm
2. bootstrapping
3. decision trees
4. entropy/infogain
5. "compute the distance of high-dimensional appearance descriptor vectors between image windows" "We derive an upper bound on appearance distance given the spatial overlap of two windows in an image, and use it to bound the distances of many pairs between two images" [**?**]
6. Iterative closest point (ICP): align 2 point clouds
7. Sparse Coding (Object Matching Pursuit (OMP))
8. Edge boxes [**?**]
9. Pearson Correlation Coefficient: http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient

Efficient operations

- XOR hamming distances
- integral images
- circulant structure (henriques)
- quad trees $O(n^2)$ distances between each n point
- lookup tables/caching

high level ideas

1. recursion

2. optimization: define a quickly converging optimization function

1. k-d tree ”kd-trees are not much better than brute-force search when the number of descriptor dimensions exceeds 20”

2. Chow-Liu trees: fully connected Bayesian network to a tree

3. Branch and bound (explore a space)

4. Rapidly exploring random trees

Math Circulant/Toeplitz matricies (Gray, [**?**])

> A great deal is known about the behavior of Toeplitz matrices the most common and complete references being Grenander and Szego [16] and Widom [33]. A more recent text devoted to the subject is Bottcher and Silbermann [5]. Unfortunately, however, the necessary level of mathematical sophistication for understanding reference [16] is frequently beyond that of one species of applied mathematician for whom the theory can be quite useful but is relatively little understood.

From wikipedia's circulant matricies: “circulant matrices are important because they are diagonalized by a discrete Fourier transform, and hence linear equations that contain them may be quickly solved using a fast Fourier transform.”

Robert Gray's definitive book on entropy and information theory [**?**]

McKay's [**?**]

2.6 Summaries of problems from other papers

“Consider a video stream taken by a hand-held camera depicting various objects moving in and out of the cameras field of view. Given a bounding box defining the object of interest in a single frame, our goal is to automatically determine the objects bounding box or indicate that the object is not visible in every frame that follows. The video stream is to be processed at frame-rate and the process should run indefinitely long. We refer to this task as long-term tracking.” [**?**] (in fact, their entire introduction is well-written)

2.7 Motivation

- Model-free online tracking is more organized than RGBD tracking
- Still no comprehensive analysis on what makes it work (some people mention work
- Some problems are solved (pose recognition with RGBD - shotton et al. [**?**]. I want to solve tracking with RGBD

2.8 Types of Papers in Robotics/Computer Vision

- Fixed benchmark, a paper that improves on the benchmark incrementally (little citations)
- Blow a benchmark out of the water. Fixed benchmark, a paper that improves on the benchmark massively (groundbreaking)
- Blow multiple benchmarks out of the water
- Introduces a new benchmark, takes off if an Ivy League/In secret paper passing list
- Introduces a new problem (usually not cited if problem is uninteresting)
- Introduces a novel method of academic elegance (something other than MCMC/supervised learning/GPs/etc.)
- Survey papers, meta-reviews, etc.
- Solves a field, eg. Shotton *et al* [**?**].

Chapter 3

Literature

3.1 Online RGB Trackers (timeline/history)

Tracking-by-detection

Hare *et al* [**?**]:

“An approach to tracking which has become particularly popular recently is tracking-by-detection [**?**], which treats the tracking problem as a detection task applied over time”

Hare *et al* [**?**]:

> State-of-the-art adaptive tracking-by-detection methods mainly focus on improving tracking performance by increasing the robustness of the classifier to poorly labelled samples resulting from this approach. Examples of this include using robust loss functions [6], [7], semi-supervised learning [8], [9], or multiple-instance learning [3], [10].

”The core component of most modern trackers is a discriminative classifier, tasked with distinguishing between the target and the surrounding environment.” [**?**].

”ARGUABLY one of the biggest breakthroughs in recent visual tracking research was the widespread adoption of discriminative learning methods” [**?**]

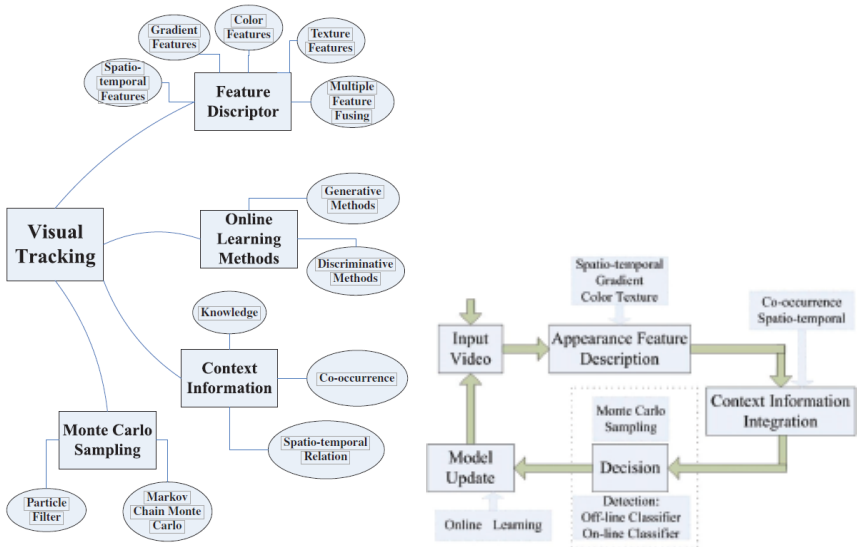

Figure 3.1: Framework of tracking models. From [**?**].

3.2 Online RGB Trackers (vision papers)

Survey papers:

1. Yang et al. [**?**]. 2011 survey of the field

2. **Pang et al's Survey** [**?**] notes biases in comparisons: usually new papers list their method as the best (because of a specific methodology); however second best paper rankings are fairly robust. A meta-analysis concludes the following methods are competitive: Struck, MIL, TLD, VTD.

3. **an extensive PAMI Survey** [**?**] claims Struck is the best, and analyses specific failure cases and how it affects specific method

4. **Appearance Model Survey** [**?**]

5. A comprehensive list of papers can be found by searching papers that have referenced Struck; which is the most frequently used benchmark to compare against.

6. **Visual Tracking Benchmark**[**?**]

7. Wu et. al [**?**] (SCM, Struck, TLD, ASLA, CXT, VTD, VTS, CSK) concludes

   (a) **Background Information** background information is critical for effective tracking. It can be ex- ploited by using advanced learning techniques to encode the background information in the discriminative model im- plicitly (e.g., Struck), or serving as the tracking context explicitly (e.g., CXT)

   (b) **local models** are important for tracking as shown in the performance improvement of local sparse representation (e.g., ASLA and SCM) com- pared with the holistic sparse representation (e.g., MTT and L1APG). They are particularly useful when the appearance of target is partially changed, such as partial occlusion or deformation.

   (c) **local models** motion model or dynamic model is cru- cial for object tracking, especially when the motion of target is large or abrupt. However, most of our evaluated tracker- s do not focus on this component. Good location predic- tion based on the dynamic model could reduce the search range and thus improve the tracking efficiency and robustness.

8. VOT2014 Results [**?**]

"The challenge considers single-camera, single-target, model-free, causal trackers, applied to short-term tracking. The model-free property means that the only supervised training example is provided by the bounding box in the first frame. The short-term tracking means that the tracker does not perform re-detection after the target is lost. Drifting off the target is considered a failure. The causality means that the tracker does not use any future frames, or frames prior to re-initialization, to infer the object position in the current frame." [**?**]

"In this paper, we focus on the problem of model-free online tracking of an object, given only the objects initial position and previous observations, within a tracking-bydetection framework."[**?**]

"Model drift occurs because factors like tracking failure, occlusions and misalignment of training samples can lead to bad model updates. One remedy is

to incorporate the first frame template or prior knowledge in the online model update procedure [20,15]. However, relying on a fixed model prior tends to restrict the trackers ability to handle large object appearance changes. Other trackers [22,32,14] use a censorship mechanism where an update is prevented when certain criteria are met (or not met). The detection of good or bad updates usually relies upon smoothness assumptions for motion and appearance changes, which are often violated in challenging scenarios. And once the censorship mechanism fails, these trackers will either miss the chance to evolve or get trapped in a background region, due to the fact that the model can only evolve forward, without a mechanism to correct for past mistakes."[**?**]

3.2.1 Pre-CVPR 2013 Trackers

Online RGB Trackers require no prior knowledge of the object, and only a bounding box of the target on the original frame. A survey of tracking methods [**?**] (SCM, Struck, TLD, ASLA, CXT, VTD, VTS, CSK), as well as papers following the survey [**?**], show the following methods are competitive for this problem:

1. **Struck** [**?**] uses a kernelized structured output SVM to directly learn displacement vectors. Gaussian kernel on 192 haar-like features. Struck's TPAMI paper [**?**].

   Kernelization does most of the work, but structured SVM formulation avoids arbitrary hand-tuned parameters: "a large part of the performance gains... can be attributed to our use of a kernelised SVM rather than a boosting-based classifier."

   "(previous) algorithms separate the adaptation phase of the tracker into two distinct parts: (i) the generation and labelling of samples; and (ii) the updating of the classifier."

   "we make use of the structured output SVM framework of Tsochantaridis et al. [15]. In particular, we extend the online structured output SVM learning method proposed by Bordes et al. [16], [17] and adapt it to the task of adaptive object tracking."

2. **SCM** [**?**]

3. **TLD** [**?**] "We develop a novel learning method (P-N learning) which estimates the errors by a pair of experts: (i) P-expert estimates missed detections, and (ii) N-expert estimates false alarms. The learning process is modeled as a discrete dynamical system and the conditions under which the learning guarantees improvement are found. We describe our real-time implementation of the TLD framework and the P-N learning.

4. **APG-L1** [**?**] "l1 norm related minimization model".

5. **MIL** [**?**] Older well-known tracker using multiple instance learning.

6. **CXT** [**?**] uses background context

7. **ASLA** [**?**]

8. **Circulant** [**?**] (TPAMI [**?**]) uses fourier transformed graham matrix to improve .

   The fastest tracker in [**?**].

   "a notable optimization is to use a fast but inaccurate classifier to select promising patches, and only apply the full, slower classifier on those [18], [19]."

9. Zhang [**?**] "propose a projection to a fixed random basis, to train a Naive Bayes classifier, inspired by compressive sensing techniques."

3.2.2 Post-CVPR 2013 Trackers

After Wu et. al's survey [**?**], the following notable methods were also published:

1. **Self-paced learning** [**?**] "we show that an accurate appearance model is considerably more effective than a strong motion model".

2. **MEEM** [**?**] claims state-of-the-art over Struck, SCM, MIL.

3. **Xiang** [**?**]

4. **Occlusion and motion reasoning for long-term tracking** [**?**] "Struck fails in the presence of long-term occlusions as well as severe viewpoint changes of the object. In this paper we propose a principled way to combine occlusion and motion reasoning with a tracking-by-detection approach."

5. Color tracker [**?**] - the **best**.

3.2.3 Analysis

Struck is a major paper, topping the benchmarks of Pang [**?**], Wu [**?**] and Smeulders [**?**]. Notably, newer papers aren't as well-tested and perhaps better. Struck is basically kernelized structured output SVMs, where (x, y) are image patches and displacements. Most of the gains come from the kernelization, though.

Circulant trackers are really, quite good.

3.3 Online RGB-D Trackers (vision papers)

Online RGB-D Trackers no prior knowledge of the object. Song et. al's survey [**?**], show **incorporating depth into tracking beats the state of the art.** It shows **Struck** [**?**] and VTD are competitive.

Gaussian Process Regression [**?**] beats struck and Song et. al's survey benchmarks. Compares against VOT2013 [**?**], Song's princeton RGBD benchmark [**?**] and Wu's CVPR RGB [**?**].

3.4 Trackers (robotics papers)

1. RSS 14: Anytime Tracking [**?**]

2. RSS 14: DART Pose Estimation [**?**] (relevant?)

3. ICRA: Small Obstacle Discovery over Images [**?**]: small object segmentation using RGB, since depth info doesn't give much.

4. ICRA 14: Tracking ping pong balls [**?**] This paper proposes a way to observe and estimate ball's spin in real-time, and achieve an accurate prediction. Based on the fact that a spinning ball's motion can be separated into

global movement and spinning respect to its center, we construct an integrated vision system to observe the two motions separately. With a pan-tilt vision system, the spinning motion is observed through recognizing the position of the brand on the ball and restoring the 3D pose of the ball. Then the spin state is estimated with the method of plane fitting on current and historical observations. With both position and spin information, accurate state estimation and trajectory prediction are realized via Extended Kalman Filter(EKF).

5. ICRA 14: road scene segmentation [**?**] "we first produce initial object hypotheses by clustering the sparse 3D point cloud. The image pixels registered to the clustered 3D points are taken as samples to learn each object's prior knowledge. The priors are represented by Gaussian Mixture Models (GMMs) of color and 3D location information only, requiring no high-level features. We further formulate the segmentation problem within a Conditional Random Field (CRF) framework, which incorporates the learned prior models, together with hard constraints placed on the registered pixels and pairwise spatial constraints to achieve final results. "

6. ICRA 14: Learning latent structure for activity recognition [**?**]. Good overview of basics.

3.5 Offline Trackers

Using the Viterbi Algorithm [**?**]. Using a modified Viterbi algorithm to do an exhaustive search [**?**].

3.6 Temporal Speech Data

Structured prediction is used in temporal data such as speech recognition. This has yet to fully permeate object tracking in videos.

RGB-D videos provide a unique set of data to images. Objects are more easily segmented based on depth data (without transformation) alone. This can be seen in LIDAR [**?**].

It is often the case that

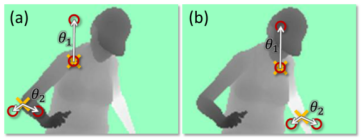

Figure 3. **Depth image features.** The yellow crosses indicates the pixel $\mathbf{x}$ being classified. The red circles indicate the offset pixels as defined in Eq. 1. In (a), the two example features give a large depth difference response. In (b), the same two features at new image locations give a much smaller response.

Figure 3.2: Shotton's Random Forest [?]

3.7 RGB-D Datasets

1. Princeton RGB-D [?]
2. Wu et al. [?]
3. Bigbird [?] (relevant?)

3.8 RGB-D Features

1. Learning rich features from rgb-d images for object detection and segmentation [?]
2. Segmentation using RGB-D data [?]
3. Shotton's Random Forest [?]: "each feature need only read at most 3 image pixels and perform at most 5 arithmetic operations; and the features can be straightforwardly implemented on the GPU. Given a larger computational budget, one could employ potentially more powerful features based on, for example, depth integrals over regions, curvature, or local descriptors"

Predicting Depth Segmenting in 3D + time [?]:

Our approach relies on the following observation: If the scene consists only of convex objects, then every depth discontinuity corresponds to an object boundary. This observation was motivated by a study on primates [ref] which concluded that depth and color perception are handled by separate channels in our nervous system that function quite

distinctly. Obviously the world does not consist only of convex objects, but nevertheless we have found this observation to have great practical utility in everyday scenes.

Eigen and Fergus [**?**] predicts depth with CNNs.

3.9 Early Tracking

Tracking pre-2005ish.

CONDENSATIONConditional Density Propagation for Visual Tracking: "The problem of tracking curves in dense visual clutter is challenging. Kalman filtering is inadequate because it is based on Gaussian densities which, being unimodal, cannot represent simultaneous alternative hypotheses. The CONDENSATION algorithm uses "factored sampling", previously applied to the interpretation of static images, in which the probability distribution of possible interpretations is represented by a randomly generated set. CONDENSATION uses learned dynamical models, together with visual observations, to propagate the random set over time. The result is highly robust tracking of agile motion. Notwithstanding the use of stochastic methods, the algorithm runs in near real-time."

http://www.cse.psu.edu/ rcollins/CollinsVLPR2012Lecture.pdf http://www.cse.psu.edu/ rcollins/-CollinsVLPR2009Lecture.pdf

3.10 Pedestrian Detection

Benson *et al* [**?**]: 10 year review paper, starting to use optical flow

Object detection of a known object category is a well-studied problem. Benenson *et al* [**?**] show four paradigms have emerged competitive: variants of Viola and Jones [**?**], histograms of oriented gradients (HOG) classified by linear support vector machines (SVM) [**?**], the deformable parts model (DPM) [**?**] and convolutional neural networks.

3.11 Object Detection

Bjorn Ommer [**? ? ? ?**] gave a talk at UBC: find 1000 random parts; group parts by their detection responses' spatial proximity in training data instances, then use

those groups as positive examples for a part.

Ommer's ECCV 12 paper [**?**]:

> we propose an approach for learning object models for detection while, simultaneously, learning to segregate objects from clutter and extracting their overall shape. For this purpose, we exclusively use bounding-box annotated training data. The approach groups fragmented object regions using the Multiple Instance Learning (MIL) framework to obtain a meaningful representation of object shape which, at the same time, crops away distracting background clutter to improve the appearance representation.

3.12 Object Proposals

Object proposals deal with finding interesting parts of an image.

Zitnick and Dollar's [**?**] Object Proposals from Edge Detections:

> the number of contours that are wholly contained in a bounding box is indicative of the likelihood of the box containing an object. We propose a simple box objectness score that measures the number of edges that exist in the box minus those that are members of contours that overlap the boxs boundary. Using efficient data structures, millions of candidate boxes can be evaluated in a fraction of a second, returning a ranked set of a few thousand top-scoring proposals. Using standard metrics, we show results that are significantly more accurate than the current state-of-the-art while being faster to compute.

[**?**] has a good overview of the field, as with Dollar's Nov 18th blogpost.

Uses Dollar's work on edge detection [**?**]

3.13 Tracklet Proposals

Tracklet proposals deal with finding interesting tubes of the video

3.14 Edge Detection

Dollar's work on edge detection [**?**]: uses random forests for structured prediction. Given 32x32 image patch, determine if each pixel in the centre 16x16 image patch belongs to an edge. Features: Piotr's integrated channel features, at each pixel and "pairwise difference features". Chug in Structured Random Forest. Achieves state of art with sharpening and multi-scale detection, running in real-time.

Idea: Dollar's work on RGBD Time series, with before and after frames.

On depth: Gupta's work outperforms Piotr's.

3.15 Thermal Images

Gade and Moeslund [**?**] give a good overview of using thermal cameras. Teutsch *et al* [**?**] also notes detection is a fundamentally different problem in thermal videos and thoroughly experiments with different feature sets for thermal object detection. A trend unique to thermal object detectors is the inclusion of a pre-processing step to find candidate regions of interest based on intensity, from which object detectors can then be used to classify the region. Many object detectors in thermal images use shape-based features, often derived from silhouettes identified by applying a threshold to intensity values. Most thermal object detection focus on humans.

3.16 Understanding Natural Images

Lenc and Vedaldi [**?**]: Understanding image representations by measuring their equivariance and equivalence

3.17 Techniques

1. Structured SVMs: vedaldi2014structuredsvm

2. Multi-resolution [**?**]: for finite resolution cameras, scale variance is needed.

FABMAP: Bag of words, descriptor, vector quantization into a visual word, Chow-liu trees for bayesian probability

3.18 Machine Learning

Gaussian Processes Bible (with matlab toolbox) [**?**]

3.19 Pedestrian Path Planning

ETH overhead dataset [**?**]

Chapter 4

Literature - Individual Paper Reviews

If it's here, it's somewhat interesting.

4.1 Papers With Good Content

Zitnick and Dollar's Object Proposals from Edges [**?**]. Insanely fast and state of art.

Shotton's Random Forests [**?**]: Quality procedures (high quality training data), solves problem.

4.2 Parking Lot Detection

Wang [**?**] Jung [**?**]

Pouria's thesis [**?**] Talk: https://www.youtube.com/watch?v=Db7EhLYf-Yk#t=1h11m07s

4.3 SLAM

LSD-Slam and PTAM for monocular seem to be state-of-art: http://vision.in.tum.de/research/lsdslam

review paper: http://www-personal.acfr.usyd.edu.au/tbailey/papers/slamtute2.

pdf

2009 review paper: http://www.le2i.cnrs.fr/IMG/publications/2172_Muhammad_El_2009.pdf

4.4 Collision Avoidance

Old paper using quadtrees[**?**]

Bigdog [**?**]

Springer handbook of robotics [**?**]: Vector field histograms, as described in Pouria's thesis [**?**]

4.5 Papers Written Well

Anything by Piotr [**?**].

4.5.1 Gao *et al* [**?**]: Transfer Learning Based Visual Tracking with Gaussian Processes Regression

Summary Another online tracking method. I haven't read into the technique yet.

Clarity Quite clear.

Method Very legitamate, compares against top 3 benchmarks circa 2014.

Comments They reference an astounding number of tracking papers. It seems sort of incremental and probably won't get cited. Why are people solving this problem? Who needs model-free online tracking?

Chapter 5

Method (My Idea)

5.1 My Idea (2 pages)

5.2 The details (5 pages)

Chapter 6

Experiments

6.1 Bulbasaur: RGBD Online Tracking Testbed Setup

Goal Get the testbed set up for RGBD data. It takes in RGBD video, outputs charts.

Code available:

1. Song *et al* [**?**] (song2013tracking): 5 RGBD videos with code for only their tracker. The code only works on Linux and is somewhat understandable. Pros: depth already present.

2. Wu *et al* [**?**] (wu2013online): 50 RGB videos WITH code for all benchmarked trackers. Code is somewhat readable, Windows-tested, but doesn't run on first pass. Pros: baseline trackers already works. Cons: hardcoded paths.

3. Kristan *et al* [**?**] (kristan2013visual): 100+ videos, superset of Wu's dataset. Supposedly generalizable. Long list of instructions to get up and running. Code seems well structured, but few comments.

4. Henriques *et al* [**?**] (henriques2015tracking): Just one tracker. Most readable code, well documented. Implements its own on all 50 RGB videos of Wu. Pure Matlab. Easy access to new featuresets.

5. Gao *et al* [**?**] (gao2014transfer): claims state-of-art. Matlab is the same format as Wu's.

Which one should I build off of? Kristan is too complicated; maybe integrate it in the future. Henriques is the cleanest, but eventually I will need the baseline trackers from Wu.

Procedure

1. I'm looking at Wu's code. Line 136 of `main_running.m` is the function call for each tracker. Each tracker returns res, a struct for the result.

2. Try and get Wu running out of the box. Change paths. If it doesn't work after 20 minutes, quit and use Henriques. I changed the path to the VLFeat library and the paths to the datasets (in `utils/configSeqs.m`). Nothing spits out.

3. Ok so go with Henriques. They use precision curves, so all we need from the ground truth is the centroid.

4. (Regardless) Port RGBD dataset to Wu's format. Wu's Data format: (x,y,w,h). Centroid is ((x+x+w)/2, (y+y+h/2))

5. Song's rgbd ground truth is in: (x, y, w, h) as well (comparing bear_front.txt to the image)

6. Changed Song's groundtruth (bear_front.txt) to (groundtruth_rect.txt) to correspond to Wu (removed last value in each row). Change rgb folder to img folder. Changed BASE_PATH.

7. made new function `loadImageFiles.m` to get an ordered list of image filenames for Song's videos

8. bear_front - Precision (20px): 0.247, FPS: 30.11 with nextpow2, bear_front - Precision (20px): 0.162, FPS: 57.76 without nextpow2.

9. changing to allow arbitrary paths

10. added a switch: wu or song
11. not getting same results on wu. with nextpow2: Basketball - Precision (20px): 0.028, FPS: 73.34. Without nexpow2: Basketball - Precision (20px): 0.923, FPS: 129.05. Fixed.
12.
13. Try Henriques with zeros appended to feature vector. Tried, it works.
14. Get color features working on Henriques?
15. (Optional) If Wu's code works: try and get Song's code in Wu's result format, otherwise get it in Henriques.
16. Song's video frames are named incorrectly. I
17. Try Henriques with depth mean as feature
18. Try Gao
19. Try Shotton features
20. Revisit the problem and assumptions.

Results

Conclusions In progress as of 2014/12/01.

FAQ

1. Why not run Song's code?

   It only works on Linux, it's not too understandable. The risk of it taking too long to understand isn't worth the benefit of it already being setup for RGBD video.

2. You're missing state-of-the-art (ECCV 14 papers, Gaussian Process paper)?

   Oops. I'll include it, thanks. For now I need to get something done.

6.2 Ivysaur

Goal Reproduce charts in Song *et al* [**?**].

Procedure

Results

Conclusions

6.3 Venusaur

Goal RGBD+t Object proposals (instead of segmentation as in [**?**])

There are many related concepts

- RGB segmentation, a well studied field
- RGB object proposals, somewhat well studied
- RGBD+t segmentation/proposals of the entire image
- RGBD+t segmentation/proposals of only specific tracks
- RGBD+t segmentation of only the object in the bounding box (depth tracking)

Procedure

Results

Conclusions

Goal

Procedure

Results

Conclusions

6.4 Charmander: Henriques Kernels

Try different kernels of henriques.

First, I modified the code of gan2015rpe to generate average precision plots over 50 videos.

Files saved in code, personal, 004charmander. Also saved in code personal gan2015rpe.

Gaussian Kernel with HOG saved as: precisionsGaussian.mat Ditto with polynomial and linear.

All three: precisionsKernels.mat

To generate plots from data, load and run:

```
figure('Number','off', 'Name',['Precisions - Average'])
hold on
plot(avgPrecisionGaussian, 'r-', 'LineWidth',3)
plot(avgPrecisionPolynomial, 'b-', 'LineWidth',3)
plot(avgPrecisionLinear, 'g-', 'LineWidth',3)
xlabel('Threshold'), ylabel('Precision')
```

2015-01-06 Tue 07:22 PM running with window_sz as a power of 2, Gaussian kernel HOG. Should give better results precisionsWindow2times.mat result: worse. interesting, I expected better, because there are more samples.

2015-01-06 Tue 07:58 PM get_features.m modified to include Piotr's channel features (requires Piotr's toolbox). Todo: include rgb-d features as well.

```
Jan 12
with window_sz as a power of 2:
With Piotr's; default except grayscale
>> run_tracker
  Basketball - Precision (20px): 0.012, FPS: 103.57
>> run_tracker
  Basketball - Precision (20px): 0.014, FPS: 137.13

with default HOG and gaussian kernel:
```

```
>> run_tracker
  Basketball - Precision (20px): 0.028, FPS: 72.15

Now with window_sz as not a power of 2:
Default, Gaussian kernel + HOG
>> run_tracker
  Basketball - Precision (20px): 0.923, FPS: 117.13

- only turn to grayscale in get_features.m

- now, using default HOG with colour
>> run_tracker
  Basketball - Precision (20px): 0.931, FPS: 103.83

- Using default HOG + Gaussian Kernel with colour on all 50:
>> run_tracker('all')
Starting matlabpool using the 'local' profile ... connected to 2 workers..mat
precisionsGaussianHOGColour.mat

MotorRolling - Precision (20px): 0.043, FPS: 50.28
    FaceOcc1 - Precision (20px): 0.828, FPS: 57.45
      Mhyang - Precision (20px): 1.000, FPS: 41.22
      Matrix - Precision (20px): 0.180, FPS: 121.91
       Dudek - Precision (20px): 0.877, FPS: 31.18
      Liquor - Precision (20px): 0.431, FPS: 68.34
        Doll - Precision (20px): 0.989, FPS: 104.90
        Dog1 - Precision (20px): 1.000, FPS: 92.37
        Deer - Precision (20px): 0.873, FPS: 29.54
     Lemming - Precision (20px): 0.275, FPS: 33.58
     Jumping - Precision (20px): 0.339, FPS: 110.77
      David3 - Precision (20px): 1.000, FPS: 32.31
     Jogging - Precision (20px): 0.231, FPS: 66.89
      David2 - Precision (20px): 1.000, FPS: 170.17
     Ironman - Precision (20px): 0.187, FPS: 65.03
        Girl - Precision (20px): 0.874, FPS: 100.89
    Freeman4 - Precision (20px): 0.530, FPS: 412.99
       David - Precision (20px): 1.000, FPS: 41.93
    Crossing - Precision (20px): 1.000, FPS: 133.52
    Freeman3 - Precision (20px): 0.911, FPS: 469.64
      Couple - Precision (20px): 0.757, FPS: 85.23
    Freeman1 - Precision (20px): 0.393, FPS: 159.47
   Football1 - Precision (20px): 0.986, FPS: 152.52
        Coke - Precision (20px): 0.931, FPS: 47.65
    Football - Precision (20px): 0.796, FPS: 73.63
    CarScale - Precision (20px): 0.806, FPS: 183.73
     CarDark - Precision (20px): 1.000, FPS: 278.80
   FleetFace - Precision (20px): 0.460, FPS: 40.18
```

```
        Car4 - Precision (20px): 0.950, FPS: 26.37
        Fish - Precision (20px): 1.000, FPS: 33.88
         Boy - Precision (20px): 1.000, FPS: 156.99
        Bolt - Precision (20px): 0.017, FPS: 115.35
  Basketball - Precision (20px): 0.931, FPS: 81.40
      Subway - Precision (20px): 1.000, FPS: 174.33
      Soccer - Precision (20px): 0.276, FPS: 32.72
      Skiing - Precision (20px): 0.074, FPS: 264.04
    FaceOcc2 - Precision (20px): 0.972, FPS: 16.93
    Skating1 - Precision (20px): 1.000, FPS: 82.53
      Tiger2 - Precision (20px): 0.855, FPS: 49.13
      Tiger1 - Precision (20px): 0.887, FPS: 25.90
     Singer2 - Precision (20px): 0.945, FPS: 16.93
     Singer1 - Precision (20px): 0.815, FPS: 37.04
     Shaking - Precision (20px): 0.027, FPS: 52.36
   Sylvester - Precision (20px): 0.843, FPS: 57.23
MountainBike - Precision (20px): 1.000, FPS: 47.24
         Suv - Precision (20px): 0.979, FPS: 79.54
    Walking2 - Precision (20px): 0.634, FPS: 38.57
       Woman - Precision (20px): 0.938, FPS: 80.81
     Walking - Precision (20px): 1.000, FPS: 140.01
     Trellis - Precision (20px): 1.000, FPS: 42.13

Average precision (20px): 0.737, Average FPS: 98.15

In hindsight, this doesn't do anything. fhog converts to grayscale. Boo.
See
colfhog_vs_fhog.png
    plots precisionsGaussian.mat (blue) and precisionsGaussianHOGColour.mat (red)

Using COLOR.
Only on the pixels, we get perfect tracking on the deer.
>> run_tracker choose gaussian gray
       Deer - Precision (20px): 1.000, FPS: 12.56

On everything:
>> run_tracker all gaussian gray
MotorRolling - Precision (20px): 0.049, FPS: 14.63
      Mhyang - Precision (20px): 1.000, FPS: 20.80
      Matrix - Precision (20px): 0.100, FPS: 47.52
    FaceOcc1 - Precision (20px): 0.946, FPS: 7.20
       Dudek - Precision (20px): 0.751, FPS: 16.40
      Liquor - Precision (20px): 0.195, FPS: 10.26
        Doll - Precision (20px): 0.602, FPS: 35.50
        Dog1 - Precision (20px): 1.000, FPS: 37.01
        Deer - Precision (20px): 1.000, FPS: 7.04
      David3 - Precision (20px): 0.583, FPS: 7.28
```

```
      David2 – Precision (20px): 1.000, FPS: 68.03
       David – Precision (20px): 0.584, FPS: 14.59
    Crossing – Precision (20px): 1.000, FPS: 100.30
      Couple – Precision (20px): 0.571, FPS: 34.17
        Coke – Precision (20px): 0.962, FPS: 19.99
    CarScale – Precision (20px): 0.694, FPS: 75.82
     CarDark – Precision (20px): 1.000, FPS: 100.74
     Lemming – Precision (20px): 0.391, FPS: 6.28
     Jumping – Precision (20px): 0.153, FPS: 55.79
     Jogging – Precision (20px): 0.228, FPS: 28.74
     Ironman – Precision (20px): 0.139, FPS: 12.93
        Girl – Precision (20px): 0.812, FPS: 58.41
    Freeman4 – Precision (20px): 0.191, FPS: 264.42
    Freeman3 – Precision (20px): 0.926, FPS: 523.62
    Freeman1 – Precision (20px): 0.439, FPS: 157.85
    Football1 – Precision (20px): 0.514, FPS: 44.03
    Football – Precision (20px): 0.796, FPS: 37.67
        Car4 – Precision (20px): 0.299, FPS: 5.93
         Boy – Precision (20px): 1.000, FPS: 44.48
   FleetFace – Precision (20px): 0.484, FPS: 13.35
        Bolt – Precision (20px): 0.029, FPS: 42.81
        Fish – Precision (20px): 1.000, FPS: 19.68
  Basketball – Precision (20px): 0.295, FPS: 15.17
      Subway – Precision (20px): 0.240, FPS: 44.98
    FaceOcc2 – Precision (20px): 1.000, FPS: 8.53
      Soccer – Precision (20px): 0.255, FPS: 5.40
       Skiing – Precision (20px): 0.136, FPS: 117.38
      Tiger2 – Precision (20px): 0.532, FPS: 11.65
    Skating1 – Precision (20px): 1.000, FPS: 23.54
      Tiger1 – Precision (20px): 0.573, FPS: 9.29
   Sylvester – Precision (20px): 0.964, FPS: 19.85
     Singer2 – Precision (20px): 0.036, FPS: 3.35
         Suv – Precision (20px): 0.521, FPS: 16.44
     Singer1 – Precision (20px): 0.521, FPS: 4.99
    Walking2 – Precision (20px): 0.404, FPS: 12.18
     Shaking – Precision (20px): 0.011, FPS: 9.31
     Walking – Precision (20px): 0.813, FPS: 26.79
MountainBike – Precision (20px): 1.000, FPS: 10.29
       Woman – Precision (20px): 0.250, FPS: 26.66
     Trellis – Precision (20px): 0.257, FPS: 9.15

Average precision (20px): 0.565, Average FPS: 46.16
```

TODO test Piotrs, make it match HOG

Piotrs DOES NOT match plain fHog, does worse on basketball, I don't know why.
I checked down to the code, I can't figure out why they're different even though they both run GradientHist and

```
Anyway. Piotr's integral images, Color (when available, else just intensity) + HOG, Gaussian Kernel.
>> run_tracker
  Basketball - Precision (20px): 0.926, FPS: 83.54

PiotrColorHist.png

>> run_tracker all
MotorRolling - Precision (20px): 0.043, FPS: 38.68
    FaceOcc1 - Precision (20px): 0.834, FPS: 41.20
      Mhyang - Precision (20px): 1.000, FPS: 32.43
      Matrix - Precision (20px): 0.370, FPS: 84.48
       Dudek - Precision (20px): 0.880, FPS: 26.45
      Liquor - Precision (20px): 0.407, FPS: 52.23
        Doll - Precision (20px): 0.976, FPS: 77.28
        Dog1 - Precision (20px): 1.000, FPS: 79.88
        Deer - Precision (20px): 0.873, FPS: 23.30
     Lemming - Precision (20px): 0.424, FPS: 29.52
      David3 - Precision (20px): 1.000, FPS: 43.10
     Jumping - Precision (20px): 0.083, FPS: 140.84
      David2 - Precision (20px): 1.000, FPS: 166.55
     Jogging - Precision (20px): 0.231, FPS: 76.74
     Ironman - Precision (20px): 0.175, FPS: 65.04
        Girl - Precision (20px): 0.862, FPS: 93.74
       David - Precision (20px): 1.000, FPS: 40.41
    Crossing - Precision (20px): 1.000, FPS: 131.48
    Freeman4 - Precision (20px): 0.230, FPS: 252.84
      Couple - Precision (20px): 0.200, FPS: 92.09
    Freeman3 - Precision (20px): 0.915, FPS: 276.00
    Freeman1 - Precision (20px): 0.387, FPS: 167.93
   Football1 - Precision (20px): 0.932, FPS: 122.50
        Coke - Precision (20px): 0.856, FPS: 51.14
    Football - Precision (20px): 0.796, FPS: 75.94
    CarScale - Precision (20px): 0.802, FPS: 131.46
     CarDark - Precision (20px): 1.000, FPS: 164.52
   FleetFace - Precision (20px): 0.465, FPS: 33.21
        Car4 - Precision (20px): 0.956, FPS: 22.18
        Fish - Precision (20px): 1.000, FPS: 26.86
         Boy - Precision (20px): 1.000, FPS: 93.31
        Bolt - Precision (20px): 1.000, FPS: 84.11
  Basketball - Precision (20px): 0.926, FPS: 63.20
      Subway - Precision (20px): 1.000, FPS: 122.48
      Soccer - Precision (20px): 0.156, FPS: 25.30
      Skiing - Precision (20px): 0.074, FPS: 162.46
    FaceOcc2 - Precision (20px): 0.835, FPS: 15.05
    Skating1 - Precision (20px): 1.000, FPS: 62.09
      Tiger2 - Precision (20px): 0.732, FPS: 38.77
```

```
      Tiger1 - Precision (20px): 0.997, FPS: 22.36
     Singer2 - Precision (20px): 0.036, FPS: 15.18
     Singer1 - Precision (20px): 1.000, FPS: 32.13
     Shaking - Precision (20px): 0.025, FPS: 38.13
   Sylvester - Precision (20px): 0.844, FPS: 47.13
MountainBike - Precision (20px): 1.000, FPS: 34.81
         Suv - Precision (20px): 0.979, FPS: 58.13
    Walking2 - Precision (20px): 0.676, FPS: 29.76
       Woman - Precision (20px): 0.940, FPS: 57.33
     Walking - Precision (20px): 1.000, FPS: 84.44
     Trellis - Precision (20px): 1.000, FPS: 29.40

Average precision (20px): 0.718, Average FPS: 75.51

above is saved as: 004charmander/precisionspiotrColorHist.mat


for i = 1:50
    prec(i) = all_precisions{i}(20);
end


The above, except with the additional gradient magnitude

>> run_tracker all
MotorRolling - Precision (20px): 0.043, FPS: 38.03
    FaceOcc1 - Precision (20px): 0.834, FPS: 40.74
      Mhyang - Precision (20px): 1.000, FPS: 33.05
      Matrix - Precision (20px): 0.370, FPS: 77.94
       Dudek - Precision (20px): 0.880, FPS: 26.00
      Liquor - Precision (20px): 0.407, FPS: 49.41
        Doll - Precision (20px): 0.976, FPS: 72.27
        Dog1 - Precision (20px): 1.000, FPS: 70.40
        Deer - Precision (20px): 0.873, FPS: 21.67
     Lemming - Precision (20px): 0.424, FPS: 25.78
      David3 - Precision (20px): 1.000, FPS: 39.81
     Jumping - Precision (20px): 0.083, FPS: 132.34
      David2 - Precision (20px): 1.000, FPS: 153.08
     Jogging - Precision (20px): 0.231, FPS: 70.43
     Ironman - Precision (20px): 0.175, FPS: 61.43
       David - Precision (20px): 1.000, FPS: 39.15
        Girl - Precision (20px): 0.862, FPS: 86.95
    Crossing - Precision (20px): 1.000, FPS: 112.19
    Freeman4 - Precision (20px): 0.866, FPS: 220.31
      Couple - Precision (20px): 0.200, FPS: 87.59
    Freeman3 - Precision (20px): 0.915, FPS: 247.36
    Freeman1 - Precision (20px): 0.387, FPS: 144.40
```

```
   Football1 - Precision (20px): 0.932, FPS: 112.94
        Coke - Precision (20px): 0.856, FPS: 47.18
    CarScale - Precision (20px): 0.802, FPS: 120.15
    Football - Precision (20px): 0.796, FPS: 71.12
     CarDark - Precision (20px): 1.000, FPS: 152.68
   FleetFace - Precision (20px): 0.465, FPS: 32.12
        Car4 - Precision (20px): 0.962, FPS: 21.49
         Boy - Precision (20px): 1.000, FPS: 84.50
        Fish - Precision (20px): 1.000, FPS: 25.66
        Bolt - Precision (20px): 1.000, FPS: 66.79
  Basketball - Precision (20px): 0.924, FPS: 56.89
      Subway - Precision (20px): 1.000, FPS: 113.46
      Soccer - Precision (20px): 0.156, FPS: 19.31
       Skiing - Precision (20px): 0.074, FPS: 68.55
    FaceOcc2 - Precision (20px): 0.835, FPS: 12.23
    Skating1 - Precision (20px): 1.000, FPS: 42.15
      Tiger2 - Precision (20px): 0.732, FPS: 30.59
      Tiger1 - Precision (20px): 0.997, FPS: 20.79
     Singer2 - Precision (20px): 0.036, FPS: 13.82
     Singer1 - Precision (20px): 1.000, FPS: 28.26
     Shaking - Precision (20px): 0.025, FPS: 35.45
   Sylvester - Precision (20px): 0.844, FPS: 42.54
MountainBike - Precision (20px): 1.000, FPS: 33.43
         Suv - Precision (20px): 0.979, FPS: 56.94
    Walking2 - Precision (20px): 0.676, FPS: 28.89
       Woman - Precision (20px): 0.940, FPS: 56.57
     Walking - Precision (20px): 1.000, FPS: 87.59
     Trellis - Precision (20px): 1.000, FPS: 31.33

Average precision (20px): 0.731, Average FPS: 67.28

precisionsPiotrColorMagHist.mat

>> run_tracker all linear
MotorRolling - Precision (20px): 0.043, FPS: 60.72
    FaceOcc1 - Precision (20px): 0.834, FPS: 59.78
      Mhyang - Precision (20px): 1.000, FPS: 56.30
      Matrix - Precision (20px): 0.360, FPS: 101.70
       Dudek - Precision (20px): 0.880, FPS: 48.24
      Liquor - Precision (20px): 0.407, FPS: 73.15
        Doll - Precision (20px): 0.976, FPS: 100.81
        Dog1 - Precision (20px): 1.000, FPS: 104.10
        Deer - Precision (20px): 0.873, FPS: 38.24
      David3 - Precision (20px): 1.000, FPS: 56.59
     Lemming - Precision (20px): 0.427, FPS: 39.16
     Jumping - Precision (20px): 0.652, FPS: 148.49
      David2 - Precision (20px): 1.000, FPS: 158.62
```

```
      Jogging - Precision (20px): 0.231, FPS: 87.10
      Ironman - Precision (20px): 0.151, FPS: 77.21
        David - Precision (20px): 1.000, FPS: 52.17
     Crossing - Precision (20px): 1.000, FPS: 163.24
         Girl - Precision (20px): 0.862, FPS: 127.86
       Couple - Precision (20px): 0.114, FPS: 124.20
     Freeman4 - Precision (20px): 0.866, FPS: 265.91
     Freeman3 - Precision (20px): 0.913, FPS: 286.02
     Freeman1 - Precision (20px): 0.387, FPS: 195.51
         Coke - Precision (20px): 0.856, FPS: 70.14
    Football1 - Precision (20px): 0.892, FPS: 157.22
     CarScale - Precision (20px): 0.806, FPS: 165.16
     Football - Precision (20px): 0.796, FPS: 113.49
      CarDark - Precision (20px): 1.000, FPS: 189.61
         Car4 - Precision (20px): 0.956, FPS: 37.17
    FleetFace - Precision (20px): 0.484, FPS: 56.61
         Fish - Precision (20px): 1.000, FPS: 52.96
          Boy - Precision (20px): 1.000, FPS: 125.23
         Bolt - Precision (20px): 1.000, FPS: 111.52
   Basketball - Precision (20px): 0.926, FPS: 80.61
       Subway - Precision (20px): 1.000, FPS: 148.34
     FaceOcc2 - Precision (20px): 0.835, FPS: 26.04
       Soccer - Precision (20px): 0.156, FPS: 44.04
        Skiing - Precision (20px): 0.074, FPS: 177.33
       Tiger2 - Precision (20px): 0.762, FPS: 52.37
     Skating1 - Precision (20px): 1.000, FPS: 80.15
       Tiger1 - Precision (20px): 0.997, FPS: 36.91
      Singer2 - Precision (20px): 0.036, FPS: 26.52
    Sylvester - Precision (20px): 0.844, FPS: 75.44
      Singer1 - Precision (20px): 1.000, FPS: 45.35
      Shaking - Precision (20px): 0.025, FPS: 59.72
          Suv - Precision (20px): 0.978, FPS: 88.62
MountainBike - Precision (20px): 1.000, FPS: 62.56
     Walking2 - Precision (20px): 0.676, FPS: 51.43
        Woman - Precision (20px): 0.940, FPS: 89.68
      Walking - Precision (20px): 1.000, FPS: 133.17
      Trellis - Precision (20px): 1.000, FPS: 50.45

Average precision (20px): 0.740, Average FPS: 96.66

Color + Mag, Linear Kernel
Average precision (20px): 0.557, Average FPS: 185.07

Linear Kernel, just gray pixels

>> run_tracker all linear gray
MotorRolling - Precision (20px): 0.043, FPS: 43.12
```

```
      Mhyang – Precision (20px): 1.000, FPS: 46.12
      Matrix – Precision (20px): 0.100, FPS: 119.72
    FaceOcc1 – Precision (20px): 0.899, FPS: 24.00
       Dudek – Precision (20px): 0.785, FPS: 32.67
      Liquor – Precision (20px): 0.195, FPS: 32.97
        Doll – Precision (20px): 0.648, FPS: 95.14
        Dog1 – Precision (20px): 1.000, FPS: 76.32
        Deer – Precision (20px): 0.887, FPS: 22.87
      David3 – Precision (20px): 0.155, FPS: 24.74
      David2 – Precision (20px): 1.000, FPS: 142.40
       David – Precision (20px): 0.410, FPS: 37.73
    Crossing – Precision (20px): 1.000, FPS: 208.09
      Couple – Precision (20px): 0.107, FPS: 92.98
     Lemming – Precision (20px): 0.170, FPS: 19.12
     Jumping – Precision (20px): 0.048, FPS: 110.38
        Coke – Precision (20px): 0.440, FPS: 62.35
     Jogging – Precision (20px): 0.231, FPS: 80.99
    CarScale – Precision (20px): 0.687, FPS: 175.88
     CarDark – Precision (20px): 1.000, FPS: 239.35
     Ironman – Precision (20px): 0.145, FPS: 40.25
        Girl – Precision (20px): 1.000, FPS: 158.62
    Freeman4 – Precision (20px): 0.304, FPS: 552.90
    Freeman3 – Precision (20px): 0.915, FPS: 820.93
    Freeman1 – Precision (20px): 0.463, FPS: 294.71
   Football1 – Precision (20px): 0.743, FPS: 112.46
    Football – Precision (20px): 0.793, FPS: 81.15
        Car4 – Precision (20px): 0.281, FPS: 12.34
   FleetFace – Precision (20px): 0.625, FPS: 26.39
         Boy – Precision (20px): 0.846, FPS: 124.82
        Fish – Precision (20px): 0.057, FPS: 41.15
        Bolt – Precision (20px): 0.034, FPS: 113.05
  Basketball – Precision (20px): 0.131, FPS: 45.22
      Subway – Precision (20px): 0.240, FPS: 130.44
    FaceOcc2 – Precision (20px): 0.986, FPS: 20.55
      Tiger2 – Precision (20px): 0.110, FPS: 33.64
      Soccer – Precision (20px): 0.158, FPS: 16.09
       Skiing – Precision (20px): 0.136, FPS: 261.79
    Skating1 – Precision (20px): 0.700, FPS: 67.58
      Tiger1 – Precision (20px): 0.282, FPS: 28.87
     Singer2 – Precision (20px): 0.036, FPS: 10.82
   Sylvester – Precision (20px): 0.807, FPS: 42.25
     Singer1 – Precision (20px): 0.949, FPS: 14.13
         Suv – Precision (20px): 0.521, FPS: 30.02
      Shaking – Precision (20px): 0.008, FPS: 30.38
    Walking2 – Precision (20px): 0.452, FPS: 35.02
MountainBike – Precision (20px): 1.000, FPS: 30.58
     Walking – Precision (20px): 0.852, FPS: 79.08
```

```
      Woman - Precision (20px): 0.250, FPS: 78.06
     Trellis - Precision (20px): 0.279, FPS: 31.12

Average precision (20px): 0.498, Average FPS: 101.03

below is
KCF  on raw
Gaussian, raw pixels
precisionGaussianPixels.mat

>> run_tracker all gaussian gray
MotorRolling - Precision (20px): 0.043, FPS: 21.82
      Mhyang - Precision (20px): 1.000, FPS: 22.80
      Matrix - Precision (20px): 0.160, FPS: 57.96
    FaceOcc1 - Precision (20px): 0.934, FPS: 11.39
       Dudek - Precision (20px): 0.751, FPS: 14.18
      Liquor - Precision (20px): 0.198, FPS: 13.92
        Doll - Precision (20px): 0.631, FPS: 42.89
        Dog1 - Precision (20px): 1.000, FPS: 34.22
        Deer - Precision (20px): 1.000, FPS: 9.47
      David3 - Precision (20px): 0.393, FPS: 10.15
      David2 - Precision (20px): 1.000, FPS: 64.48
       David - Precision (20px): 0.512, FPS: 21.99
    Crossing - Precision (20px): 1.000, FPS: 131.33
      Couple - Precision (20px): 0.621, FPS: 46.11
     Lemming - Precision (20px): 0.381, FPS: 8.25
        Coke - Precision (20px): 0.873, FPS: 33.84
    CarScale - Precision (20px): 0.687, FPS: 74.32
     Jumping - Precision (20px): 0.153, FPS: 45.71
     CarDark - Precision (20px): 1.000, FPS: 117.50
     Jogging - Precision (20px): 0.231, FPS: 35.95
     Ironman - Precision (20px): 0.145, FPS: 16.20
        Girl - Precision (20px): 0.890, FPS: 83.74
    Freeman4 - Precision (20px): 0.191, FPS: 256.92
    Freeman3 - Precision (20px): 0.926, FPS: 517.43
    Freeman1 - Precision (20px): 0.439, FPS: 148.29
   Football1 - Precision (20px): 0.743, FPS: 58.50
    Football - Precision (20px): 0.796, FPS: 36.57
   FleetFace - Precision (20px): 0.484, FPS: 12.91
        Car4 - Precision (20px): 0.299, FPS: 5.63
         Boy - Precision (20px): 0.846, FPS: 63.66
        Fish - Precision (20px): 1.000, FPS: 21.34
        Bolt - Precision (20px): 0.034, FPS: 63.03
  Basketball - Precision (20px): 0.299, FPS: 22.91
      Subway - Precision (20px): 0.240, FPS: 64.09
    FaceOcc2 - Precision (20px): 1.000, FPS: 9.15
      Soccer - Precision (20px): 0.222, FPS: 8.22
```

```
      Skiing - Precision (20px): 0.136, FPS: 156.57
      Tiger2 - Precision (20px): 0.110, FPS: 17.10
    Skating1 - Precision (20px): 0.700, FPS: 35.67
      Tiger1 - Precision (20px): 0.678, FPS: 14.55
     Singer2 - Precision (20px): 0.036, FPS: 5.49
   Sylvester - Precision (20px): 0.964, FPS: 23.02
     Singer1 - Precision (20px): 0.507, FPS: 7.66
         Suv - Precision (20px): 0.521, FPS: 17.66
     Shaking - Precision (20px): 0.008, FPS: 16.51
MountainBike - Precision (20px): 1.000, FPS: 16.77
    Walking2 - Precision (20px): 0.412, FPS: 19.62
       Woman - Precision (20px): 0.250, FPS: 42.95
     Walking - Precision (20px): 0.830, FPS: 46.87
     Trellis - Precision (20px): 0.243, FPS: 17.02

Average precision (20px): 0.550, Average FPS: 52.89


below:
precisionsPolynomialPixels.mat
>> run_tracker all polynomial gray
MotorRolling - Precision (20px): 0.043, FPS: 18.82
    FaceOcc1 - Precision (20px): 0.947, FPS: 10.12
      Mhyang - Precision (20px): 0.999, FPS: 18.42
      Matrix - Precision (20px): 0.100, FPS: 54.05
       Dudek - Precision (20px): 0.755, FPS: 15.20
      Liquor - Precision (20px): 0.197, FPS: 14.97
        Doll - Precision (20px): 0.490, FPS: 39.17
        Dog1 - Precision (20px): 1.000, FPS: 34.52
        Deer - Precision (20px): 0.972, FPS: 9.46
      David3 - Precision (20px): 0.155, FPS: 10.13
      David2 - Precision (20px): 1.000, FPS: 61.81
       David - Precision (20px): 0.459, FPS: 16.63
    Crossing - Precision (20px): 1.000, FPS: 100.70
      Couple - Precision (20px): 0.250, FPS: 37.97
     Lemming - Precision (20px): 0.382, FPS: 7.95
     Jumping - Precision (20px): 0.048, FPS: 46.56
        Coke - Precision (20px): 0.876, FPS: 26.26
    CarScale - Precision (20px): 0.687, FPS: 80.50
     Jogging - Precision (20px): 0.231, FPS: 31.55
     CarDark - Precision (20px): 1.000, FPS: 108.64
     Ironman - Precision (20px): 0.133, FPS: 16.06
        Girl - Precision (20px): 0.890, FPS: 68.88
    Freeman4 - Precision (20px): 0.300, FPS: 236.23
    Freeman3 - Precision (20px): 0.926, FPS: 428.67
    Freeman1 - Precision (20px): 0.457, FPS: 127.33
   Football1 - Precision (20px): 0.743, FPS: 51.57
```

```
    Football - Precision (20px): 0.796, FPS: 30.38
   FleetFace - Precision (20px): 0.598, FPS: 10.87
        Car4 - Precision (20px): 0.281, FPS: 4.90
        Fish - Precision (20px): 1.000, FPS: 15.38
         Boy - Precision (20px): 0.846, FPS: 47.07
        Bolt - Precision (20px): 0.014, FPS: 51.31
  Basketball - Precision (20px): 0.081, FPS: 19.64
      Subway - Precision (20px): 0.240, FPS: 53.75
    FaceOcc2 - Precision (20px): 1.000, FPS: 7.63
      Soccer - Precision (20px): 0.158, FPS: 6.79
       Skiing - Precision (20px): 0.136, FPS: 122.00
      Tiger2 - Precision (20px): 0.115, FPS: 12.93
    Skating1 - Precision (20px): 0.700, FPS: 26.28
      Tiger1 - Precision (20px): 0.624, FPS: 11.15
     Singer2 - Precision (20px): 0.036, FPS: 4.81
   Sylvester - Precision (20px): 0.859, FPS: 19.03
     Singer1 - Precision (20px): 0.707, FPS: 6.77
     Shaking - Precision (20px): 0.008, FPS: 13.97
         Suv - Precision (20px): 0.519, FPS: 15.19
MountainBike - Precision (20px): 1.000, FPS: 14.93
    Walking2 - Precision (20px): 0.430, FPS: 17.12
       Woman - Precision (20px): 0.250, FPS: 35.02
     Walking - Precision (20px): 0.818, FPS: 40.59
     Trellis - Precision (20px): 0.285, FPS: 13.18

Average precision (20px): 0.531, Average FPS: 45.46

Running Mag only, linear. Gets a straight line
precisionsLinearMag.mat

>> run_tracker all linear hog
MotorRolling - Precision (20px): 0.049, FPS: 179.46
    FaceOcc1 - Precision (20px): 0.231, FPS: 168.04
      Mhyang - Precision (20px): 0.300, FPS: 215.63
      Matrix - Precision (20px): 0.160, FPS: 216.20
       Dudek - Precision (20px): 0.078, FPS: 160.80
      Liquor - Precision (20px): 0.427, FPS: 167.56
        Doll - Precision (20px): 0.151, FPS: 218.23
        Dog1 - Precision (20px): 0.177, FPS: 290.76
        Deer - Precision (20px): 0.028, FPS: 138.64
      David3 - Precision (20px): 0.024, FPS: 169.88
     Lemming - Precision (20px): 0.022, FPS: 141.07
      David2 - Precision (20px): 0.283, FPS: 348.80
     Jumping - Precision (20px): 0.323, FPS: 328.69
     Jogging - Precision (20px): 0.459, FPS: 219.68
       David - Precision (20px): 0.238, FPS: 166.40
     Ironman - Precision (20px): 0.133, FPS: 212.62
```

```
    Crossing - Precision (20px): 0.117, FPS: 286.88
      Couple - Precision (20px): 0.093, FPS: 246.87
        Girl - Precision (20px): 0.896, FPS: 259.10
    Freeman4 - Precision (20px): 0.177, FPS: 370.76
        Coke - Precision (20px): 0.055, FPS: 183.34
    Freeman3 - Precision (20px): 0.146, FPS: 389.91
    CarScale - Precision (20px): 0.052, FPS: 257.37
    Freeman1 - Precision (20px): 0.061, FPS: 340.73
   Football1 - Precision (20px): 0.054, FPS: 250.60
     CarDark - Precision (20px): 0.145, FPS: 275.01
    Football - Precision (20px): 0.061, FPS: 276.79
        Car4 - Precision (20px): 0.237, FPS: 161.24
   FleetFace - Precision (20px): 0.008, FPS: 211.74
         Boy - Precision (20px): 0.196, FPS: 249.55
        Fish - Precision (20px): 0.200, FPS: 204.52
        Bolt - Precision (20px): 0.317, FPS: 244.08
    FaceOcc2 - Precision (20px): 0.595, FPS: 154.53
      Subway - Precision (20px): 0.074, FPS: 278.94
  Basketball - Precision (20px): 0.073, FPS: 205.71
      Soccer - Precision (20px): 0.176, FPS: 149.26
       Skiing - Precision (20px): 0.049, FPS: 271.55
      Tiger2 - Precision (20px): 0.011, FPS: 155.60
    Skating1 - Precision (20px): 0.215, FPS: 202.70
      Tiger1 - Precision (20px): 0.124, FPS: 138.72
     Singer2 - Precision (20px): 0.057, FPS: 118.32
   Sylvester - Precision (20px): 0.225, FPS: 244.24
     Singer1 - Precision (20px): 0.017, FPS: 142.30
     Shaking - Precision (20px): 0.038, FPS: 165.45
         Suv - Precision (20px): 0.099, FPS: 241.62
MountainBike - Precision (20px): 0.048, FPS: 179.24
    Walking2 - Precision (20px): 0.064, FPS: 182.52
       Woman - Precision (20px): 0.307, FPS: 235.09
     Walking - Precision (20px): 0.017, FPS: 285.28
     Trellis - Precision (20px): 0.385, FPS: 177.34

Average precision (20px): 0.169, Average FPS: 221.59


Gaussian, Mag only

>> run_tracker all gaussian hog
MotorRolling - Precision (20px): 0.006, FPS: 153.21
    FaceOcc1 - Precision (20px): 0.231, FPS: 141.40
      Mhyang - Precision (20px): 0.001, FPS: 162.46
      Matrix - Precision (20px): 0.010, FPS: 218.33
       Dudek - Precision (20px): 0.078, FPS: 123.44
      Liquor - Precision (20px): 0.427, FPS: 145.19
```

```
      Doll - Precision (20px): 0.151, FPS: 192.21
      Dog1 - Precision (20px): 0.001, FPS: 245.60
      Deer - Precision (20px): 0.014, FPS: 116.73
    David3 - Precision (20px): 0.008, FPS: 140.79
  Lemming - Precision (20px): 0.001, FPS: 119.03
   Jumping - Precision (20px): 0.323, FPS: 301.10
    David2 - Precision (20px): 0.004, FPS: 318.43
   Jogging - Precision (20px): 0.003, FPS: 192.78
  Ironman - Precision (20px): 0.006, FPS: 179.55
     David - Precision (20px): 0.238, FPS: 138.18
  Crossing - Precision (20px): 0.008, FPS: 252.68
    Couple - Precision (20px): 0.007, FPS: 222.79
      Girl - Precision (20px): 0.002, FPS: 229.16
  Freeman4 - Precision (20px): 0.177, FPS: 352.47
      Coke - Precision (20px): 0.055, FPS: 157.91
  Freeman3 - Precision (20px): 0.007, FPS: 356.27
  CarScale - Precision (20px): 0.052, FPS: 230.61
  Freeman1 - Precision (20px): 0.006, FPS: 302.50
 Football1 - Precision (20px): 0.054, FPS: 236.38
   CarDark - Precision (20px): 0.005, FPS: 286.65
  Football - Precision (20px): 0.003, FPS: 252.59
      Car4 - Precision (20px): 0.005, FPS: 121.32
 FleetFace - Precision (20px): 0.001, FPS: 159.29
       Boy - Precision (20px): 0.005, FPS: 211.75
      Fish - Precision (20px): 0.002, FPS: 146.71
      Bolt - Precision (20px): 0.003, FPS: 215.17
  FaceOcc2 - Precision (20px): 0.001, FPS: 109.08
    Subway - Precision (20px): 0.006, FPS: 252.83
Basketball - Precision (20px): 0.073, FPS: 177.02
    Soccer - Precision (20px): 0.176, FPS: 119.59
     Skiing - Precision (20px): 0.049, FPS: 267.34
    Tiger2 - Precision (20px): 0.003, FPS: 136.94
  Skating1 - Precision (20px): 0.005, FPS: 179.24
    Tiger1 - Precision (20px): 0.014, FPS: 112.25
   Singer2 - Precision (20px): 0.003, FPS: 91.46
 Sylvester - Precision (20px): 0.001, FPS: 193.49
   Singer1 - Precision (20px): 0.017, FPS: 117.21
    Shaking - Precision (20px): 0.011, FPS: 140.70
       Suv - Precision (20px): 0.099, FPS: 199.61
MountainBike - Precision (20px): 0.004, FPS: 143.27
  Walking2 - Precision (20px): 0.002, FPS: 146.94
     Woman - Precision (20px): 0.002, FPS: 196.53
   Walking - Precision (20px): 0.002, FPS: 234.90
    Trellis - Precision (20px): 0.002, FPS: 140.76

Average precision (20px): 0.047, Average FPS: 191.64
```

6.5 Charmeleon: Get Asus Point Cloud Data

(January 12) Get images/videos of Asus point cloud data. Final goal: negative space object detection.

6.6 Charizard

Writeup: tracklet proposals.

6.7 Squirtle: Danelljan Color Tracker

Color tracking code, modified to work with modified version of henriques2015 (gan2014rpe)

```
code/personal/007squirtle

results/paramsWorking.mat: params variable of original.
>> run_tracker
Center Location Error: 5.11 pixels
Distance Precision: 100 %
Overlap Precision: 100 %
Speed: 42.6 fps

My modified version:
results/paramsWorkingNope.mat
>> run_tracker
        deer - Precision (20px): 0.000, FPS: 43.22

bug fixed!
params.init_pos = floor(pos) + floor(target_sz/2);
to
params.init_pos = floor(pos);
(it was already done in my new load function)

For Color
>> run_tracker
        deer - Precision (20px): 1.000, FPS: 43.59
For Charmander
>> run_tracker
        Deer - Precision (20px): 0.873, FPS: 38.22
  Basketball - Precision (20px): 0.999, FPS: 79.97

color is bettah!
```

6.8 Wartortle

```
creating dataset/wu2013visual_color
removing colorless
Car4
David2
Dog1
Dudek
```

eh..not worth it, yet. I was going to make this a colour-video only dataset.

6.9 Blastoise

Get Asus RGBD Camera taking pictures.

```
Step 1:
found camera

Step 2: install driver
Installing Asus Xtion Pro Live RGBD Camera

http://support.asus.com/download.aspx?SLanguage=en&p=19&m=Xtion+PRO+LIVE&hashedid=hahEFPMWY9UVDL7z

download from 'utilities'
XtionCenter Package
Xtion Apps for Windows 8/8.1
Please backup and remove the original applications/SDK before installing the Windows 8/8.1 version

running V1144_20140121.exe

(during running, some windows message in the bottom right popped up saying
'not enough resources for USB 2.0' or something like that)
Not enough USB Controller resources

http://answers.microsoft.com/en-us/surface/forum/surfpro-surfdevice/not-enough-usb-controller-resources-error/f
so apparently i have an unpowered usb device

From PDF:
Please run <Samples\Bin\Release\
NiViewer.exe> after installation.

The default path is <C:\Program Files\
OpenNI\Samples\Bin\Release\NiViewer.
exe>. SDK is successfully installed when the
depth map shows on the left corner of the
screen and the color image shows on the
right corner of the screen.
```

4.
The sample code is included in the SDK for
your reference. You can find the sample code
from the following paths.

a.
 OpenNI samples: The default path is:
<C:\Program Files\OpenNI\Samples>.

b. NITE samples: The default path is:
<C:\Program Files\Prime Sense\NITE\
Samples>

Doesn't work. Boo.

6.10 Caterpie

Getting Piotr's Object Proposals up and running on RGBD images.

cloned git reposityry, now in C:\Code\zitnick2014edge\

Could not compile. I don't have OpenMP for parralelization.

Works with prebuilt binaries.

edgeBoxesDemo.m gives object proposal boxes in Piotr's bbs format.

Added:
```
% Edge boxes show
figure(2)
imshow(I);
bbApply('draw', bbs, 'red');
```

to show all object proposals

GANedgeBoxesDemo.m: initial test, showing values for bbApply

Conclusions: Gets 2d boxes alright.

Getting NY dataset to test RGBD object proposals.
silberman2012indoor

Whoa. http://www.cs.berkeley.edu/~sgupta/
S Gupta might have an extension.

```
Additional trained models for edge detection (only BSDS is included):
http://vision.ucsd.edu/~pdollar/files/datasets/edges/

I downloaded them and put them in zitnick2014edge\models\additional\
```

6.11 Metapod

Test Piotr's object proposals in NYUD dataset.

6.12 Butterfree

6.13 Weedle

6.14 Kakuna

6.15 Beedrill

6.16 Pidgey

6.17 Pidgeotto

6.18 Pidgeot

6.19 Rattata

6.20 Raticate

6.21 Spearow

6.22 Fearow

6.23 Ekans

6.24 Arbok

6.25 Pikachu

6.26 Raichu

6.27 Sandshrew

6.28 Sandslash

6.29 NidoranM

6.30 Nidorina

Appendix A

Metachapter: How to Do Research

Purpose: to give some guidance as to what to do next.

A.1 How to do research

Todo: make into checklist form: How to do research [**?**] [**?**] [**?**]

Chapman in 1988 [**?**]: informal rules of thumb advice. Each section corresponds to a chapter in his document, in the same order.

A.2 How to read

Keshav *et al* [**?**], the three pass approach. Pass 1:

1. Carefully read the title, abstract, and introduction
2. Read the section and sub-section headings, but ignore everything else
3. Glance at the mathematical content (if any) to determine the underlying theoretical foundations
4. Read the conclusions
5. Glance over the references, mentally ticking off the ones youve already read

At the end of the first pass, you should be able to answer the five Cs:

1. Category: What type of paper is this? A measurement paper? An analysis of an existing system? A description of a research prototype?
2. Context: Which other papers is it related to? Which theoretical bases were used to analyze the problem?
3. Correctness: Do the assumptions appear to be valid?
4. Contributions: What are the papers main contributions?
5. Clarity: Is the paper well written?

If a reviewer cannot understand the gist after one pass, the paper will likely be rejected; if a reader cannot understand the highlights of the paper after five minutes, the paper will likely never be read. For these reasons, a graphical abstract that summarizes a paper with a single well-chosen figure is an excellent idea and can be increasingly found in scientific journals.

Second pass:

1. Look carefully at the figures, diagrams and other illustrations in the paper. Pay special attention to graphs. Are the axes properly labeled? Are results shown with error bars, so that conclusions are statistically significant? Common mistakes like these will separate rushed, shoddy work from the truly excellent.

2. Remember to mark relevant unread references for further reading (this is a good way to learn more about the background of the paper).

The second pass should take up to an hour for an experienced reader.

Third pass: Reimplement. You should identify and challenge every assumption in every statement. Moreover, you should think about how you yourself would present a particular idea. This comparison of the actual with the virtual lends a sharp insight into the proof and presentation techniques in the paper and you can

very likely add this to your repertoire of tools. During this pass, you should also jot down ideas for future work.

TODO: http://www.cs.columbia.edu/hgs/etc/writing-style.html

TODO http://research.microsoft.com/enus/um/people/simonpj/papers/giving-a-talk/giving-atalk.htm

TODO http://www.ee.ucr.edu/rlake/Whitesides writing res paper.pdf

TODO http://greatresearch.org/2013/10/18/the-paper-reviewing-process/

Chapman [**?**]: "Many researchers spend more than half their time reading"

> The time to start reading is now. Once you start seriously working on your thesis youll have less time, and your reading will have to be more focused on the topic area. During your first two years, youll mostly be doing class work and getting up to speed on AI in general. For this it suffices to read textbooks and published journal articles.

"The reading lists for the AI qualifying exams at other universitiesparticularly Stanfordare also useful"

> There are three phases to reading. The first is to see if theres anything of interest in it at all. Second is to find the part of the paper that has the good stuff. Most fifteen page papers could profitably be rewritten as one-page papers; you need to look for the page that has the exciting stuff. Often this is hidden somewhere unlikely. What the author finds interesting about his work may not be interesting to you, and vice versa. Finally, you may go back and read the whole paper through if it seems worthwhile

> Read with a question in mind. How can I use this? Does this really do what the author claims? What if...? Understanding what result has been presented is not the same as understanding the paper. Most of the understanding is in figuring out the motivations, the choices the authors made (many of them implicit), whether the assumptions and formalizations are realistic, what directions the work suggests, the problems lying just over the horizon, the patterns of difficulty that keep

coming up in the authors research program, the political points the paper may be aimed at, and so forth.

"If you are interested in an area and read a few papers about it, try implementing toy versions of the programs being described. This gives you a more concrete understanding."

A.2.1 Reading a paper checklist

- How can I use this?
- Does this really do what the author claims?
- Are assumptions and formalizations realistic?
- Implement a toy version of the program

A.3 Getting connected

Chapman [**?**]: "its important to get plugged into the Secret Paper Passing Network. This informal organization is where all the action in AI really is."

This can be done by

- Mailing lists
- Talk about an idea youve had with someone who knows the field; they are likely to say, Have you read X?
- When you read a paper that excites you, send it to people who might be interested in it. They'll return the favor.
- Discussion groups
- Read papers on people's desks, filing cabinets
- Distribute draft copies of things you write ("Please do not photocopy or quote" on the front page). Most people dont read most of the papers theyre given, so dont take it personally

- When you finish a paper, send copies to everyone you think might be interested.
- Make yourself the bridge between two groups of interesting people working on related problems who arent talking to each other
- Keep a log of interesting references.
- Try talking to them about the really good or unbelievably foolish stuff youve been reading. Hang out, dinners, lunches, informal gatherings.
- Get a business card.
- At conferences, walk up to someone whose paper youve liked, say "I really liked your paper", and ask a question.
- Get summer jobs away at other labs.

"Its important to get a variety of people who will regularly review your work, because it's very easy to mislead yourself (and often your advisor as well) into thinking you are making progress when you are not, and so zoom off into outer space."

A.4 Learning other fields

- Take a graduate course. This is solidest, but is often not efficient.
- Read a textbook. Not a bad approach, but textbooks are usually out of date, and generally have a high ratio of words to content.
- Find out what the best journal in the field is, maybe by talking to someone who knows about it. Then skim the last few years worth and follow the reference trees. This is usually the fastest way to get a feel of what is happening, but can give you a somewhat warped view.
- Find out whos most famous in the field and read their books.
- Hang out with grad students in the field.
- Go to talks.

A.5 Keeping a Research Notebook

> Record in your notebook ideas as they come up. Nobody except you is going to read it, so you can be random. Put in speculations, current problems in your work, possible solutions. Work through possible solutions there. Summarize for future reference interesting things you read.

Methods?

- HTML Website. Pros: easily accessed from the web. Cons: Hard to update
- One big Latex file. Pros: easy to transfer to a paper. Cons: No hyperlinks. I'll use this.

Vera Johnson-Steiner's book Notebooks of the Mind: Describes how creative thought emerges from the accumulation of fragmented ideas.

Healy [**?**] on choosing your workflow applications:

> Rather than figuring out but not recording a solution to a problem you might have again, write down the answer as an explicit procedure. Instead of copying out some archival material without much context, file the source properly, or at least a precise reference to it.

His typefaces are nice. Myraid Pro, Pragmata Pro and Minion Pro.

A.6 Writing

A.6.1 General

> Writing down your ideas is the best way to debug them. Usually you will find that what seemed perfectly clear in your head is in fact an incoherent mess on paper.
>
> Realize that writing is a debugging process. Write something sloppy first and go back and fix it up. Starting sloppy gets the ideas out and

gets you into the flow. If you cant write text, write an outline. Make it more and more detailed until its easy to write the subsubsubsections. If you find it really hard to be sloppy, try turning the contrast knob on your terminal all the way down so you cant see what you are writing. Type whatever comes into your head, even if it seems like garbage. After youve got a lot of text out, turn the knob back up and edit what youve written into something sensible.

Perfectionism can also lead to endless repolishing of a perfectly adequate paper. This is a waste of time. (It can also be a way of semideliberately avoiding doing research.)

List of papers that are well-written:

- None so far!

List of poorly written papers/quotes:

- None so far!

General tips:

- Put the sexy stuff up front, at all levels of organization from paragraph up to the whole paper.
- explain why it works and why its interesting
- Do not imitate math texts; their standard of elegance is to say as little as possible, and so to make the readers job as hard as possible. This is not appropriate for AI.
- After you have written a paper, delete the first paragraph or the first few sentences. Youll probably find that they were content-free generalities, and that a much better introductory sentence can be found at the end of the first paragraph of the beginning of the second.

A.6.2 Goals for Getting Published

Standards are surprisngly low. Criteria:

- Has something new to say
- Isn't broken in some way

Strategies:

- Try a lot
- Make sure it's readable
- Circulate drafts beforehand
- Read backissues of conference and make sure style/content is appropriate
- Read the information for authors page specified by conference
- Read best paper awards

A.6.3 Abstract

Print this off and use this as a checklist whenever I write a paper. Stuff from Simon Peyton-Jones' talk [**?**].

- Four sentences:
- state the problem
- state why it's an interesting problem
- say what your solution achieves
- say what follows from your solution

A.6.4 Intro

- (1 page in conference)
- Describe the problem
- State your contributions (write the list of contributions fist)
- that's all
- use an example to introduce the problem
- Reader thinks"gosh, if they can really deliver this, that'd be exciting; Id better read on" [**?**]
- Contributions should be refutable
- No "the rest of the paper is". Instead, use forward references from the narrative in the introduction. The introduction (including the contributions) should survey the whole paper, and therefore forward reference every important part

A.6.5 The Problem

- 1 page

A.6.6 Related Work Purpose (1-2 pages)

- show my problem is unsolved
- show my problem is interesting
- show my idea works
- show how my idea compares to other ideas
- No related work yet! Include at end.
  - Problem 1: describing alternative approaches gets between the reader and your idea (I feel tired)

  - Problem 2: the reader knows nothing about the problem yet; so your (carefully trimmed) description of various technical tradeoffs is absolutely incomprehensible (I feel stupid)
  - instead, Concentrate single-mindedly on a narrative that
    * Describes the problem, and why it is interesting
    * Describes your idea
    * Defends your idea, showing how it solves the problem, and filling out the details
    * On the way, cite relevant work in passing, but defer discussion to the end

- show how my idea compares to other ideas
- Be generous to the competition. In his inspiring paper [Foo98] Foogle shows.... We develop his foundation in the following ways...
- Giving credit to others does not diminish the credit you get from your paper
- Acknowledge weaknesses in your approach
- Failing to give credit to others can kill your paper: You don't know that it's an old idea (bad) or You do know, but are pretending it's yours (very bad)
-

A.6.7 Method

- In a paper you MUST provide the details, but FIRST convey the idea
- Introduce the problem, and your idea, using EXAMPLES and only then present the general case
- Explain it as if you were speaking to someone using a whiteboard
- Conveying the intuition is primary, not secondary
- Once your reader has the intuition, she can follow the details (but not vice versa)

- Even if she skips the details, she still takes away something valuable

Evidence

- Your introduction makes claims; The body of the paper provides evidence to support each claim
- Check each claim in the introduction, identify the evidence, and forward-reference it from the claim
- Evidence can be: analysis and comparison, theorems, measurements, case studies

A.7 Giving Talks

"Since revising a talk is generally much easier than revising a paper, some people find that this is a good way to find the right way to express their ideas. (Mike Brady once remarked that all of his best papers started out as talks.)"

"Cornering one of your friends and trying to explain your most recent brainstorm to him is a good way both to improve your communication skills, and to debug your ideas."

A.8 Programming

> Like papers, programs can be over-polished. Rewriting code till its perfect, making everything maximally abstract, writing macros and libraries, and playing with operating system internals has sucked many people out their theses and out of the field.

A.9 How to Select a Graduate Advisor

- How much direction do you want?
- How much contact do you want?
- How much pressure do you want?

- How much emotional support do you want?
- How seriously do you want to take your advisor?
- What kind of research group does the advisor provide?
- Do you want to be working on a part of a larger project?
- Do you want cosupervision?
- Is the advisor willing to supervise a thesis on a topic outside his main area of research?
- Will the advisor fight the system for you?
- Is the advisor willing and able to promote your work at conferences and the like?
- Read the Labs research summary.
- Read recent papers of any faculty member whose work seems at all interesting.
- Talk to as many faculty members as you can during your first semester.
- Talk to grad students of prospective advisors and ask what working for him or her is like.
- Attend faculty member's research group meetings

A.10 How to Choose a Thesis Topic and Avoid Wasting Time

The essential requirement of a Masters thesis is that it literally demonstrate mastery: that you have fully understood the state of the art in your subfield and that you are capable of operating at that level. It is not a requirement that you extend the state of the art, nor that the Masters thesis be publishable.

The actual writing of the PhD thesis generally takes about a year, and an oft-confirmed rule of thumb is that it will drag on for a year after you are utterly sick of it.

Read other people's theses.
Choosing a topic:

- A good thesis topic will simultaneously express a personal vision and participate in a conversation with the literature.

- Your topic must be one you are passionate about. Nothing less will keep you going. Your personal vision is your reason for being a scientist, an image or principle or idea or goal you care deeply about. It can take many forms. Maybe you want to build a computer you can talk to. Maybe you want to save the world from stupid uses of computers. Maybe you want to demonstrate the unity of all things. Maybe you want to found colonies in space. A vision is always something big. Your thesis cant achieve your vision, but it can point the way.

- At the same time, science is a conversation. An awful lot of good people have done their best and theyre written about it. Theyve accomplished a great deal and theyve completely screwed up. Theyve had deep insights and theyve been unbelievably blind. Theyve been heros and cowards. And all of this at the same time. Your work will be manageable and comprehensible if it is framed as a conversation with these others. It has to speak to their problems and their questions, even if its to explain whats wrong with them. A thesis topic that doesnt participate in a conversation with the literature will be too big or too vague, or nobody will be able to understand it.

- The hardest part is figuring out how to cut your problem down to a solvable size while keeping it big enough to be interesting. Solving AI breadth-first is a common disease; youll find you need to continually narrow your topic. Choosing a topic is a gradual process, not a discrete event, and will continue up to the moment you declare the thesis finished. Actually solving the problem is often easy in comparison to figuring out what exactly it is.

If your vision is a fifty-year project, whats the logical ten-year subproject, and whats the logical one-year subproject of that? If your vision is a vast structure, whats the component that gets most tellingly to its heart, and what demonstration would get most tellingly to the heart of that component?

- An ideal thesis topic has a sort of telescoping organization. It has a central portion you are pretty sure you can finish and that you and your advisor agree will meet the degree requirements. It should have various extensions that are successively riskier and that will make the thesis more exciting if they pan out. Not every topic will fit this criterion, but its worth trying for.

- Some people find that working on several potential thesis projects at once allows them to finish the one that works out and abandon the ones that fail.

- Topics can be placed in a spectrum from flakey to cut-and-dried. Flakier theses open up new territory, explore previously unresearched phenomena, or suggest heuristic solutions to problems that are known to be very hard or are hard to characterize. Cut-and-dried theses rigorously solve wellcharacterized problems. Both are valuable; where you situate yourself in this spectrum is a matter of personal style.

- The further work sections of papers are good sources of thesis topics.

be able to explain simply how each part of your theory and implementation is in service of the goal.

Make sure once youve selected a topic that you get a clear understanding with your advisor as to what will constitute completion.

Try a simplified version of the thesis problem first. Work examples. Thoroughly explore some concrete instances before making an abstract theory

There are a number ways you can waste a lot of time during the thesis. Some activities to avoid (unless they are central to the thesis): language design, userinterface or graphics hacking, inventing new formalisms, overoptimizing code, tool building, bureaucracy. Any work that is not central to your thesis should be minimized.

A.11 How to do Resarch Methodology

A.12 Emotional Factors in the Research Process

The entire section by Chapman [**?**] is worth a read.

Biograpahies: Gregory Batesons Advice to a Young Scientist, Freeman Dysons Disturbing the Universe, Richard Feynmanns Surely You Are Joking, Mr. Feynmann!, George Hardys A Mathematicians Apology, and Jim Watsons The Double Helix

Appendix B

Daily Log

B.1 2014 December

To do:

- RPE intial draft
- Giraffe paper camera-ready copy
- add stuff from one note, webisite

Done:

- none!

Doing experiment **??**.

Appendix C

Appendix

This would be any supporting material not central to the dissertation. For example:

- additional details of methodology and/or data;
- diagrams of specialized equipment developed.;
- copies of questionnaires and survey instruments.

Appendix D

Career

D.1 Questions for Job Fairs

As a busy student, how do you keep up to date with technology developments in industry? What was the last technology that you heard of and thought 'that's really cool'? 'uncool'? Suppose you're locked in a building and can't leave. How many ways can you think of to measure the exterior temperature? Resume related questions: Fairtunes, C&O, Waterloo. How can you prevent deadlock in a concurrent system? Write a function to add 64 bit numbers using 32bit arithmetic. What is a thread? What is a process? Write a function that, given an alphabetic string, outputs the characters a) sorted b) duplicates removed and c) in lower case. Do it using only a few (¡26) bytes of storage. (eg: "Mississippi" -¿ "imps") Insert into a circular doubly linked list. A square island, with side length n meters, lies in the center of a square pond, with side length n+2 meters. Given only two boards of length 0.99 meters, how can you reach the island from the mainland? ASCII strings are typically encoded using 1 byte per character, even though for alphanumeric text, only the first 7 bits are used (the MSB is always 0). Write a function that compresses a c string by outputting a string of bytes where each group of seven bits represents each character of input. For example, the 3 byte string 01111111, 01011100, 00000101 would compress to the 21 bit string 111111110111000000101. itoa() (He first suggested "reverse the word ordering in a string", then malloc()/free()) Tell me about a time your beliefs were challenged in the workplace. Tell me about a time when

you discovered a fundamental error in the design of your project. Design and implement the ”shuffle” function for a 200 disc CD changer. Your only constraint is that no song may be repeated until every other song has been played once.

D.2 Questions for Interviews

D.3 Job Fair

Look professional, smile, clear and confident-sounding voice, a firm handshake and good eye contact. Wear my blue shirt, perhaps get it tailored.

Introduction Hi, I'm Victor. I'm a second year Master's student in Computer Science doing research in Robotics and Computer Vision. I'll be looking for full-time positions starting in September. Does [company] have any opportunities in software development?

Questions to learn about company culture, organization:

- What's pretty hot to know nowadays in Microsoft? What programming languages?
- Are there code reviews?
- On your 'About us' section of your webpage you mention X.
- I printed out a copy of a job description on your website. What do you mean by 'you need X?'
- How do you like your team? Can new hires meet potential teams?

Follow up email Hi John, it was greating speaking with you at the UBC Technical Career Fair last Wednesday. Thanks for telling me about the machine learning positions at Microsoft. As mentioned, I'll be looking for full-time positions starting in September and the positions you've describe seem like a good fit for me. Here's a copy of the resume I gave you. Feel free to reach me at 2369997236 or this email address, vhg@cs.ubc.ca.

D.4 Misc Resources

https://www.cs.ubc.ca/students/undergrad/careers/finding/preparing-technical-career-fair-tips

http://www.reddit.com/r/cscareerquestions/comments/20ahfq/heres_a_pretty_big_list_of_programming_interview/:

> You probably could get away with a single page cheatsheet but beyond that you probably will be spending more time looking things up and that will count against you. The questions are also very academic in nature but still application oriented so you need to know when, where, and how to use all the different data structures and quickly adapt them to your needs in the question. They will then ask you about performance and a bunch of other things(at least in my interviews) so in order to have a complete cheat sheet you need to have or remember everything you probably learned in your first 2 years of CS classes feeding on your last 2 years of classes. The recruiter I talked to said people at Google said that the phone interview and the process in general was harder than the hardest test they had in school but very similar to a final exam for one of their classes.

Be well prepared, study hard. Buy a few good books Cracking the Coding Interview Programming Interviews Exposed How Would You Move Mount Fuji? Talk to friends. Aggressively pursue interview opportunities. They rarely come to you. Make sure you can perform well with little to no sleep. Know what they want. Then show you have that. Enjoy the game. Contact HR if you havent heard from them (seeJC Oct 6). Be enthusiastic. No two interview experiences are alike. Prepare for anything and everything. Research shows that the interviewer makes up their mind in the first thirty seconds that they meet you. Dont have a cover letter. Be honest with yourself and your interviewers.

D.5 Interview Questions

D.5.1 How to Answer Interview Questions

As an interviewer, I would expect you to know proper syntax for common things in whatever language you work in (presumably Java), as well as very common API (substring, working with collections, etc). If you can't write a for loop, then that would be problematic. Don't know the try-with-resources syntax? That's OK. If you don't know that substring exists, problem. If you use "put" instead of "add" (or vice versa) for a collection - not a big deal, that's what IDEs are for. And even if you mess up on something basic, if you still did fine on the rest of it, I'll overlook that.

Algorithm:

1. Ask clarifying questions
2. Give an example and verify: simple example
3. Give an example and verify: corner case
4. Think about possible solutions and complexity
5. Lower bound complexity
6. Start writing
7. Walk through code
8. Test with: Simple example (walk through code exactly)
9. Test with: b. Corner cases

D.5.2 General

- Find the most frequent integer in an array
- Find pairs in an integer array whose sum is equal to 10 (bonus: do it in linear time)
- Given 2 integer arrays, determine of the 2nd array is a rotated version of the 1st array. Ex. Original Array A=1,2,3,5,6,7,8 Rotated Array B=5,6,7,8,1,2,3

- Write fibbonaci iteratively and recursively (bonus: use dynamic programming)
- Find the only element in an array that only occurs once.
- Find the common elements of 2 int arrays
- Implement binary search of a sorted array of integers
- Implement binary search in a rotated array (ex. 5,6,7,8,1,2,3)
- Use dynamic programming to find the first X prime numbers
- Write a function that prints out the binary form of an int
- Implement parseInt
- Implement squareroot function
- Implement an exponent function (bonus: now try in log(n) time)
- Write a multiply function that multiples 2 integers without using *
- HARD: Given a function rand5() that returns a random int between 0 and 5, implement rand7()
- HARD: Given a 2D array of 1s and 0s, count the number of "islands of 1s" (e.g. groups of connecting 1s)

D.5.3 Strings

- Find the first non-repeated character in a String
- Reverse a String iteratively and recursively
- Determine if 2 Strings are anagrams
- Check if String is a palindrome
- Check if a String is composed of all unique characters

- Determine if a String is an int or a double
- HARD: Find the shortest palindrome in a String
- HARD: Print all permutations of a String
- HARD: Given a single-line text String and a maximum width value, write the function 'String justify(String text, int maxWidth)' that formats the input text using full-justification, i.e., extra spaces on each line are equally distributed between the words; the first word on each line is flushed left and the last word on each line is flushed right

D.5.4 Trees

- Implement a BST with insert and delete functions
- Print a tree using BFS and DFS
- Write a function that determines if a tree is a BST
- Find the smallest element in a BST
- Find the 2nd largest number in a BST
- Given a binary tree which is a sum tree (child nodes add to parent), write an algorithm to determine whether the tree is a valid sum tree
- Find the distance between 2 nodes in a BST and a normal binary tree
- Print the coordinates of every node in a binary tree, where root is 0,0
- Print a tree by levels
- Given a binary tree which is a sum tree, write an algorithm to determine whether the tree is a valid sum tree
- Given a tree, verify that it contains a subtree.
- HARD: Find the max distance between 2 nodes in a BST.
- HARD: Construct a BST given the pre-order and in-order traversal Strings

D.5.5 Stacks, Queues, and Heaps

- Implement a stack with push and pop functions
- Implement a queue with queue and dequeue functions
- Find the minimum element in a stack in O(1) time
- Write a function that sorts a stack (bonus: sort the stack in place without extra memory)
- Implement a binary min heap. Turn it into a binary max heap
- HARD: Implement a queue using 2 stacks

D.5.6 Linked Lists

- Implement a linked list (with insert and delete functions)
- Find the Nth element in a linked list
- Remove the Nth element of a linked list
- Check if a linked list has cycles
- Given a circular linked list, find the node at the beginning of the loop. Example: A–¿B–¿C –¿ D–¿E –¿C, C is the node that begins the loop
- Check whether a link list is a palindrome
- Reverse a linked list iteratively and recursively

D.5.7 Sorting

- Implement bubble sort
- Implement selection sort
- Implement insertion sort
- Implement merge sort
- Implement quick sort

D.5.8 Dynamic Programming

From http://en.wikipedia.org/wiki/Dynamic_programming#Algorithms_that_use_dynamic_programming:

- Recurrent solutions tolattice modelsfor protein-DNA binding
- Backward inductionas a solution method for finite-horizondiscrete-timedynamic optimization problems
- Method of undetermined coefficientscan be used to solve theBellman equationin infinite-horizon, discrete-time,discounted,time-invariantdynamic optimization problems
- Manystringalgorithms includinglongest common subsequence,longest increasing subsequence,longest common substring,Levenshtein distance(edit distance)
- Many algorithmic problems ongraphscan be solved efficiently for graphs of boundedtreewidthor boundedclique-widthby using dynamic programming on atree decompositionof the graph.
- TheCockeYoungerKasami (CYK) algorithmwhich determines whether and how a given string can be generated by a givencontext-free grammar
- Knuth's word wrapping algorithmthat minimizes raggedness when word wrapping text
- The use oftransposition tablesandrefutation tablesincomputer chess
- TheViterbi algorithm(used forhidden Markov models)
- TheEarley algorithm(a type ofchart parser)
- TheNeedlemanWunschand other algorithms used inbioinformatics, includingsequence alignment,structural alignment,RNA structure prediction
- Floyd's all-pairs shortest path algorithm
- Optimizing the order forchain matrix multiplication

- Pseudo-polynomial timealgorithms for thesubset sumandknapsackandpartitionproblems
- Thedynamic time warpingalgorithm for computing the global distance between two time series
- TheSelinger(a.k.a.System R) algorithm for relational database query optimization
- De Boor algorithmfor evaluating B-spline curves
- DuckworthLewis methodfor resolving the problem when games of cricket are interrupted
- The value iteration method for solvingMarkov decision processes
- Some graphic image edge following selection methods such as the ”magnet” selection tool inPhotoshop
- Some methods for solvinginterval schedulingproblems
- Some methods for solvingword wrapproblems
- Some methods for solving thetravelling salesman problem, either exactly (inexponential time) or approximately (e.g. via thebitonic tour)
- Recursive least squaresmethod
- Beattracking inmusic information retrieval
- Adaptive-critic training strategy forartificial neural networks
- Stereo algorithms for solving thecorrespondence problemused in stereo vision
- Seam carving(content aware image resizing)
- TheBellmanFord algorithmfor finding the shortest distance in a graph
- Some approximate solution methods for thelinear search problem
- Kadane’s algorithm for themaximum subarray problem