8. Modernization Efforts

Cleaning up the code and adding newer technology

Ray Smith, Google Inc.

Code Cleanup

- Conversion to C++ completed
- Thread-compatibility: can run multiple instances in separate threads
- Multi-language (single and mixed) capability:
  copes with jpn, heb, hin, and mixes such as hin+eng
- Removed many hard-coded limits, eg on character set, dawgs
- New beam search
- ResultIterator for accessing the internal data
- More generic classifier API for plug-n-play experiments
- Removed lots of dead code, including IMAGE class.

OpenCL activity (Partnership with AMD)

AMD

Motivation:

- Accelerated Processing Units (APUs) and integrated GPUs are now common place across Desktop, Mobile and Server platforms. GPUs have ~10x compute capability compared to CPUs
- OpenCL is a platform independent open-source parallel programming language that exploits GPU capabilities
- Tesseract processing pipeline consists of several parallel-friendly steps mainly in image preprocessing & Character / Word recognition
- Hence Tesseract OCR processing has been accelerated using OpenCL

Open CL Progress (Data provided by AMD team)

Speed-up so far 36%, with the class pruner still to do!

AMD

Times shown in seconds on 6 page color business letter on AMD A10-7850K Kaveri machine with OpenCL1.2

ClassPruner, feature extraction, MasterMatcher still to do. OpenCL2.0 coming.

Module	Native CPU	OpenCL	Speed-up ratio
Tiff Image Read + color mapping	3.837	2.546	1.507
RGB to gray	5.157	0.691	7.463
Histogram	4.423	0.384	11.518
Thresholding	5.474	0.268	20.425
Morphology (layout analysis)	4.950	0.323	15.325
Master Matcher (knn classifier)	25.954	9.249	2.806
Other (work in progress)	74.209	76.626	0.968
Total	124.004	91.168	1.360

Plug-n-Play Classifier API (Highlights)

```
class ShapeClassifier {
  ...
  virtual int UnicharClassifySample(const TrainingSample& sample, Pix* page_pix,
                                    int debug, UNICHAR_ID keep_this,
                                    GenericVector<UnicharRating>* results);

  virtual int ClassifySample(const TrainingSample& sample, Pix* page_pix,
                             int debug, UNICHAR_ID keep_this,
                             GenericVector<ShapeRating>* results);
```

- Either `UnicharClassifySample` or `ClassifySample` may be implemented.
- Difference is whether to return a `UNICHAR_ID` or indirect through a `Shapetable`.

Input data for Classifier

```
class TrainingSample : public ELIST_LINK {

  Pix* GetSamplePix(int padding, Pix* page_pix) const;
  UNICHAR_ID class_id() const;
  int font_id() const;
  const INT_FEATURE_STRUCT* features() const;
  const MicroFeature* micro_features() const;
```

Provides access to everything from an image of the character through to the current tesseract features.

Algorithmic Modernization

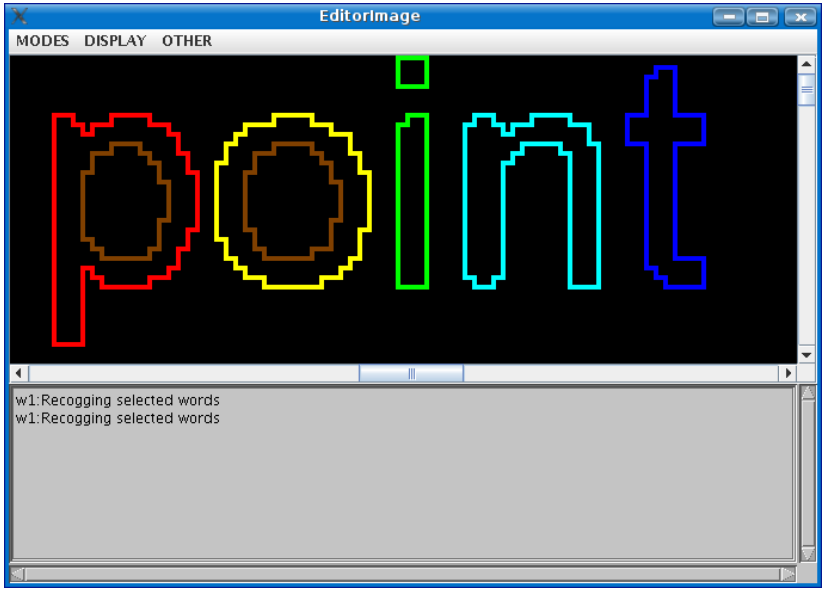
- Cube added Convolutional NN, but improvement was disappointing.
- It would be nice to eliminate the polygonal approximation…
- It would be nice to eliminate the need for accurate baseline normalization
  => New classifier experiments

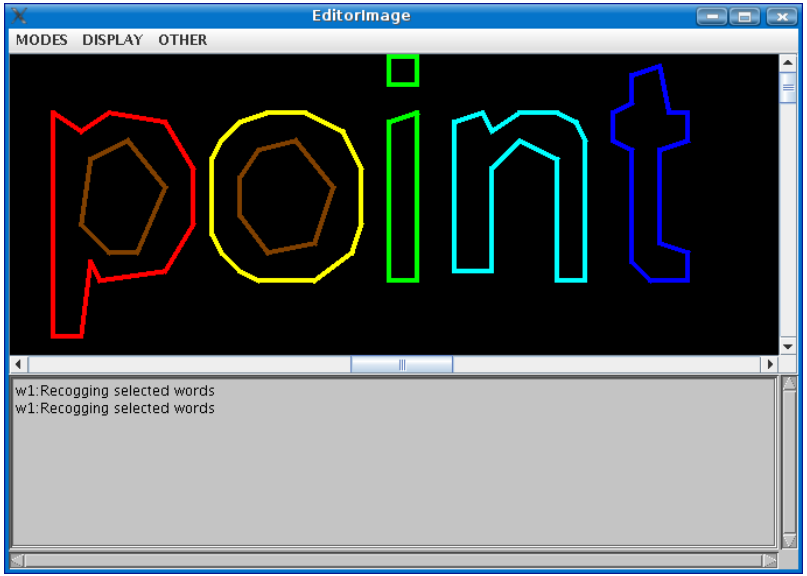
Cube

- Standard Convolutional Neural Network character classifier.
- Uses over-segmentation and (a different) beam search for character segmentation.
- Implemented as an alternative word recognizer.
- Achieves about 50% reduction in errors on Hindi when run together with Tesseract's word recognizer.
- Improvement for other languages disappointing: few % for major slow-down (3x).
- Training code unmaintained, unused.

Eliminate the Polygonal Approximation?

Pixel edge step outlines -> Polygonal approximation





Challenge: Eliminate the Polygonal Approximation!

- Allow feature extraction from greyscale by eliminating the dependence on the polygonal approximation.
- Make it faster and more accurate for CJK, Indic, and OSD.
- Keep everything else as constant as possible:
  - Same feature definition
  - Same segmentation search/word recognizer
  - Same training data, but add scope for significantly more fonts.

Idea 1: Modernize. Apply boosting to Class Pruner

Result:

- Improves accuracy on Training data.
- Generalization hard to achieve.
- Spreading difficult to incorporate into boosting framework.

Idea 2: Pure Logic is Superior to Emotion, Captain

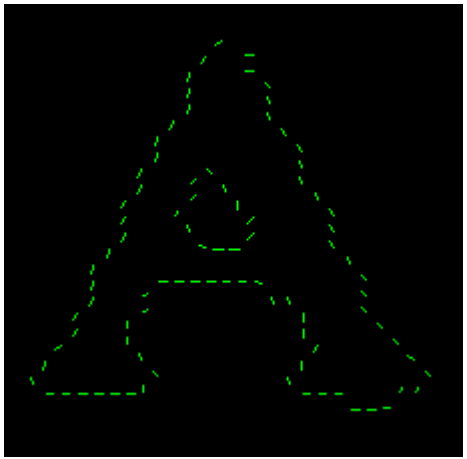
Q: If Tesseract is a kNN classifier, why doesn't it achieve 100% accuracy on its training data?
A: It clusters its training samples (by font) and uses the cluster means only => any individual sample may not be nearest to the correct cluster center.

Q: How can we achieve 100% accuracy on the training data?
A: Pure Logic. OR together all the training samples in the class pruner.

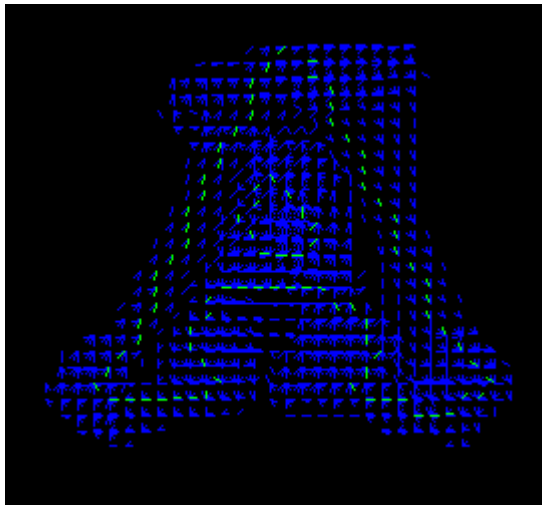
Q: Doesn't that make it accept anything as anything?
A: Not with proper shape clustering.

Binary Pruner in Action

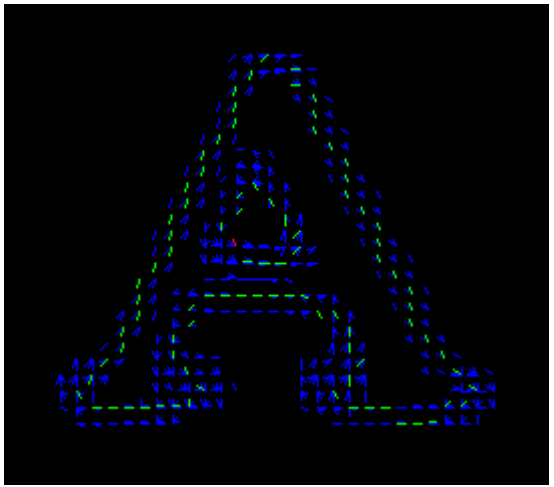
Quantized Features



Matched against multiple fonts



Matched against a single font



How about Generalization?

- Early experiments showed the BinaryPruner less accurate on unseen fonts when trained on 200 fonts than the ClassPruner trained on 32 fonts.
- Adding a 'halo' classifier working on the near neighbors of the training samples fixed the problem - at a cost:

Character subst errs	-2.5% (UNLV)
Flat word drops	-0.15%
Bulk drops	+2.3%
CPU	+110%

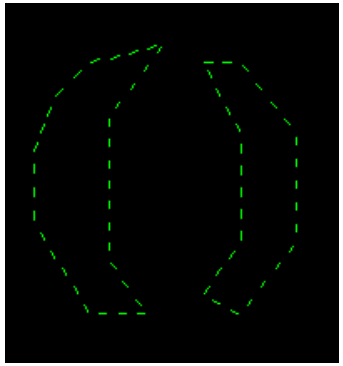
Non-Obvious observation:

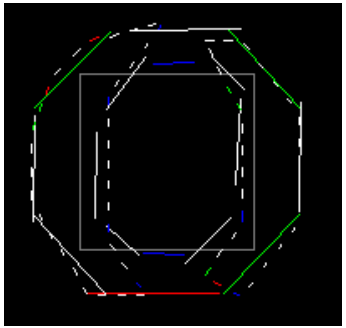
Despite being designed over 20 years ago, the current Tesseract classifier is incredibly difficult to beat with so-called modern methods. (Without cheating by changing features or upping the number of training fonts)

Why?

Statistical Dependence Strikes Again

- Small features to large proto matching accounts well for large-scale features
- Binary quantization of feature space is a loser
- HMMs, DBNs and LSTMs have a better chance than other methods





LSTM Integration

- LSTM code based on OCROpus Python implementation.
- Expanded capabilities including 2-D, and input of Tesseract features, as well as images.
- Fully integrated with Tesseract at the line level. (May change)
- Visualization with existing Viewer API.
- Training code included (unlike cube).
- Parallelized with openMP.
- Likely will get OpenCL implementation too.
- Coming in next release of Tesseract.

Thanks for Listening!

Questions?