EE292B Project Guidelines and Suggestions

Handed out: April 5, 2005

Note: this document is subject to change until the handout date.

1 Proposals

Project proposals should be handed in by **April 28**. For team projects, a single proposal may be handed in. The proposal should be between about 500 and 1500 words and should identify the problem being addressed, describe the goals of the project, and list who will work on it. A proposal for a team project should sketch the planned division of work.

2 Project Reports

The project report is due in class on **June 2**. A single report may be handed in for team projects, prefaced by a page indicating clearly who did what. The length of the rest of the report should be approximately the same as for an IEEE ICASSP or ICIP conference paper — about 3000 words.

3 Project Expectations

Projects can be undertaken individually or in teams of up to three. The grade on the project will constitute 40% of the grade in the class; it should represent a commensurate amount of effort on the part of each project member.

Ideally, a project should strive to be of the quality and scope of an IEEE conference contribution (e.g., ICIP or ICASSP), with a difference being that the project need not be entirely novel. For instance, your project might consist of re-implementing and testing a previously published method. Also, writing a survey or review paper is a possible alternative to a computer project. In addition to clearly describing what was done and discussing / analyzing the results, the writeup should demonstrate:

a) A clear understanding of the problem being addressed;

b) Awareness of major relevant prior work;

c) Critical analysis of prior work (especially if a review/survey paper); and

d) Analysis or reasoning behind the proposed approach.

4 Suggestions for Project Topics

You are free to propose any topic you wish related to general theme of the course; the following are some possibilities. This list will grow until this document is handed out in class.

a) *VQ-based inverse halftoning.* This topic would explore in depth the use of vector quantization (VQ) to perform inverse halftoning, as proposed by Ting and Riskin (see the handout on inverse halftoning). Ting and Riskin applied the technique to error-diffusion halftones using a neighborhood size for which all of the halftone patterns could be stored exactly. This project could explore its application to ordered-dither halftones, and could consider larger neighborhoods. Several high-resolution grayscale and binary scans of halftones are available as test/training images.

b) *Resolution enhancement.* Problem Set 2 considered changing the size of an image by linear filtering. Other techniques that take advantage of knowledge of the specific type of image can be more effective, especially when the image is being enlarged. In that case, spatial detail — information not present in the input image — must be "invented" based on prior assumptions about the class of images to which the input is assumed to belong. A project along these lines might look at applying new texture synthesis methods as recently described by Li-Yi Wei (http://graphics.stanford.edu/projects/texture/) and Alyosha Efros (http://www.cs.berkeley.edu/ efros/research/synthesis.html) to perform resolution enhancement on text images.

c) *Page segmentation.* Investigate one or more methods of classifying regions of a scanned or rasterized document image as halftone, line drawing, or text. There is a vast literature on this subject; one family of viable approaches, based on morphology, was presented by Dan Bloomberg in lecture on October 10. This approach will be explored in a problem on Problem Set 4. The idea of a project here would be to explore alternative ways of using these or other methods.

d) *High-fidelity compression of high-resolution grayscale scans.* While noise removal, deskewing, and binarization are usually essential for making scanned documents maximally useful, these processes lose some information and may not be carried out in a way that anticipates all future uses of the scanned documents. It therefore can be desirable to archive the scanned document in its raw form, just after scanning in grayscale (or color) and at high resolution. Lossless compression is usually not very effective because the least-significant bits tend to be noisy. However, a

near-lossless technique, perhaps with some guarantee of fidelity, might compress extremely well on average. The JPEG-LS standard attempts to do this for images in general, but it is not necessarily the best choice for scanned document images. This project would explore near-lossless techniques that are specifically optimized for grayscale scanned document images.

e) *Switched-model compression.* In the mixed-raster content (MRC) approach to document image compression, a page image is first segmented into regions of different types. Each region is then compressed using a separate technique appropriate for its type, and the relevant segmentation information in the form of a mask is also compressed. An alternative would be to use a single general-purpose compression technique, such as vector quantization, that can be tuned to a variety of region types by adjusting its model or codebook. A page image could then be compressed using this general-purpose compressor, being careful to switch the model or codebook from region to region. This project would review the literature on this type of coding, then propose and carry out computer experiments in this type of compression.

f) *Document authentication.* How do you verify that a paper version of a document has not been significantly altered, assuming you have the original? Suggestion: Use the JBIG2 compression image processing software supplied in Problem set 3 (appropriately modified) to verify that each image component is sufficiently similar to the one in the original. You will need to register the scanned (paper) version, e.g., using a 3-point affine transform, with the electronic original, in order to know which components to compare.

g) *JBIG2 for OCR* Build a character recognition ("OCR") program, using only a JBIG2 classifier and a dictionary to resolve the character codes from the character classes. Demonstrate this works on an image that is built from an original scanned image and the JBIG2 compression files, but to which you have applied a global character substitution table (e.g., each character in class 4 is substituted by the template from class 49, etc.)

h) *Word Template Dictionaries.* Apply the JBIG2 compression (appropriately modified) to documents on a word basis, resulting in a set of word equivalence classes. From these classes alone, find both the most common and the most significant words in a document.

i) *Estimation and restoration.* Use the JBIG2 encoder supplied in Problem Set 3 to make an improved version of a scanned binary image. Start with 300 ppi binary. For each component class, generate both a binary version and a grayscale version of the character, using all available instances. Substitute these improved templates in the original. Then use the improved grayscale templates to generate $2\times$, $4\times$ and $8\times$ magnified versions of the

original. (The 4× and 8× versions will be very large; just use part of the image.)

j) *Mixed raster content segmenter.* Build a foreground/background segmenter for a mixed raster content document compression system, using either the top-down hierarchical k-means clustering method for projecting foreground and background cluster center values down to individual pixels, as described in class, or using another method of your choice. Test your segmenter on a variety of scanned pages with mixed text and images.

k) *Segmentation for text.* Given some complicated color layout pages (e.g., catalog advertising) with text both inside and outside the images, develop a program that extracts all the text as a binary image, that can be input to an OCR engine. Compare the results of your text extractor with the generic one built into the OCR engine.

l) *Music interpretation.* Write a program that extracts a musical score from the printed paper. You might want to do it first on just a single part, and if that is successful go on to a piano score.

After capture and analysis of the content, in order to debug your program, it is useful to output it in an auditory and/or visual form. You might do either of the following:

- convert the result to midi and play it back
- re-typeset the result and print it out.

The first of these is easier, and it will allow you to easily transpose the piece to a new key. There are commercial programs that print midi. If you want the additional (and time-consuming) challenge of writing the low-level print driver yourself, you have 2 choices for output:

- directly composing the midi into a bitmap
- indrectly composing the output in PostScript

m) *Mosaic generator.* Write a program that takes a number of overlapping pictures (taken with a hand-held digital camera) and forms a mosaic of them, without visible stitching. Take the pictures quickly and at the same fixed focus so that you don't have problems with changing cloud patterns or scale. (It is very hard if you have to scale each image separately.)