

SmartNails - Display and Image Dependent Thumbnails

Kathrin Berkner^a and Edward L. Schwartz^a and Christophe Marle^b

^aRicoh Innovations, Inc., 2882 Sand Hill Road, Suite 115, Menlo Park, CA 94025

^bMicrosoft, One Microsoft Way, Redmond, WA 98052.

ABSTRACT

In order to overcome poor readability of text and recognizability of image features in low resolution thumbnails, a novel image representation of compound document images - a SmartNail representation - is presented. SmartNails are replacements or supplements to traditional thumbnails for compound documents and contain cropped and scaled image and text segments. Image-based analysis and text-based analysis are merged to generate a layout for a particular display size with selected readable text and recognizable image regions. The analysis is efficiently performed by using information from document layout analysis and JPEG 2000 compressed file headers.

Keywords: compound document, reformatting, thumbnail, automatic scaling and cropping, JPEG 2000.

1. INTRODUCTION

The variety of displays used to browse and view images has created a need to adapt an image representation to the size constraints of a given display. For example, a high-resolution (print resolution) image might need to be displayed on a desktop monitor, on a PDA, or as a thumbnail in a desktop application. The conventional thumbnail representation obtained by uniformly scaling the whole image to fit small-sizes displays is often unsatisfactory, especially for document images. Those are likely to loose readability of the text when rescaled to fit on a cell-phone.

For example, thumbnails has been shown in the literature and in use in commercial products to contain very valuable information. For document images typically OCR results are obtained, keywords extracted and displayed in combination with a document thumbnail. However, OCR results are not always perfect (e.g. some fonts are difficult to be recognized), keyword listings do not provide any association with the spatial context of the keyword in the document, and no information is extracted from non-text areas.

Several approaches have been proposed to improve representations of documents on a smaller display or in thumbnails. Besides color-coding keywords⁷, another keyword-centric approach is to create a more informative representation by pasting augmented keywords, that were obtained from OCR results, onto a scaled version of the image¹¹ maintaining the original location of the words in the document as best as possible. This approach has been applied in¹¹ to web documents. However, keywords are not available for image regions in compound document images. Furthermore, OCR errors may degrade this approach's representation. In³ the vulnerability to OCR errors for reformatting a scanned document to a width-constrained PDA display is overcome by using only the document layout analysis information to reflow text given its bounding boxes and following the reading order. A text-centered method for summarization of web documents for handheld devices is presented in¹.

Besides the text-centered approaches, there has also been work done from an image-point of view in order to adapt images for display on small devices. An approach of changing image content in document for a small target display is presented in¹² as re-coding of images for constrained displays with the goal of fast transmission over bandlimited channels. Automated cropping of images to fit a given display size is introduced in^{6,8}. However, those approaches do not incorporate any text specific characteristics of document images.

The third author performed the work as a summer intern at Ricoh Innovations, Inc.
Email: {berkner, schwartz}@rii.ricoh.com, cmarle@microsoft.com

In this paper a method for automatically reformatting a scanned document containing text and image regions for display on a width- and height-constrained display is presented. Three operations — scaling, cropping, and rearranging the original document layout in the final display — are allowed to produce a pictorial document representation — a *SmartNail*TM — for a given display. The goal is to include most important text regions at readable resolution and image parts at appropriate resolutions, while keeping the overall thumbnail layout close to the original document layout. The main steps for creation of a SmartNail consist of an image-based resolution-sensitive segmentation using JPEG 2000 header data (Section 2), a text-based segmentation using document layout analysis information (Section 3), a process of merging image- with text-based segmentation results (Section 4), and a resolution-sensitive layout step (Section 5) for final composition resulting in a list of crop, scale, and paste instructions. Executing those instructions on the JPEG 2000 compressed file allows fast extraction of selected data. Experiments (Section 6) and conclusions (Section 7) are presented.

2. AN IMAGE-POINT-OF-VIEW: ANALYSIS BASED ON JPEG2000 HEADER DATA

Creating a thumbnail from an image-analysis point of view has the goal of identifying important segments of an image and determining how much these segments can be reduced in size by scaling without losing the characteristics that made them important. Key to this is to use a practical, easily computed measure of importance. Lossy image compression is based on the goal of allocating bits in a compressed data file to the important information in an image. A bit distribution from the compressed data of a lossy image compressor can be used as an importance measure if it meets the following three requirements: the codestreams have (i) spatial locality, (ii) resolution/frequency locality, and (iii) the compressor is well designed for images, i.e. makes reasonable choices about importance information to allocate bits to.

JPEG 2000⁴ is a newly standardized wavelet-based compression scheme for digital images that meets these requirements. The JPEG 2000 image compression algorithm consists of the following steps:

- 1) Optionally divide the image into independent rectangular tiles.
- 2) Take the wavelet transform of each tile.
- 3) Partition the wavelet domain at each level and sub-band to form local groups of typically 32 x 32 or 64 x 64 coefficients termed *code-blocks*.
- 4) Encode each code-block independently using an arithmetic coder.
- 5) Organize the coded coefficients into one or more *layers* to facilitate progression.
- 6) Assemble all the coded and organized data into smaller units called *packets*.
- 7) Store the organizational information required to parse the coded data in *packet headers*.

The packet headers contain information such as number of bits allocated to each code-block. At low bit-rate, this information reflects the visual importance of a code-block. Due to the specific encoding scheme in the wavelet domain a code-block is localized in the spatial and frequency domain. In⁶ a *multiresolution bit distribution* $B = B(r, (i, j))$ is derived from the packet headers. For each code-block location (i, j) at resolution r the value $B(r, (i, j))$ denotes the number of bits spent by the encoder to encode the specific code-block area at resolution r . In⁶ the distribution B was used to perform automatic cropping of images to a given fixed size. The cropping locations and dimensions were obtained by analyzing the distribution B . For the resolution-sensitive segmentation task of this paper a Bayesian-type segmentation technique is applied to the bit distribution that yields a segmentation map grouping pixel blocks with similar preferred resolution. In detail, each code-block location (i, j) is labeled with its preferred resolution r^* , where r^* is computed as

$$r^*(i, j) = \arg \min_r \sum_{l=r}^L w_{r,l} \cdot B(l, (i, j)), \quad (1)$$

where L denotes the maximal level of wavelet decomposition and $w_{r,l}$ are specifically chosen weights. Figure 1 shows a multiresolution bit distribution for $L = 5$ and a multiresolution-dependent segmentation map as the result of the segmentation process. Segmentation map values visualized with the same shade of gray have the same optimal resolution r^* as defined in Eq.(1).

Since only the J2K header information is used, the resolution-dependent segmentation map is computed without any decoding of wavelet coefficients.

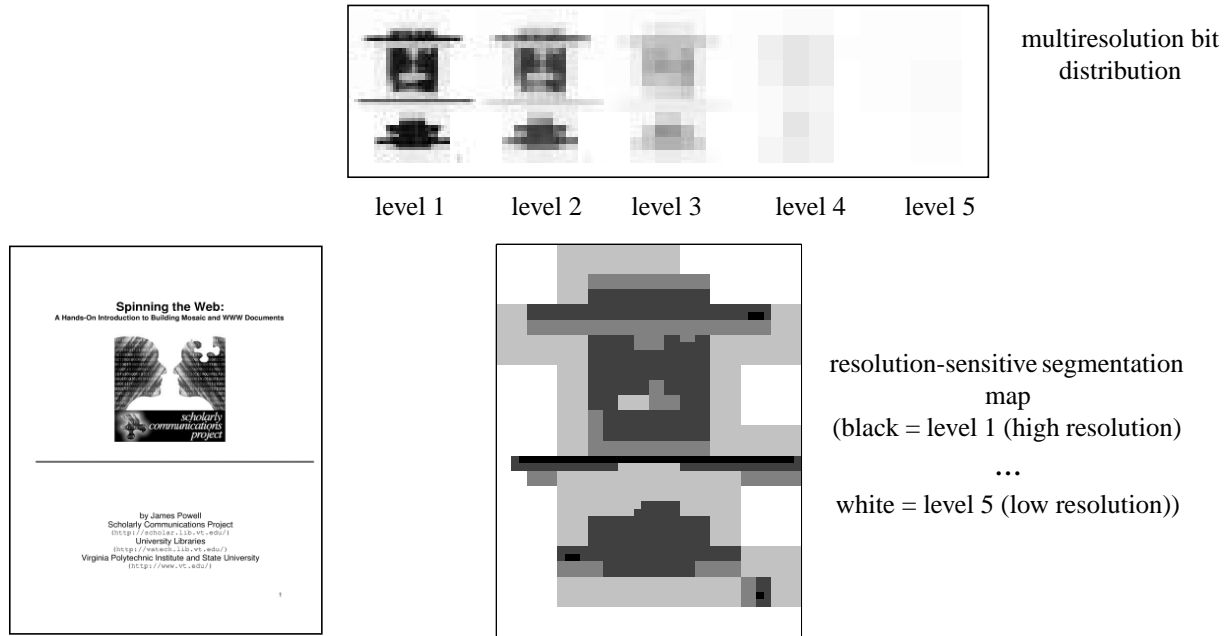


Figure 1 — Original document (left) and resolution-sensitive segmentation map (right) derived from multiresolution bit distribution (top).

Next a connected component analysis is performed on the segmentation map. For each resolution, the segmented areas are divided into connected components. As a result, each component has a resolution r^* assigned to it, i.e. a *resolution attribute*. An *importance attribute* I is also assigned to each component S and is defined as the number of bits necessary to code component area at resolution determined by its resolution attribute: $I(S) = \sum_{(i, j) \in S} \sum_{r=r^*(S)}^L B(r, (i, j))$. For the remainder of this paper, the components are called *segments*.

The segments provide a coarse segmentation of a document into regions with different dominant frequencies. Due to the coarse code-block resolution and no specific text knowledge in the JPEG 2000 encoder, the segments may not completely capture text zones or align with them. On the other hand, image regions are well characterized.

3. A TEXT-POINT-OF-VIEW: ANALYSIS BASED ON OCR LAYOUT ANALYSIS

From a text point of view, displaying text at a readable resolution is very important. A goal for automatic thumbnail creation is to include as much important text as possible while keeping the text readable.

Initially, a document layout analysis is performed. This may be specifically designed for thumbnail creation or may be shared with OCR used for other purposes such as keyword search. The bounding boxes for text zones and words are obtained from the layout analysis. This information provides a segmentation map for the text portions of a document including a hierarchy between segments and sub-segments. In this paper a text zone is called a *segment*; a sub-segment consists of individual words called *fragments*. A text segment is described by a rectangular bounding box. The aspect ratio of a segment can be changed by reflowing the fragments (i.e. the words).

Next, a resolution and an importance attribute is assigned to each segment. A resolution attribute is derived from the minimal scaling factor that still keeps the text at a readable character size. The minimal readable character size is determined empirically for a given display device type, e.g. CRT monitor or LCD screen. In the experiments in this paper, a 72 dpi CRT monitor was used and a minimal readable character size of 6 pixels was determined. For each segment a *resolution attribute* is defined as the ratio of the *Minimal Readable Character Size* (MRCS) and the *Maximum Font Size in a segment* (MFS) (MRCS and MFS both in pixel units). An *importance attribute* is derived as a product of *font size* and *position of the text segment on the page*. The position parameter puts more weighting on segments places in the middle of

the top of a page than on segments placed on the bottom of a page. Table 1 shows the exact formulas for resolution and importance attributes for image and text segments. Even though the choice for the importance attribute for text segments is rather heuristic it performed satisfactory in the result reported in Section 6. Alternative importance measures for text segments may include a measure for bold font. For image segments alternative importance measures could weight specific objects, such as people's faces. While a layout analysis provides generally a good segmentation into text zones, image regions are either not recognized or if recognized not further characterized with respect to resolution and importance properties.

Table 1 — Attributes of text and image segments, where (S_x, S_y) = center of gravity of S , (w, h) = document width and height.

segment S	resolution $res(S)$	importance $I(S)$
type(S) = image	$res_{image}(S) = 1/r^*(S)$	$I_{image}(S) = \sum_{(i,j) \in S} \sum_{r=r^*(S)}^L B(r, (i, j))$
type(S) = text	$res_{text}(S) = MRCS/MFS(S)$	$I_{text}(S) = MFS(S) \cdot \left(1 - \frac{ S_x - w/2 }{w}\right) \cdot \left(1 - \frac{\max(S_y, h/2)}{3h}\right)$

4. MERGING IMAGE-BASED AND TEXT-BASED ANALYSIS

The goal of the merging is to combine the results from image and text analysis in order to have good characterization of text while simultaneously providing a characterization of image regions in a resolution-sensitive fashion. To achieve this goal two steps are necessary. In a first step the different image- and text-based segmentation maps have to be combined into one segmentation map. In a second step the attributes associated with each image component or text segment have to be unified. Before these steps are performed the reference grids for text segments (pixels) and image segments (code-blocks) have to be normalized.

4.1 Merging of segments

The image-based analysis from Section 2 does not distinguish text from non-text regions, i.e. an image segment can contain text and image regions, whereas a segment from the text analysis contains mainly text.

An image segment that does not contain any text area is assumed to have dominantly image content and is added to the final segment list without being analyzed any further. In the case that an area with text is represented in both a text segment and an image segment, two scenarios occur.

- 1) Text area is appropriately represented by a text segment. The image segment contains most of the text, but also some image areas. The text in the image segment may be cropped due to the coarse code block resolution.
- 2) The text segment does not contain all text of a coherent text area, e.g. when the layout analysis represents single characters of large font size as individual text zones.

In the first scenario the image segment contains *significant* text content. In the second scenario the text content is *insignificant*.

In order to represent coherent text areas as best as possible by a segment, a key decision in the merging process of image and text segments is therefore, to determine whether an image segment has significant text content redundantly represented by some of the text segments. In that case text segments is used to represent text areas. The text segments are subtracted from the image segment such that the remaining image segment contains mainly image content (e.g. background).

The details of this text-significance analysis of image segments are described in the following.

- 1) Each image segment is checked for overlap with any of the text segments. The overlap of a text component T_m and an image component I_n is defined as

$$\text{overlap}(I_n, T_m) = \min \left(\frac{\text{card}(I_n \cap T_m)}{\text{card}(T_m)}, \frac{\text{card}(I_n \cap T_m)}{\text{card}(I_n)} \right).$$

where $\text{card}(I_n)$ is the number of pixels in I_n . If an image segment I_n has $\sum_m \text{overlap}(I_n, T_m) < \epsilon_1$, it is considered to have dominantly image content and is added to the final segment list with type *image*.

- 2) In the case of a significant overlap ($\geq \epsilon_1$) with text segments a more detailed analysis of the overlap is performed. All text segments with an overlap greater than a threshold ϵ_2 are subtracted from the image segment. The result may be one connected segment with more holes, or several smaller connected segments. In order to determine the connected segments in the subtracted image a connected component analysis is performed on it. The result is a collection of image segments that do not have any significant overlap with text segments anymore. This collection is added to the final segment list with type *image*.
- 3) All text segments are also added to the final segment list with type *text*.

In the following this merging process is expressed in pseudo-code.

```

for n=1:N
  if  $\sum_m \text{overlap}(I_n, T_m) > \epsilon_1$ 
    old_I_n = I_n
    new_I_n = old_I_n
    for m=1:M
      if  $\text{overlap}(I_n, T_m) > \epsilon_2$ 
        new_I_n = old_I_n - T_m
      end
    end
    old_I_n = new_I_n
  end
  new_I_n_segments = Connected_Component_Analysis(new_I_n)
  add new_I_n_segments (type image) to merged_segment_list
else
  add I_n (type image) to merged_segment_list
end
end
for m=1:M
  add T_m segments (type text) to merged_segment_list
end

```

For the experiments described in Section 6 the thresholds ϵ_1 and ϵ_2 were tuned to $\epsilon_1 = 0.3$ and $\epsilon_2 = 0.25$.

4.2 Merging of attributes

The original importance attributes for image and text segments were computed via very different metrics (see Table 1) and are not compatible. Therefore, it is necessary to define an importance measure for the segments in the merged-segment-list that reflects the importance of a single segment with respect to the entire collection of segments. The multiresolution bit distribution derived from the J2K header data is available for all regions in the image. Therefore, it is chosen as the base element for a unified importance measure.

Since an image segment is allowed to have non-rectangular shape, but the final crop operation is being performed on segment bounding box, the bit-measure for segments is modified in the following way. Let S be a merged segment and $R(S)$ be the smallest rectangle that includes S . Furthermore define

$$P(S) = \frac{\text{card}(S)}{\text{card}(R(S))} \text{ to be the percentage of segments pixels in the rectangle } R(S) \text{ and}$$

$$E(S) = \sum_{r=\text{res}(S)}^L w_r B(r, R(S)) \text{ to be the weighted number of bits required for } R(S) \text{ at resolution } \text{res}(S).$$

Then the bit-measure for the bounding box of a segment is defined as

$$I_{\text{bounding_box}}(S) = P(S) \cdot E(S).$$

A text-importance measure is defined as

$$T(S) = \begin{cases} I_{text}(S), & \text{if type}(S) = \text{text} \\ 0, & \text{if type}(S) = \text{image} \end{cases}.$$

The unified importance measure I of a segment S is now defined as

$$I(S) = \alpha \cdot \frac{I_{\text{bounding_box}}(S)}{N_1} + \beta \cdot \frac{T(S)}{N_2}, \quad (2)$$

where $N_1 = \sum_{r=1, \dots, L} B(r; D)$ is the total entropy of the document image D , and $N_2 = \text{card}(D)$ is the image size in pixels. This measure can be interpreted as adding a bonus to text regions giving them more importance for the final composition. The parameters α and β are introduced to allow flexible weighting of text or image regions, respectively. If $\alpha=0$ only the text segments of the analysis are considered. For some applications, e.g. data base with a lot of mixed text/image documents the text bonus might be selected to be large compared to the bit-measure whereas for a dominating text-only data base the bonus might be selected to be much smaller. It should be noticed, that constant values for α and β can change the sorting order of text segments when the unified importance measure I is used. In order to avoid this effect the experiments in Section 6 were performed setting

$$\alpha = \begin{cases} 0, & \text{if type}(S) = \text{text} \\ 1, & \text{if type}(S) = \text{image} \end{cases} \quad \text{and} \quad \beta = \begin{cases} 1, & \text{if type}(S) = \text{text} \\ 0, & \text{if type}(S) = \text{image} \end{cases}.$$

The resolution attribute for text and image segments as originally defined in Table 1 is simply a scaling factor and does not need to be modified for the segments in the merged-segment list. Image segments carry the resolution from the image analysis res_{image} , text segments carry the resolution from the text analysis res_{text}

5. RESOLUTION-SENSITIVE LAYOUT FOR FINAL COMPOSITION

5.1 Resolution-sensitive layout as optimization problem

In order to fit document segments in the final display three operations are allowed to be performed on the segments: *scaling*, *cropping* and *positioning with respect to the reference location*. The quality of a possible layout fit as a combination of scaling, cropping and positioning of a collection of segments $\{S_i\}$ is measured by a cost function

$$Cost = Cost(\{S_i\}, \{s_i, B_i, p_i\}),$$

where s is the amount of scaling, B is the set discarded in the cropping operation, and p denotes the positioning. This cost function determines the distortion of the composited image from the original document. The ‘‘best fit’’ is that vector of triplets $\{(s_i^*, B_i^*, p_i^*)\}$ that is a feasible solution and results in the smallest distortion, i.e. results in the lowest cost:

$$\{(s_i^*, B_i^*, p_i^*)\} = \arg \min_{\{s_i, B_i, p_i\}} Cost(\{S_i\}, \{s_i, B_i, p_i\}). \quad (3)$$

At this point the question of what cost function to choose becomes important. The cost should increase if more scaling has to be applied, i.e. with decreasing scale factor s . The cost should also increase if more pixels are discarded in the cropping operation, i.e. when the size of the set B increases.

The cost function used in this paper is of the following type

$$Cost(S) = I/\text{importance_of_S_after_scaling_and_cropping} + \text{change_in_positioning_of_S}. \quad (4)$$

This layout problem is similar to the classical floor-plan problem⁵, but differs since it allows scaling and cropping of segments. A related problem is the layout for video key frames on paper as in ¹⁰, where the size of video frames is allowed to vary, but is determined by the shot-importance measure for key frames. The image size in ¹⁰ does not depend on the resolution properties of the image.

5.2 Greedy algorithm

Solving the optimization problem as stated in Eq.(3) in general is very complex. Therefore, a greedy approach is chosen to trade off complexity versus optimization accuracy. The greedy approach fits the segments one by one into the available space. The following presumptions are applied.

- 1) The fitting order is determined by the unified importance measure I from Eq.(2). Segments with largest importance value are fit first.
- 2) Use of the cropping operation is restricted to text segments only. For those segments cropping is defined as removing fragments following the inverse reading order, i.e. words from a text zone can be removed starting from the end of the text zone. In general, it is possible to structure image segments into fragments and imply a cropping order by allowing first to crop background and later objects, e.g. faces of people etc., but that is beyond the scope of this paper.
- 3) Cropping and scaling are mutually exclusive, i.e. either one or the other can be applied. This allows a simplification of the denominator “importance_of_S_after_scaling_and_cropping” in Eq.(4) to “importance_after_scaling*importance_of_crop_amount.”
- 4) A linear relation between importance and scaling is assumed, i.e. $importance_after_scaling = s \cdot I$.
- 5) Given a segment S and its partition into its fragments F_1, \dots, F_m , a linear relation between segment and fragment importance values are assumed: $I(S) = \sum_{k=1}^m F_k$.
- 6) Cropping and scaling are penalized equally, i.e. scaling a segment by 1/2 results in the same cost as discarding 1/2 of it.
- 7) For image and text segments a minimal scaling factor s_{min} is given in order to prevent scaling to meaningless small sizes. For text segments a lower bound on the cropped segments is given by requiring the importance of a cropped segment to be larger than a threshold ϵ (in the experiments in Section 6 the bounds were set to $s_{min} = thumbnail_scale_factor$ and $\epsilon = 0.3$).

With these presumptions, the notation used to define the cost function is now as follows.

- A segment S is given by a geometric area A and an intensity function f_A , i.e. $S=(A, f_A)$.
- Its anchor point in the original document is denoted by $q(S)$.
- Scaling is defined as multiplication with a scalar, cropping of the area A by discarding the set B as $A \setminus B$.
- The position of an area A with respect to a reference point X is denoted by $p(A, X)$.
- A placed segment S with respect to the display canvas C , i.e. $p(s(A \setminus B, f_{A \setminus B}), q(C))$, is denoted by $\Lambda(S)$.
- The metric d measures the Euclidean distance.
- The *thumbnail_scale_factor*, i.e. the scale factor that is necessary to scale the document to canvas size, is denoted by s_c .
- Segments are indexed in order of maximum importance value: $I(S_0) \geq \dots \geq I(S_N)$.
- Given a segment S_i the already placed segments are indexed by counter $\sigma(0), \dots, \sigma(k-1)$, $s(j) < i$ for $j = 0 \dots k-1$.

Using these notations, the cost function is defined as follows:

$$Cost(S_i, s_i, B_i, p_i) = \begin{cases} \infty, & \text{if (i): } s_i < s_{min}, \text{ or (ii): } I(A_i \setminus B_i, f_{A_i \setminus B_i}) < \epsilon, \text{ or (iii): } [\bigcup_{k < i} \Lambda(S_{\sigma(k-1)})] \cap \Lambda(S_i) \neq \emptyset \\ \frac{1}{(I(S_i) - I(B_i, f_{B_i})) \cdot s_i \cdot I(S_i)} + d[s_c \cdot q(A_i \setminus B_i, p(s \cdot q((A_i \setminus B_i), \Lambda(S_{\sigma(k-1)})))]), & \text{otherwise} \end{cases} \quad (5)$$

Conditions (i) and (ii) guarantee that meaningless scaling and cropping to very small size is prevented by setting the cost function to ∞ . Condition (iii) checks the available white space.

The details of the greedy algorithm are explained by the following pseudo-code.

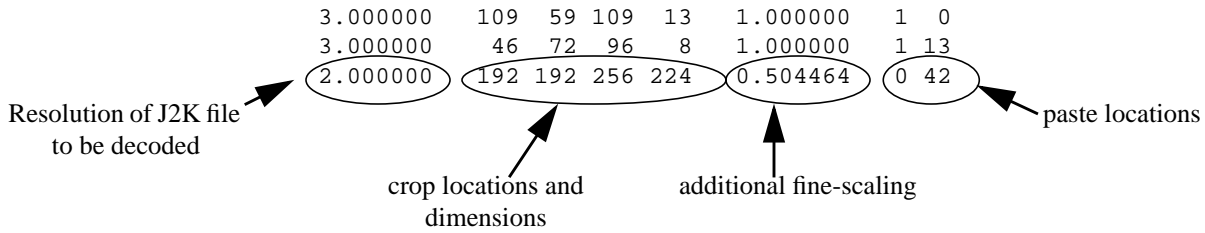


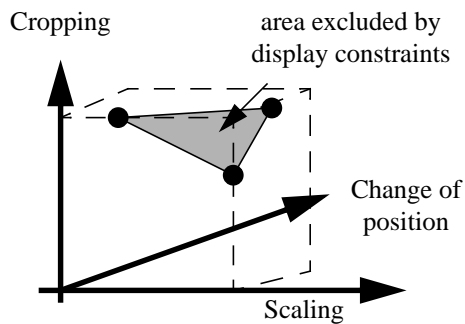
Figure 2 — Example of crop-scale-paste instruction list for three segments used to create the SmartNail (c) in Figure 4.

- Given N segments with resolution attributes r and importance attributes s , order the segments with respect to the largest importance value, resulting in the order S_0, \dots, S_N .
- Set minimal scale $s_{\min} > 0$ for segments of type text and image and set cropping threshold $\epsilon > 0$ for text segments.
- Search for first feasible solution
 $i = 0$,
while ($i < N$ && $\text{Cost}(S_i, s_i, B_i, p_i) == \infty$)
 $i++$
 $\sigma(0) = i$
- Evaluate cost function for remaining segments
 $k = 0$
for $i = \sigma(0), \dots, N-1$
 $(s_i^*, B_i^*, p_i^*) = \arg \min_{\{s_i, B_i, p_i\}} \text{Cost}(S_i, s_i, B_i, p_i)$.
if $\text{Cost}(s_i^*, B_i^*, p_i^*) < \infty$
 $\sigma(k) = i$
 $k++$
end
end
 $K = k$
- Create output list of areas $\sigma(0), \dots, \sigma(K-1)$ containing $(s_{\sigma(i)}^*, B_{\sigma(i)}^*, p_{\sigma(i)}^*)$.

The output of this layout algorithm is a crop-scale-paste instruction list that is being computed without any decoding of the original image data. That data have to be accesses only for the final image creations. Figure 2 shows an example of a crop-scale-paste instruction list for the SmartNail in the top right corner of Figure 4.

5.3 Three-dimensional parameter space for layout operations

In order to fit document segments in the final display three operations are allowed to be performed on the segments: scaling, cropping and positioning with respect to the reference location. These three operations are sufficient to express a variety of common image processing tasks for documents. As an example, traditional scaling is performed by not allowing any cropping or change in positioning, but only allow to alter the scaling parameter. In contrast, text reflow is performed by changing relative positioning between text segment while neither allowing scaling or cropping. The table in Figure 3 shows a characterization of common document processing operations in terms of the three parameters. The diagram gives a schematic overview of the three-dimensional parameter space. The display size constrains the space, simulated by the shaded area. Layout is possible for all combinations inside the constrained 3-dimensional parameter space.



	reflow	cropping	traditional scaling	white space removal	re-layout	enlargement	change of aspect ratio	summarization
scale	1	1	γ	1	1	γ	γ	γ
crop	0	β	0	0	0	0	0	β
change in position	α	0	0	α	α	α	α	α

Figure 3 — Three-dimensional parameter space for layout operation (left) and table of document processing tasks expressed in the three-dimensional space (right). The parameters α , β , and γ have values in $[0,1]$.

6. EXPERIMENTS AND RESULTS

This section starts with an example in order to explain the different techniques derived in the previous sections. The example document (left in Figure 4) is reformatted for a display of 130x160 pixels. Figure 4 shows the results applying the greedy layout algorithm optimizing the cost function from Eq. (5) using the image-only, the text-only, and the merged analysis results. The final images are called *SmartNails* in contrast to traditionally scaled thumbnails. In the image-based SmartNail segments (connected components of the segmentation map in Figure 1) with highest importance values contain image and text. In the text-based SmartNail, text is reflowed by repositioning of fragments. Repositioning is necessary to keep the text at a readable size. The SmartNail produced by the merged analysis keeps an image segment and one text segment with repositioned and cropped fragments. Figure 4 shows also SmartNails for two additional display sizes. Text reflow based on bounding boxes of layout analysis was already introduced in ³ with a final layout based on reading order only, but not on combinations of scaling, cropping, and positioning of the text segments

The parameters in the merging and layout process from Sections 4 and 5 used to create the SmartNails in Figure 4 were the ones that were tuned for a data base of 100 compound document images (grayscale) at a resolution of 300dpi. The data set contained technical articles, letters, memos, magazine pages, brochures, etc. Figure 5 shows a SmartNails and traditional thumbnails for a sample selection of 12 documents fitting display size 130x160.

In documents containing text-only in font style and size that is recognized well by state-of-the art OCR systems one could argue that displaying OCR results in ASCII form should be sufficient. However, initial subjective evaluations by users using image-based reflow showed that the image appearance of text portions was considered to be very pleasing. Furthermore – as already mentioned in ³ – image-based reflow avoids visible character recognition errors and does not require detailed language knowledge. For document images containing text that is hard to be recognized by an OCR system, e.g. text in graphic font style, large size font, text on image background etc., the text-only method would result in unsatisfactory results since logical groups of characters are typically not recognized correctly. In those cases – examples are shown in Figure 5 – the combined text and image analysis forms more coherent units of text and image content. In the case that the text-based analysis fails completely, the image analysis will still provide a resolution-sensitive thumbnail representation. That way the presented SmartNail approach is a very universal one that is suitable for working on heterogeneous document collections.

In an initial investigation on computational complexity of the algorithms it could be concluded that the complexity of the Smartnails algorithms applied for 300dpi originals was twice the complexity of traditional downsampling, whereas for 600dpi originals SmartNail and traditional thumbnail complexity were in the same range. Detailed results on the computational complexity of the SmartNail algorithms and their sensitivity to dpi-resolution of the original document are reported in ².

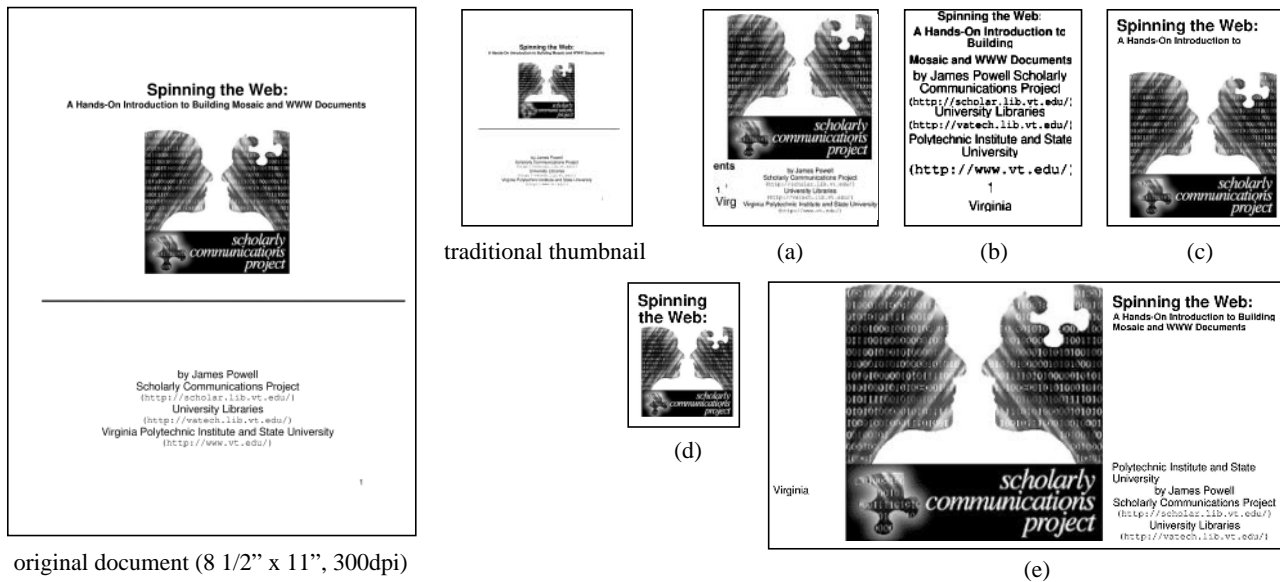


Figure 4 — Original document (left, scaled by 1/8), a traditional thumbnail of size 130x160, and SmartNails with various parameter settings and for various display sizes: (a) 130x160, image-based analysis only, (b) 130x160, text-based analysis only, (c) 130x160, merged analysis, (d) 80x100, merged analysis, (e) 400x200, merged analysis.

When looking at the SmartNails in Figure 4 the question arises whether it is useful for a user to substitute regular thumbnails with SmartNails since the original document layout is likely to be changed. A formal user study has not been completed yet, but usability was discussed informally by having several people use a browsing scenario displaying thumbnails and SmartNails. In the case that a user had seen a document before SmartNails seemed to be less useful as a replacement of thumbnails, but more useful as an additional pop-up view. A pop-up view has been used in the literature in user interface design and in commercial products (e.g. fish eye views). A thumbnail-specific application can be found in ⁹, where the authors reported on experiments using the keyword-enhanced thumbnails from ¹¹ in pop-up views. A screen shot of a SmartNail in a pop-up view is shown in Figure 6. The legibility of text was seen as an important feature by the users, as well as maintaining the reading order.

7. CONCLUSIONS

In this paper novel techniques for resolution-sensitive analysis and layout of compound document images are introduced that allow flexible repurposing of documents for constrained displays such as thumbnails or PDA displays. The final image, called a SmartNail, consists of a selection of cropped and scaled document segments that are recomposed in order to fit the available display space while maintaining recognizability of image features and readability of text and keeping the layout close to the original document layout. The resolution-sensitive analysis combines image-based multiresolution analysis derived from header data of a JPEG 2000 codestream with text-based document layout analysis information. Resolution and importance attributes are assigned to text and image regions. The layout step operates in a three-dimensional parameter space, constrained by the display size, allowing scaling, cropping and repositioning of segments. Since no substitution of text by ASCII characters is used, SmartNails do not show OCR errors. The image analysis of JPEG 2000 header data only (approximately 5% of all compressed data) allows fast processing of print resolution images. An obvious direction for future research is a more detailed investigation of parameter choices for specific classes of documents in combination with user studies.

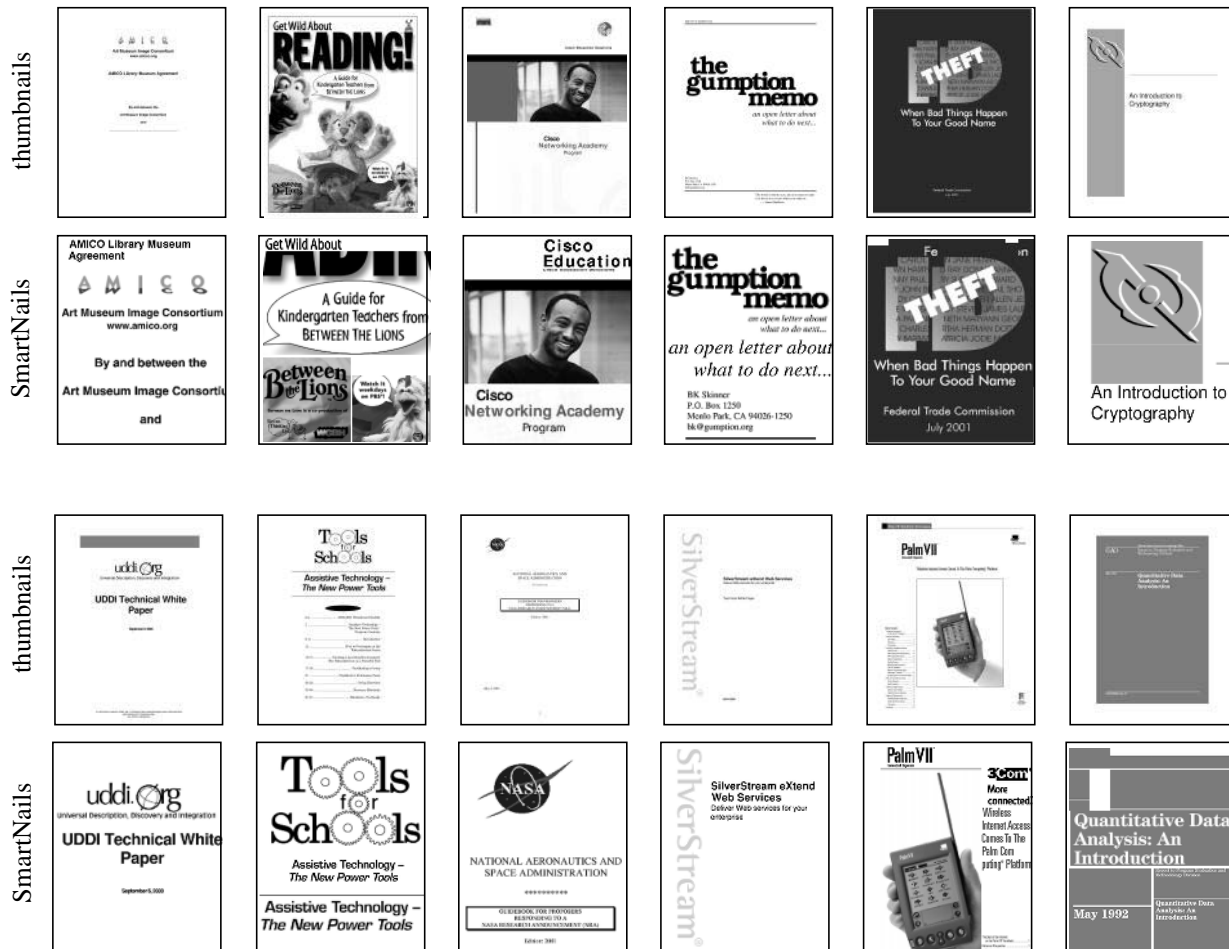


Figure 5 — Twelve example document thumbnails and SmartNails fitting a display size of 130x160 pixels.

REFERENCES

1. Alam, H., Rachmat, H., Kumar, A., Rahman, F., Tarnikova, Y., Wilcox, C., "Web Page Summarization for Hand-held Devices: A Natural Language Approach," Proc. of ICDAR'03, vol. 2, pp. 1153-1157, 2003.
2. Berkner, K., Schwartz, E.L., "Resolution-sensitive document image analysis for document repurposing," to appear in Proc. of 37th Asilomar Conference in Signal, Systems, and Computers, Nov. 2003, Pacific Grove, CA.
3. Breuel, T., Janssen, W., Popat, K., Baird, H., "Paper to PDA," Proc. 16th ICPR, vol. 4, pp. 476-479, Quebec City, Canada, 2002.
4. ITU-T Rec. T.800 | ISO/IEC 15444-1:2000, Information Technology - JPEG 2000 Image Coding System.
5. Meller, R.D., Gau, K.-Y., "The Facility Layout Problem: Recent and Emerging Trends and Perspectives," Journal of Manufacturing Systems, vol. 15, no. 5, pp. 351-366, 1996.
6. Neelamani, R., Berkner, K., "Adaptive Representation of JPEG 2000 Images using Header-based Processing," Proceedings of ICIP2002, vol. 1, pp. 381-384, Rochester, NY, 2002.
7. Ogden, W., Davis, M., Rice, S., "Document thumbnail visualizations for rapid relevance judgements: When do they pay off?," in TREC 1998, pp. 528-534, 1998.
8. Suh, B., Ling, H., Bederson, B.B., Jacobs, D., "Automatic Thumbnail Cropping and its Effectiveness," UIST 2003, ACM Symposium on User Interface Software and Technology, CHI Letters, 5(2), to appear.

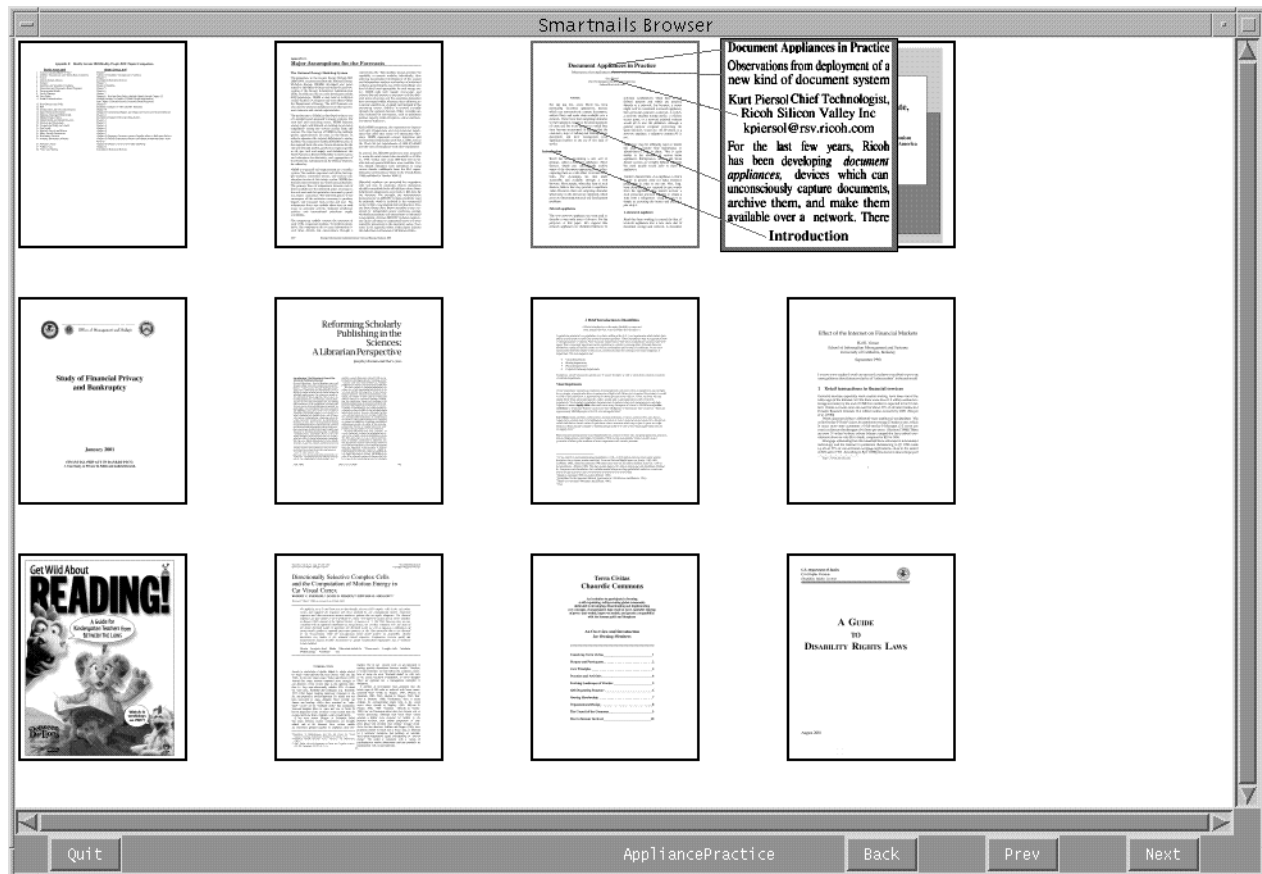


Figure 6 — Screen shot with SmartNail in pop-up view. The lines point to locations of text segments in the traditional thumbnail.

9. Suh, B., Woodruff, A., Rosenholtz, R., Glass, A., “Popout Prism: Adding Perceptual Principles to Overview+Detail Document Interfaces,” Proc. of CHI’02, pp. 251-258, 2002.
10. Uchihasi, S., Foote, J., “Summarizing video using a shot importance measure and a framepacking algorithm,” Proceedings of ICASSP, pp. 3041-3044, 1999.
11. Woodruff, A., Faulring, A., Rosenholz, R., Morrison, J., Pirol, P., “Using Thumbnails to Search the Web,” Proc. of SIGCHI’01, 2001.
12. Wu, J., Lopresti, D., “Resource-Optimized Delivery of Web Images to Small-Screen Devices,” Proceedings of SPIE, vol. 5010, pp. 144-155, 2003.